PERSONAL COMPUTER WORLD

VOLUME 1, NUMBER 1 1978

50p U.S.$1.50

PERSONAL POWER IS HERE

THE GATES OF REASON

THE GINGERBREAD MAN'S COMPUTER

BASIC PONTOON

THE ELEGANT MINMON

Europe's first magazine for personal computers for home and business use

NASCOM 1
— NEW
— BRITISH
— PERFECTLY AT HOME

WIN A NASCOM 1 TWO PROGRAMMABLES BOOKS • Details Inside

THIS YEAR, GET YOUR OWN COMPUTER

Bywood Electronics is pleased to be advertising in this, the first issue of 'Personal Computer World', a magazine which, we feel, has a great future and one which will go a long way to educate the engineers and hobbyists in a field of technology which has both present and future applications.

Bywood Electronics is one of industry's leading experts in the technology of clock circuits and their application. With some 72 different circuits, all of which are coupled to displays, they can be used as a teaching tool, or by the enthusiastic amateur. This system is called Monolithic Horometric Integration (MHI), a term which originated at Bywood's and is now accepted nationally as a term for describing digital clock technology and peripheral applications.

Bywood Electronics was the first company to take some of the mystique out of microprocessor usage. The company has designed and produced what are undoubtedly two of the lowest cost microprocessor evaluation kits, SCRUMPI I and II, to appear on the market. Designed around the National Semiconductor SC/MP microprocessor, the kits give invaluable hands-on experience to engineers of all levels of expertise.

Bywood Electronics, distributes all the support devices for the SC/MP microprocessor — RAMs, ROMs etc, and also provides a PROM programming service. The company operates a large software library which includes device information, circuit design for the clock kits, SA400 mini floppy diskette storage drive manuals and the SC/MP support manuals.

CLOCK CHIPS & KITS

TYPE	SPECIAL FEATURES	£CHIP	£KIT
MM5309	7 seg + BCD. RESET ZERO	8.53	12.50
MM5311	7 seg + BCD	4.26	8.00
MM5312	7 seg + BCD 4 DIGIT ONLY	5.65	
MM5313	7 seg + BCD	6.50	
MM5314	7 seg + BASIC CLOCK	4.26	7.00
MM5315	7 seg + BCD RESET ZERO	6.50	
MM5316	Non-mpx ALARM	7.50	
MM5318	7 seg + BCD External digit select	4.93	8.00
MM5371	ALARM. 50 Hz	12.19	
MM5378	CAR Clock. Crystal control. LED	9.86	14.00
MM5379	CAR Clock. Crystal control. Gas discharge	9.86	
MK5025	ALARM. SNOOZE	5.60	9.00
MK50395	UP/DOWN Counter — 6 Decade	12.10	15.10
MK50396	UP/DOWN Counter — HHMMSS	12.10	15.10
MK50397	Up/DOWN Counter — MMSS.99	12.10	15.10
FCM7001	ALARM. SNZ. CALENDAR. 7 seg	9.00	12.50
FCM7002	ALARM. SNZ. CALENDAR. BCD	9.00	
CT7003	ALARM. SNZ. CALENDAR. Gas discharge	9.00	
FCM7004	ALARM. SNZ. CALENDAR. 7 seg	9.00	12.50
AY5. 1202	7 seg. 4 digit	4.76	
AY5. 1230	7 seg. ON and OFF ALARM	5.25	TBA

All above clock kits include clock PC board, clock chip, socket and CA3081 driver IC. MH15378 also includes crystal and trimmers. When ordering kit, please use prefix MHI, e.g. MHI 5309.

BITS & BYTES

74C00	Quad NAND	0.25	
74C04	Hex Inverter	0.25	
74C10	Triple NAND	0.25	
74C42	BCD Decoder	0.95	
74C157	Quad Selector	2.25	
74C163	4 bit counter	1.15	
74C164	PISO register	1.15	
74C165	SIPO register	1.15	
74C173	3S Quad latch	0.95	
74LS139	Dual 2-4 Dec	1.50	
DM8095	3S Hex buffer	1.75	
DM8096	Inv 8095	1.75	
DM81LS95	3S 8 bit buff	1.45	
DM81LS96	Inv 95	1.45	
DM81LS97	3S 4+4 buffer	1.45	
DM81LS98	Inv 97	1.45	
MM2102-2	1Kx1 RAM	2.11	
MM2112-2	256x4 RAM	3.08	
MM74C920	256x4 CMOS RAM	11.83	
XX2114	1Kx4 RAM	24.00	
MM1702Q	256x8 EPROM	11.90	
MM5204Q	512x8 EPROM	10.95	
MM2708Q	1024x8 EPROM	31.15	
	EPROM prices for blank devices		
ER3401	1024x4 EAROM	28.85	
MM5307AA	Baud Rate Gen	12.68	
MM5303	(AY-5-1013) UART	6.34	
Xtal for 5307		TBA	
DM8678	Char Gen	15.20	
	(both CAB & BWF avail.)		

CLOCK MODULES

LT601 Alarm Clock Module, similar to MA1002	6.00
MTX1001 Transformer	0.90

PAYMENT TERMS

Cash with order, Access, Barclaycard (simply quote your number). Credit facilities to accredited account holders. 15% handling charge on goods ordered and paid for then cancelled by customer.
All prices exclude 8% VAT PLEASE SEND 30p POST AND PACKING

DISPLAYS

DL707, 704, 701 0.3"	1.70	Litronix class 2 product	
DL727, 728, 721 0.5" (2 dig.)	4.31	DL707E	0.85
		DL727E (2 dig.)	2.00
DL747, 750, 746 0.6"	2.82	DL747E	1.80

MHI DISPLAY KITS

MHI707/4	digit 0.3"	7.60	MHI707E/4	4.30
MHI707/6		11.00	MHI707E/6	5.70
MHI727/4	0.5"	9.70	MHI727E/4	5.30
MHI727/6		13.80	MHI727E/6	7.20
MHI747/4	0.6"	11.40	MHI747E/4	7.20
MHI747/6		17.30	MHI747E/6	9.90

Any one or two of the above MHI display kits will interface directly with any of the MHI clock kits

CASES (with perspex screen)

VERO 1. 8" x 5½" x 3"	3.00
VERO 2. 6" x 3¼" x 2¼"	3.00

SOCKETS

24, 28 or 40 pin	0.60
Soldercon strip skts. 50 pins	0.30

We stock:

E.T.I. System 68 MPU kits, VDU kits, Case kit. Comprehensive selection of Hardware and Software support. Digital clock chips, kits, displays. Please send SAE for our catalogue.

Products from:

Fairchild, General Instruments, Liton, Litronix, Mostek, Motorola, National Semi, S.M.C., Vero.

BYWOOD

BYWOOD ELECTRONICS
68 Ebberns Road
Hemel Hempstead
Herts HP3 9QRC
Tel. 0442 62757

PERSONAL COMPUTER WORLD

UK 50p
US $1.50

Vol 1, No 1
1978

Europe's first magazine for personal computers for home and business use

Editorial and Advertising Office:

62A Westbourne Grove
London W2
Phone: 01-229 5599

CONTRIBUTORS:

We welcome interesting articles written simply and clearly. You need not be a specialist to write for us. MS should not be more than 3000 words long, lines double spaced, with wide margins. Line drawings and photographs wherever possible. Enclose a stamped self-addressed envelope if you would like your article returned.

Manufacturers, suppliers and dealers are welcome to contribute technical articles, and send product information, but we are pledged to an independent viewpoint and will publish evaluations and reasoned criticism or praise, space permitting. Naturally there will be right of reply. Views expressed in articles are not necessarily those of Personal Computer World.

We may make arrangements to offer our readers products at special prices, for a limited period, in line with the policy outlined above.

We would like to add to our pool of consultants. We value independence and objectivity, and if you think that you would be able to help us, write and let us know something about your interests and background.

Published bimonthly by Intra Press, 62A Westbourne Grove, London W2. Subscription rates: Britain and the Continent — £4 for six issues, £8 for 12 issues. Prices include postage. USA — $10 for six issues, $20 for 12 issues. Prices include postage. Printed by The Yale Press Ltd, Delga House, Carmichael Road, South Norwood, London SE25 5LY. Sole UK Distributors: Seymour Press Ltd, 334 Brixton Road, London SW9, England. Distribution to specialist shops and abroad by Intra Press.

Publisher's Letter

Dear Reader,

Let me tell you about some of our plans for the future. One, go monthly as soon as possible this year. Two, publish reasonably priced books on personal computing. Three, organise non-profit courses for absolute beginners. Four — and this needs careful thought and planning — hold seminars and exhibitions with the emphasis on personal computers for the home and really small business, priced reasonably to exhibitors and the public, so that we encourage even the smallest concern to put its ideas and products forward. There are many bright people who do not have the opportunity to show how good they are. We are in business to give them the opportunity to do just that.

I would like to emphasize as strongly as I can that any projects we undertake will be presented at the lowest possible price to the public, giving everyone a chance to enter the personal computer world. We expect readers to interact with us, shaping and improving this magazine. In the personal computer world, we need each other.

Happy computing!

PERSONAL COMPUTER WORLD

Europe's first magazine for personal computers for home and business use

CONTENTS

The BOOK PAGE has been held over until the next issue.

THE NEWBEAR COMPUTING STORE

Hardware Components Section

Memories

2102-1	£1.50
2102L-1	£1.60
21L02	£1.60
2112	£3.04
4027 (250ns)	£6.00
2114 (450ns)	£10.00
4116 (250ns)	t.b.a.

Please enquire for 16K dynamic and 100 up prices.

Motorola Microcomputing IC's

MC6800P	£14.00
MC6820P	£6.20
MC6850P	£6.74
MC6810AP	£3.61
MC6830L7	£11.33
MC8602P	£2.88
MC14536P	£3.69
MC3459	£2.53

Zilog Microcomputing IC's

Z-80 CPU 2.5MHz	£15.50
Z-80 P10 PS	£10.00
Z-80 CTC PS	£10.00

Microprocessors

8080A	t.b.a.
SC/MP Mk II	t.b.a.
650z	1493
F-8	t.b.a.
2650	t.b.a.

Interfacing IC's

MC1488P	£1.40
MC1489P	£1.40
75150P	£1.30
75150N	£1.20
75154	£2.50
4N33	£1.95

For V.24, RS232C.

Buffers

81LS95	£1.43
81LS96	£1.43
81LS97	£1.43
81LS98	£1.43
8T26	£1.84
8T95	£1.60
8T97	£1.60
74367	£1.30

Standard TTL

7400	16
7402	16
7403	16
7404	24
7406	27
7410	18
7420	18
7430	18
7432	34
7437	36
7440	18
7454	18
7472	28
7474	34
7486	34
7493	57
7495	65
74107	34
74132	52
74151	70
74153	74
74155	78
74157	68
74163	88
74164	£1.00
74175	£1.30
74193	£1.30
74283	£1.10

74LS00	.18
74LS01	.21
74LS02	.21
74LS03	.21
74LS04	.26
74LS05	.26
74LS08	.21
74LS09	.21
74LS10	.21
74LS11	.26
74LS12	.21
74LS13	.55
74LS14	1.26
74LS15	.21
74LS20	.21
74LS21	.26
74LS22	.21
74LS26	.31
74LS27	.21
74LS28	.40
74LS30	.29
74LS32	.24
74LS33	.40
74LS37	.30
74LS38	.30
74LS40	.27
74LS42	.88
74LS47	.96
74LS48	.96
74LS51	.21
74LS54	.21
74LS55	.21
74LS73	.34
74LS74	.38
74LS75	.55
74LS76	.34
74LS78	.34
74LS83	1.05
74LS86	.38
74LS107	.38
74LS109	.38
74LS112	.38
74LS113	.38
74LS114	.38
74LS125	.56
74LS126	.56
74LS132	.90
74LS136	.38
74LS138	1.05
74LS139	1.05
74LS151	.96
74LS153	.96
74LS154	1.98
74LS155	1.05
74LS156	1.05
74LS157	.96
74LS158	.88
74LS160	1.22
74LS161	1.22
74LS162	1.22
74LS163	1.22
74LS164	1.18
74LS168	2.03
74LS169	2.03
74LS170	2.61
74LS173	2.28
74LS174	1.18
74LS175	1.10
74LS189	2.82
74LS190	2.02
74LS191	2.02
74LS193	1.86
74LS196	1.21
74LS197	1.21
74LS247	.96
74LS248	.96
74LS249	.96
74LS253	1.05
74LS257	1.09
74LS258	1.09
74LS266	.38
74LS283	.96
74LS289	1.05
74LS352	.96
74LS353	.51
74LS365	.51
74LS366	.51
74LS367	.51
74LS368	.51
74LS386	.38
74LS670	3.04

All IC's are 100% guaranteed and manufactured by Texas, Signetic, National etc.

Connectors & Sockets

Subminiature D Type

9 way plug	.58
9 way socket	.68
9 way cover	.74
15 way plug	.74
15 way socket	1.09
15 way cover	.83
25 way plug	.92
25 way socket	1.50
25 way cover	.85
37 way plug	2.10
37 way socket	3.11
37 way cover	1.19

Edge Connectors

80 way 0.1" single sided	1.80
45 way 0.1" single sided	1.30
43 way 0.156" double sided *(for Motorola exorcisor, D1, D2 kit bus)*	4.20
64 way plug for Eurocard to DIN 41612	1.65
64 way socket for Eurocard to DIN 41612	2.99
10 way plug Molex for S-50 bus	.30
10 way socket Molex for S-50 bus	.32

D-I-L Sockets (low profile gold plated)

14 pin	.24
16 pin	.25
24 pin	.36
40 pin	.66

D-I-L plugs

16 pin	.60
24 pin	1.20

Scotchflex connectors

40 way plug	2.40
40 way socket	2.62

Printed Circuit Boards

For: 77-68 (8"x8" 0.1" edge 78 way)	
CPU board	10.00
4K RAM board	10.00
Bootstrap loader board	10.00
Prototyping p.c.b. (low pack density)	9.95
Prototyping p.c.b. (high pack density)	10.91
For: E-77 (double euro card)	
4K RAM board	6.25
Prototyping p.c.b. (low pack density)	7.52
For: 3U 43 way systems (4½"x8")	
2K RAM board	8.50
Back plane (Vero) board approx. 10"x4"	1.47
For: S-50 bus	
4K RAM board	15.00
For: S-100 bus	
4K RAM board	15.00

Miscellaneous

Subminiature switches

Single pole change over toggle	.69
Single pole change over (momentary action)	1.00
D-I-L 8 way SPST	1.32

LED's 0.2" High Luminance

Red	.15
Green	.18
Yellow	.18
Orange	.18

Crystals

10.000	2.60	5.000	2.60
4.433619MHz			1.50*
1.000MHz			3.50

10 way ribbon cable stranded	.20/foot
10 way ribbon cable solid	.20/foot

Computing Stationery

C30 Cassettes Memorex MrX2 .75

Coding Form Pads

Lightweight Hexadecimal (for 6800 etc)	.80
Heavyweight Hexadecimal (for 6800 etc)	1.10
Lightweight Hexadecimal (for Z-80)	.80
Heavyweight Hexadecimal (for Z-80)	1.10
8 hole paper tape. 8" spool	1.00
8 hole paper tape. Fan fold.	2.00
Teletypewriter paper roll	1.00
7" floppy discs IBM compatible	4.00

*12½% VAT

Goods are normally shipped within 24 hours.
Barclay card & Access VAT at 8% for Hardware Components.
30p postage and packing unless otherwise stated.
Cheques to be made out to 'The Newbear Computing Store'

Send for an up-to-date catalogue to:

The Newbear Computing Store

7 Bone Lane, Newbury.
Tel. 0635-46898.

Callers welcome Monday to Saturday 9.00a.m.-5.30p.m.
The Newbear Computing Store is a division of Newbury Laboratories Ltd.

Last minute product news

Last minute product news: The Mind Reader, a sharp little electronic secretary from *Optimisation Ltd* . . . the AMI-COS, a well thought-out modular 6800 system from *Tirro Electronics* . . . *Rockwell International* shows its hand through *Pelco Electronics* and the famous little KIM-1 computer . . . the PCM-12, a minicomputer for the upper end of small business applications, marketed by *Services to Computer Industry Ltd* . . . the enhanced power of the North Star minifloppy offered by *Comart* . . . the TAP-A-DIP, presented by the *British Central Electrical Co* (human beings are a *lot* more original than computers) tests dual inline packages . . . The *Micronics Company* with its MICROS, a Z80 based system, makes a bid for a share of the market in personal computing . . . *Maplin* the mail-order house, markets a beginner's computer based on the SC/MP microprocessor . . . some of these will be mentioned in more detail in our PRODUCTS PAGE in Vol 1 No 2, along with news of other products.

★After being mentioned in the *Sunday Times* colour supplement we received eye-opening letters from all kinds of alert people: one writer is disabled and can only type holding a pencil in his mouth: can microprocessors aid the disabled? . . . another writer wants to know how to use small computers for calculating the daily feed rations for a herd of 120 dairy cows . . . small businessmen suddenly realise that personal computers are for them, and welcome *PCW* . . . letters from universities, technical colleges, schools, libraries, town halls, accountants, chemists, bakers, butchers, design houses . . . letters from gentlemen of means . . . gentlemen without means . . . mean gentlemen (a year's subscription to *PCW* **free**?) . . . hundreds of letters more.

★A new Computer Workshop Ltd. is opening in Manchester. Principal is Adam Wiseberg, and the address is: 29, Hanging Ditch, Manchester M4 3ES.

★General Instrument Microelectronics, which has what is claimed to be the largest and most advanced MOS-LSI microcircuit facility (in Glenrothes, Scotland), has outgrown its offices and is moving to new ones at Regency House, 1-4 Warwick St, London W1. This after "six years of steady expansion."

★The *Research Machines* 380Z Computer, distributed by *Sintel,* will be evaluated for our next issue by PCW consultant Mike Dennis.

★Please — don't leave things till the last minute. We're pretty flexible, and our printers cringe when we rush at them with scraps of late, late news. But even we can't insert something that's arrived after printing. So if you have interesting things to send (letters, articles, exhibition news, product news, advertisements) send them now. Early birds get to enjoy magnificent sunrises.

★Help us to be prompt. Enclose a self-addressed stamped envelope whenever you write to us.

PCW THE SEARCHER PCW

This routine is written for an M6800 system with MIKBUG compatible monitor such as MIKBUG, SWTBUG or SMARTBUG. It enables one to search a block of memory for a specified byte sequence.

After the program is loaded, the first address to be searched is typed into memory locations $A002 and $A003, and the last address to be searched is typed into memory locations $A004 and $A005.

The program itself starts at location $3F30 so it fits just at the top of a 16 k system, though it is easily relocated.

The program will prompt with ENTER THE BYTES at which time up to 15 bytes may be entered, terminated with a CARRIAGE RETURN. Two sample printouts are shown.

This routine is a small part of a comprehensive monitor and operating system which will be published in a future issue of PERSONAL COMPUTING WORLD.

f.

```
ENTER THE BYTES: 45 4E

0217 41 50 50 45 4F 44 00 12 DD 41    APPEND...A
02DC 14 2A 52 45 4E 55 4D 42 45 52    .*RENUMBER
02E7 06 A5 52 45 4E 00 06 A5 52 45    ..REN...RE
0BFE 53 20 52 45 4E 55 4D 42 45 52    S RENUMBER
0C7B 4F 54 20 45 4E 4F 55 47 48 20    OT ENOUGH

ENTER THE BYTES: A6 00 81 0D 26

07FB 97 27 0D A6 00 81 0D 26 F5 5A    .'.....&.Z
0F3F 97 95 08 A6 00 81 0D 26 2C DE    .......&,.

ENTER THE BYTES:
```



```
00001                        NAM       SEARCH
00002                        OPT       L,M,NOS,P,NOG
00003 3F30                   ORG       $3F30
00004 3F30 BD 3FD5 SE12      JSR       NEWLIN
00005 3F33 CE 3FDB           LDX       £PROMPT  This prints the
00006 3F36 BD E07E           JSR       PDATA1   ENTER THE BYTES prompt
00007                * FIRST is the memory location where the list
00008                * of bytes to be searched for starts
00009 3F39 CE A020           LDX       £FIRST
00010 3F3C FF A018           STX       BYTPTR
00011                * BYTPTR is a pointer to the byte list
00012 3F3F 5F                CLR B
00013                * The B accumulator counts up the numter of bytes
00014                * in the list, and the number is stored in the
00015                * location labelled BYTCNT
00016                * The next routine, called MORE, gets in
00017                * the list of bytes and stores them until
00018                * CARRIAGE RETURN ($0D) ic hit
00019 3F40 BD E1AC MORE      JSR       INCH
00020 3F43 81 0D             CMP A     £$0D
00021                * When we have GOT'EM all, the byte count
00022                * is stored and two new lines are printed
00023 3F45 27 17             BEQ       GOTTEM
00024 3F47 BD E0AC           JSR       INHEX+2
00025 3F4A 37                PSH B
00026 3F4B BD E057           JSR       BYTE+2
00027 3F4E 33                PUL B
00028 3F4F FE A018           LDX       BYTPTR
00029 3F52 A7 00             STA A     X
00030 3F54 08                INX
00031 3F55 FF A018           STX       BYTPTR
00032 3F58 BD E0CC           JSR       OUTS
00033 3F5B 5C                INC B
00034 3F5C 20 E2             BRA       MORE
00035 3F5E F7 A015 GOTTEM    STA B     BYTCNT
00036 3F61 8D 72             BSR       NEWLIN
00037 3F63 8D 70             BSR       NEWLIN
00038 3F65 FE A002           LDX       BEGA
00039                * BEGA contains the address of the first
00040                * location in memory that is to be searched
00041                * PRESA contains the present address being
00042                * searched
00043                * ENDA contains the last address to be
00044                * searched
00045                *
00046                * This bit of code is self modifying
00047                * The X register points to PRESA. First
00048                * the byte pointed to by X is compared
00049                * with the byte FIRST, if they are the same
00050                * then PRESA+1 is compared to FIRST+1
00051                * and so on until BYTCNT bytes have been
00052                * matched.  If they don't match the
00053                * program branches to NOMACH.
```

```
00055 3F68 F6 A015 SE0     LDA B      BYTCNT
00056 3F6B 7F 3F7B         CLR        SE1+1
00057 3F6E FF A016         STX        PRESA
00058 3F71 CE A020         LDX        #FIRST
00059 3F74 FF 3F7D         STX        SE2+1
00060 3F77 FE A016         LDX        PRESA
00061 3F7A A6 00    SE1    LDA A      X
00062 3F7C B1 A020 SE2     CMP A      FIRST
00063 3F7F 26 49           BNE        NOMACH
00064 3F81 7C 3F7B         INC        SE1+1
00065 3F84 7C 3F7E         INC        SE2+2
00066 3F87 5A              DEC B
00067 3F88 26 F0           BNE        SE1
00068 3F8A FF A016         STX        PRESA
00069                 * The bytes have been found, so the address
00070                 * is output
00071 3F8D CE A016         LDX        #PRESA
00072 3F90 BD E0C8         JSR        OUT4HS
00073                 * The pointer is moved back three places
00074 3F93 FE A016         LDX        PRESA
00075 3F96 09              DEX
00076 3F97 09              DEX
00077 3F98 09              DEX
00078                 * and 10 bytes are printed in the 2 HEX
00079                 * characters and 1 space format
00080 3F99 C6 0A           LDA B      #10
00081 3F9B BD E0CA SE50    JSR        OUT2HS
00082 3F9E 5A              DEC B
00083 3F9F 26 FA           BNE        SE50
00084                 * 3 spaces are printed
00085 3FA1 BD E0CC         JSR        OUTS
00086 3FA4 BD E0CC         JSR        OUTS
00087 3FA7 BD E0CC         JSR        OUTS
00088                 * The same happens again but with an
00089                 * ASCII printout if the code is not less
00090                 * than $20 (space) but is less than $7B
00091                 * (curly brackets)
00092 3FAA FE A016         LDX        PRESA
00093 3FAD 09              DEX
00094 3FAE 09              DEX
00095 3FAF 09              DEX
00096 3FB0 C6 0A           LDA B      #10
00097 3FB2 A6 00    SE53   LDA A      X
00098 3FB4 81 20           CMP A      #$20
00099 3FB6 2D 04           BLT        SE51
00100 3FB8 81 7B           CMP A      #$7B
00101 3FBA 2D 02           BLT        SE52
00102                 * if it is nonprintable a full stop is
00103                 * substituted
00104 3FBC 86 2E    SE51   LDA A      #'.
00105 3FBE BD E1D1 SE52    JSR        OUTCH
00106 3FC1 08              INX
00107 3FC2 5A              DEC B
00108 3FC3 26 ED           BNE        SE53
```



```
00109                 * A NEW LINE command is issued and, if
00110                 * ENDA has not been reached, the search
00111                 * is resumed.
00112 3FC5 8D 0E           BSR        NEWLIN
00113 3FC7 FE A016         LDX        PRESA
00114 3FCA 08       NOMACH INX
00115 3FCB BC A004         CPX        ENDA
00116 3FCE 26 98           BNE        SE0
00117 3FD0 8D 03           BSR        NEWLIN
00118 3FD2 7E 3F30         JMP        SE12
00119 3FD5 CE 3FED NEWLIN  LDX        #CRLF
00120 3FD8 7E E07E         JMP        PDATA1
00121                 *
00122                 * TEXT STRINGS
00123                 *
00124 3FDB 45       PROMPT FCC        /ENTER THE BYTES: /
00125 3FEC 04              FCB        4
00126 3FED 0D       CRLF   FCB        $D,$A,0,0,0,0,4
00127                 *
00128                 * EXTERNAL EQUATES
00129                 *
00130       E055    BYTE   EQU        $E055
00131       E07E    PDATA1 EQU        $E07E
00132       E0AA    INHEX  EQU        $E0AA
00133       E0C8    OUT4HS EQU        $E0C8
00134       E0CA    OUT2HS EQU        $E0CA
00135       E0CC    OUTS   EQU        $E0CC
00136       E1AC    INCH   EQU        $E1AC
00137       E1D1    OUTCH  EQU        $E1D1
00138       A002    BEGA   EQU        $A002
00139       A004    ENDA   EQU        $A004
00140       A015    BYTCNT EQU        $A015
00141       A016    PRESA  EQU        $A016
00142       A018    BYTPTR EQU        $A018
00143       A020    FIRST  EQU        $A020
00144              END

TOTAL ERRORS 00000
```

```
S00B0000534541524348202OFE
S11E3F30BD3FD5CE3FDBBDE07ECEA020FFA0185FBDE1AC810D2717BDE0AC3764
S11E3F4BBDE05733FEA018A70008FFA018BDE0CC5C20E2F7A0158D728D70FEA7
S11E3F66A002F6A0157F3F7BFFA016CEA020FF3F7DFEA016A600B1A02026497E
S11E3F817C3F7B7C3F7E5A26F0FFA016CEA016BDE0C8FEA016090909C60ABD48
S11E3F9CE0CA5A26FABDE0CCBDE0CCBDE0CCFEA016090909C60AA60081202D94
S11E3FB704817B2D02862EBDE1D1085A26ED8D0EFEA01608BCA0042698 8D031F
S11E3FD27E3F30CE3FED7EE07E454E544552205448452042595445533A200489
S10A3FED0D0A0000000004AE
S9030000FC

TOTAL ERRORS 00000
```

MK14–the only low-cost keyboard-addressable microprocessor!

The new Science of Cambridge MK14 Standard Microprocessor kit

Just £39.95 (+ £3.20 VAT, and p&p)

The MK14 National Semiconductor Scamp-based Microprocessor Kit gives you the power and performance of a professional keyboard-addressable unit – for less than half the normal price! For less than £44.00 you can have your own microprocessor. One with a specification that makes it perfect for the engineer who needs to keep up to date with digital systems, or for use in school science departments. It's ideal for hobbyists and amateur electronics enthusiasts, too.

But the MK14 isn't just a training aid. It's been designed for practical performance, so you can use it as a working component of, even the heart of, larger electronic systems and equipment.

MK14 Specification

* Hexadecimal keyboard
* 8-digit LED display
* 512 x 8 Prom, containing monitor program and interface instructions
* 256 bytes of RAM
* 4MHz crystal
* 5V Stabiliser
* Single 6V power supply
* Space available for extra RAM and RAM I/O

Free Manual

Every MK14 Microprocessor kit includes a free Operation Manual. It contains operational instructions and examples for training applications, and numerous programs including math routines, timing, general purpose sequencing, games, etc.

Designed for fast, easy assembly

Each 31-piece kit includes everything you need to make a full-scale working microprocessor, from 14 chips, a 4-part keyboard, display interface components, to PCB, switch and fixings.

The MK14 can be assembled by anyone with a fine-tip soldering iron and a few hours' spare time, using the step-by-step illustrated instructions provided.

Tomorrow's technology – today!

"It is not unreasonable to assume that within the next five years...there will be hardly any companies engaged in electronics that are not using microprocessors in one area or another."

Phil Pittman, Wireless World, Nov. 1977

The low-cost computing power of the microprocessor is already being used to replace other forms of digital, analogue, electro-mechanical, even purely mechanical forms of control systems.

The Science of Cambridge MK14 Standard Microprocessor Kit allows you to learn more about this exciting and rapidly advancing area of technology. It allows you to use your own microprocessor in practical applications of your own design. And it allows you to do it at a fraction of the price you'd have to pay elsewhere.

Getting your MK14 Kit is easy. Just fill in the coupon below, and post it to us today, with a cheque or PO made payable to Science of Cambridge. And, of course, it comes to you with a comprehensive guarantee. If for any reason, you're not completely satisfied with your MK14, return it to us within 14 days for a full cash refund.

Science of Cambridge Ltd,
6 Kings Parade,
Cambridge,
Cambs., CB2 1SN.
Telephone: Cambridge (0223) 311488

To: Science of Cambridge Ltd, 6 Kings Parade, Cambridge, Cambs., CB2 1SN.

Please send me an MK14 Standard Microprocessor Kit. I enclose cheque/money order/PO for £43.55 (£39.95 + 8% VAT and 40p p&p).

Name

Address *(please print)*

Allow 21 days for delivery.

Science of Cambridge

HCW

Editorial

Where we stand

The world is full of divisions: rich and poor, north and south, developed and underdeveloped, literate and illiterate. Soon, if we do not make a great effort, there will be another division — between those of us who are aware of the uses and the power of the personal computer, and those who are not. It will be the difference between surviving the information tidal wave and being drowned by it. If we cannot handle the dizzying rate at which the world is changing there will be those who will step forward and handle it for us — at a price which will increase at the same dizzying rate. And we can, all of us, help ourselves. At last science and technology have given us a product which isn't big, noisy, greedy for energy and expensive. It extends personal freedom, personal control, personal knowledge, personal enterprise — without depriving others.

The heart of the small computer is the micro-processor, which delicately adapts itself to carry out the user's commands. These commands may be stored in just a little memory to do a specific task, like tracking the path of the sun for a solar battery: we can have the fine control which means efficiency, the ability to do things that before cost too much. This is hope for the whole world.

Ten to fifteen years ago a computer that had the same power as one of today's small ones would occupy a whole room and cost hundreds of thousands of pounds.

It may turn out to be a miracle of history that just when it seemed that the computer giants had given the titans of government and business the means to reduce us to numbers, the micro-computer for personal use turned up at the scene, increasing our power to turn ourselves from numbers back into people. By that I mean we can no longer be regarded as data for and attendants on the computer but can now use it for our own purposes. I am not saying that there has been a deliberate conspiracy — in fact, there has been legislation recently in the USA, Germany and Britain to guard against the misuse of information stored on computers. But the attitudes are certainly there. The attitudes which make otherwise reasonable people yield to the temptation of piling confusing detail upon redundant detail, all expressed in the language of high priests and idiot savants.

Personal Computer World is part of the 'small is beautiful' movement. We know that there is plenty of goodwill and interest among experts and specialists, and that the presence of articles which are for those of us who want to get acquainted with the microworld together with articles which are more specialised will appeal to those who understand that we must 'hang together, or we shall hang separately'.

Is personal computing only for Americans?

When thinking of computers it is almost a reflex to think of the United States. That's understandable but must be changed because Britain and the Continent are a storehouse of genius and technological sophistication. It's easy to forget that the first computer embodying von Neumann's ideas was built at Cambridge — England. Personal computers are flourishing in the United States — they started there, and the market is growing very fast. **Personal Computer World** takes its inspiration from the many excellent American magazines — all of them young — which serve computerists in the US. We aim to do the same for Britain and the Continent. We admire the energy and imagination shown by personal computerists in the US, and we are certain that we are about to take off on this side of the Atlantic. The signs are here. When a modestly priced computer like the NASCOM 1 sells faster than its creators ever hoped, it means that we are at breakthrough point. And NASCOM 1 is not the only small computer in the market. There are others, and others will follow. **Personal Computer World** will do its part in keeping readers informed; that is why though insisting on objectivity we frankly ask for and acknowledge information from manufacturers. I must say, however, that some people we wrote or talked to seemed to be immersed in torpor. The contents of this first issue reflect in part the quickness of response of others we contacted or who contacted us.

Will we only write about European computers? And deprive our readers of knowledge about computers like the PET (being launched in England in February), the Radio Shack TRS80 (here probably in March), the Cromemco Z2, the Poly 88, the SWPTC 6800, the Imsai, the Altair and the Apple (all already here)? No way. After all, we are all part of the personal computer world, and the Americans are nothing if not world leaders in personal computers.

In the beginning . . .

We're not all of us experts, but we can do certain elementary things to make concrete our interest in personal computing.

Simply, read about computers. Buy a copy or two of *Byte, ROM, Kilobaud, Personal Computing, Dr Dobbs Journal* — all American magazines, all very good. After reading six issues of **Personal Computer World** you will be able to look a computer in the eye. You will also know that what goes on inside it really isn't mysterious, and those words buzzing all around you may be specialised but certainly aren't the mantras leading to nirvana. Also read at least a couple of books. See the reviews in our book page.

Join a club — and if there isn't one around, try to

start one. Use the **Personal Computer World** Open Page to inform others in your area — it's meant for you.

Either build a computer or buy one ready made. Building a computer needs some skill, so first practice putting together simpler kits. Buying a computer is more expensive, but now computer systems which don't cost much more than a good hifi are appearing on the British and Continental markets. In either case, it is necessary for your personal satisfaction to learn some programming — that is, writing a set of instructions for the computer to follow. It is always possible to buy readymade programs; but being able to write your own, however simple, is all part of showing the computer who's boss. Personal power is definitely here.

Come With Us to the CASBA

Why shouldn't we have a Computer Association for Small Business Applications? We at **Personal Computer World** think it is a good idea that small business should have its own voice — its own clout — in the computer world. We would like letters from readers. If sufficient interest is shown, we will put people in touch with each other, and together with them move towards organising the Association. **Personal Computer World** sees itself firstly as a catalyst, and then as a medium for the actvities of CASBA.

Keep the postman busy

Readers of this issue will see that **Personal Computer World** serves the personal computerist — as beginner, hobbyist, recreationer, educationalist and small businessman. We're at the beginning of an exciting period, exploring the vast spaces of a micro-world. We intend to keep people interested and informed, and we shall get better each time **Personal Computer World** goes round.

Write to us and let us know which articles you like best or least. We take our readers seriously, and intend giving more space to your letters than is normally the practice with other magazines.

OUR COVER STORY FOR THE NEXT ISSUE

A nation of PET lovers?

The PET computer is here. We have arranged with Kit Spencer of Commodore Business Machines to feature it in our next issue.

Kit Spencer, who proved to be pleasant and straightforward, says he is a renegade physicist who went into industrial electronics — and now is with Commodore Business Machines. Incidentally, he says that a new division of CBM, Commodore Systems, is being set up to handle PET in the U.K.

Kit Spencer explained the apparent lack of information from CBM during the past year. (See John Coll's article in this issue.) His company, he says, does not believe in creating a consumer demand which it cannot satisfy quickly. Further, it does not believe in selling a computer to a personal user and leaving it at that. It wants to make sure that there is a full service, with good software and maintenance support.

He readily agreed to write an article for us about his company's approach to personal computing. Further — a sure sign of confidence — CBM is to let us have a PET for evaluation.

Perhaps the best established computer store in Britain is the Computer Workshop Ltd, which is at Ifield Rd in London, and sells the remarkable SWPTC 6800. In the next issue we hope to have the story of its principals, Messrs. Ashbee and Burnet: how they took what must have been the difficult decision to go into the then unknown territory of personal computing, and the adventures (no other word for it) they've had so far. Readers can also look forward to a full evaluation of the SWPTC 6800 in a future issue.

Hot on the heels of the PET comes the TANDY TRS80, being launched in March. We're keeping in touch, and the signs are good that Tandy will cooperate with us in presenting its personal computer to our readers.

WATCH OUT FOR OUR NEXT ISSUE, OUT APRIL 18th!

OUR PERSONAL WHO'S WHO

Contributors Page

Publisher **A. Zgorelec** was a journalist for a newspaper in Yugoslavia before coming to Britain. He worked for the BBC's overseas service, and is now a newsagent. As the owner of a ready-made system, his special interest is in BASIC programs for business and games.

Editor **M. N. Solomon** has worked in a library, been in the RAF, been a teacher of maths, knows a few languages (programming and everyday), comes from a large family and would make a great nepotist except that his influence corresponds exactly to his affluence.

John H. Miller-Kirkpatrick is a contributor to **Electronics Today International**, and the principal of Bywood Electronics.

Guy Kewney has single-handedly kept the personal computer scene alive in the pages of **Computing**. Has made some valuable suggestions to us; we hope to follow them up in future issues.

Tim Moore is the creator of Bear Microcomputer Systems, and the Newbear Computing store which is — yes, in Newbury. His manual for the 77-68 microcomputer is a model of clear writing. Tim is noted for his helpfulness and healing touch — people occasionally drag dead systems to his door, hoping for the laying on of hands.

Peter Crofton-Sleigh runs a bee farm in Malmesbury, Wiltshire. He has constructed his own telescope, and a friend is now designing a microprocessor controlled unit to control its movements. When the project is completed it will be featured in our magazine.

Ashok Richter was born in India. He has the knack of putting junk together in such a way that people clamour to buy the results off him. Right now he's teaching himself programming.

K. S. Borland is the managing director of Nasco Sales Ltd, of which Lynx is part. His enthusiasm for his product and the future of personal computing in Europe is unfeigned.

Mike Dennis is one of our consultants. He is an electronics engineer, with some teaching experience. He has built his own Z80 system since getting interested in microprocessors just over a year back. Mike's system: 1K static RAM, 2K executive ROM, intelligent VDU, keyboard together with high-speed serial interface to MPU, and a CUTS interface to cassette. He has plans to build a 4K dynamic RAM board.

PCW consultant **Neil Harrison** first wanted to build his own computer three years ago, and by the time he could afford to build anything the Z80 appeared on the scene — it's now the CPU of his own entirely homebrew system, and has been running for almost a year. Details: Z80 at 2MHz, 4K static RAM, 2K ROM operating system (Zapple), 64 x 16 line TV display plus ASCII keyboard, CUTS cassette interface, papertape reader, digital/-analogue converter.

Michael Whitney works on the Legal Aid Society's ICL computer system. Together with his colleagues he has modified and improved software.

Colin Chatfield works for the holiday branch of Gingerbread, the organisation for one-parent families. He exemplifies what one person with imagination, and co-operation from his employers, can do.

David Jackman has worked in the USA and is now a principal at Casu Electronics. He confesses that it was a lot more difficult to write his article than to design his small business computer. He is also an interface expert.

Victor Nicola and his colleague **Guilbert Percival** work at Data General. Both are engineers.

John M. Anderson is another one of our contributors who disproves the assertion that there is no enterprise left in Britain. He is the principal of J & A Computers.

GIGO is well known to computer fans; just in case you're beginning to fan yourself in, **GIGO** means Garbage In Garbage Out.

SAM? Well, let's say **SAM** stands for Slightly Aimless Missile.

Mike Lord is the editor of the Amateur Computer Club's newsletter. The newsletter is valuable, and so is membership of the club.

PCW consultant **Charles Sweeten** is the Secretary of MUSE, an educational organisation for the use of computers in education. MUSE prides itself on being totally independent.

His friend and colleague **John Coll** is another one of our consultants. He is an electronics expert who teaches at Oundle. Readers will see a regular (approximately) four-page feature which Charles Sweeten and John Coll will put together for our magazine.

Dr Werner Sieber did chemistry at the Swiss Federal Institute of Technology, and was in a research team both in Switzerland and at Imperial College, London, working on the synthesis of zeolite molecular sieves. (*You* look it up, then write us a letter). Dr Sieber just has to be one of the best undiscovered photographers around.

BUZZBUZZ . . . BUZZBUZZ . . .
BUZZBUZZ . . . *BUZZWORDS*

Buzzwords are computer jargon. Like all jargon, they can be mystifying. From the next issue on, Personal Computer World will give simple explanations of those words or abbreviations which readers have written in about. As you can see, reading this magazine is not a passive activity.

PERSONAL POWER IS HERE

J. H. Miller-Kirkpatrick

Personal Computers in 1978

First of all let us try and define the term Personal Computing. Many years ago when I was working on large IBM mainframe computers, the cost of an installation to handle simple accounting and stock routines for a medium sized company was discussed in terms of millions of pounds and parts thereof. The advent of Mini Computers a few years ago brought the installation cost down to the £100,000 area and was thus a feasible proposition for more and more small companies and research establishments. With the advent of LSI technology and finally the first Microprocessor chips the cost of a computing machine has dropped to prices measured in thousands of pounds and promises to drop to hundreds of pounds in 1978.

A computing machine capable of interaction with a human being comprises basically only four units — a human operated input device such as switches or keyboard; an output device such as LED display, TV, printer; a central processing unit and a memory. The cost prices of items in the first two categories are reasonably static due to the use of these devices in many other applications. The cost of the items in the last two categories have dropped considerably during the last couple of years and although prices have not reached rock bottom the curve of price changes has flattened and there should be no major price changes during 1978. However, the prices of components to manufacturers currently promises several simple computing systems at under £500 during 1978 with prices dropping towards £250 by this time next year.

What is a computer?

Basically a computer is a machine capable of doing the same job repetitively and accurately with the ability to react to programmed decisions and thus alter the job that it is doing.

The job that it is doing can be anything from running an On-Line banking system for the big banks to controlling a very simple switching system for central heating or model trains.

I realised a few years ago that my hobby was 'Logic', that seemed to be the only way to integrate my fascination for programming computers with that of playing with electronic circuits. I became involved with the measurement of time, a subject which has produced some very fine pieces of art/workmanship/logic called clocks, simply because I was interested in the complex logic of cogs and wheels. The Microprocessor is the latest extension of this hobby and my main interest in Personal Computing is in the development of generalised computing machines so that one box is capable of doing several different jobs. Other 'Logic' hobbyists such as games players, crossword designers or solvers, statisticians, etc will be interested in Personal Computing from the point of view of using the equipment to further their own particular branch of the hobby. Another area of Personal Computing covers the use of the equipment

simply to make life easier: a complex diary, a personal accounting system for home or small office, mailing lists, etc.

For the designers

For the designer who wants to design his own personal computer with his own software and never-ending hardware, modifications systems start as low as £50 for a simple MPU, switches, lamps and minimum memory. Ideal for electronics hobbyists or engineers as a first step in the use of an MPU, such a system can be used as a development tool in many MPU or non-MPU applications.

The next group of products in general centre at about £250 for a system with a simple keyboard and calculator-style display. At this level your MPU has a little more memory: usually some in RAM for your use and some in PROM or ROM to control the keyboard/display routines. At this level some of the optional extras that make life easier still can be added, examples of these are TTY keyboard/Printers, VDU interfaces, Cassette interfaces and software. As an example the Motorola kit at £200 includes a cassette interface so that programs/data can be stored on tape; the National LCDS at £300 has an optional TTY interface, cassette interface and versions of Assembler and Basic programs in PROM or paper tape form. Beware, at the level of the £200 kit the extras are not optional and you need £500 of TTY before you can do anything. Before you buy make sure what else you need; like buying HiFi, an amplifier is no good without speakers and a tuner or record deck; these non-optional extras can add considerably to the cost of your system.

Of course you can opt to build your system out of parts, like buying a HiFi system from different manufacturers rather than buying a 'Music Centre'. Several suppliers will be supplying on this basis but you will need about £1000 to buy yourself a decent system off-the-shelf. Most of the suppliers are enthusiasts themselves and will not mind offering advice to buyers who have money to spend. Tell your supplier what you want to do and let him advise you which parts you need immediately and which parts you can add later. Most people buying at this level will not be interested in the mechanics of the system but more in what it will do and what software is available. Software engineers will use this type of system for designing simple 'packages' to sell to other users of similar systems, who, in turn, will use the packages for entertainment and/or business.

At the upper level of personal computing your system will cost you up to £5000-£6000 and will include VDU, printer, tapes, disks, etc. At this level most systems sold will be primarily for small business use, not just small manufacturing or service businesses but also the small shopkeeper and one-man business. The system becomes the master filing system with sales ledgers, stock, mailing lists, diary, and multitudinous other applications all being handled by one machine.

Also using this type of equipment is the software consultant who will design and implement your requirements on your system for a small fee. For the full small-company business package this could add another £5000 to the installation price, but if you can modify your manual system to fit into the requirements of a standard software package then the design cost is spread over several users with the same basic requirements and thus could add only a few hundreds of pounds to the cost.

The current situation at the beginning of 1978 is more or less as I have just described. During 1978 we will see a new product come onto the market which is typically a system with output to a TV, interface to a cassette recorder and an optional interface for printers, etc. Prices for these 'music centre' type of systems are likely to be between £250 and £1000 depending on their facilities and country of origin. On systems such as these imported units can include freight, insurance and currency charges as a large percentage of the cost. Some American manufacturers have now set up facilities in Europe and even in the UK. Several UK companies have designed their own equipment which is just coming onto the market at present at prices of about £200 to £500 for a suitable kit of parts. Ready built the units are a little more expensive. By April or May 1978 several such systems will be available but still from specialised shops and mail order companies. Perhaps by Christmas they may be available in most areas of the country.

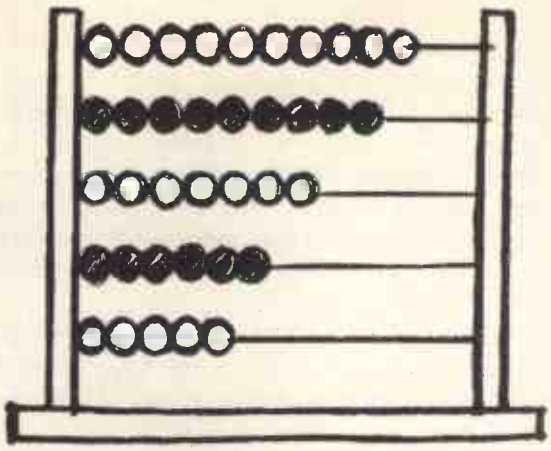
Simple versions of this type of system will be used for training, simple repetitive message handling such as shop window advertising, a design unit for 'one-off specials' or to handle one simple job in the home or office such as a diary or mailing lister. More complex versions will include graphics facilities and the ability to use BASIC or a similar English-based programming language, a full typewriter style keyboard and 80 x 24 character output to a TV. Unlike HiFi the higher cost of this equipment does not relate to its quality but more to its facilities and thus capabilities. If you do not need all of the facilities offered by the more expensive equipment at present then buy one of the lower cost systems. The prices of the higher cost systems will probably drop in time for Christmas '78, by which time you may know which facilities will suit you best. In the meantime, you can exercise your new hobby with the lower-cost devices.

Hobby computing in the home can involve all of the family (if you intend to take over the family TV it *has* to involve all of the family). A home computer for TV games, accounts, diary, recipes, homework, etc. could become more important than the colour TV or even the car in the average household before 1980. Computing machines are going to affect most people's lives over the next few years — it makes sense to make friends with them now: next time the Gas Board's computer sends you a stupid bill you can send back a reply from your own computer!

Welcome to the World of Personal Computing.

Past Procession

GUY KEWNEY



Anybody taking his first look at a computer today would be astonished to hear that it is descended from an adding machine. As Mr Spock might say: 'Illogical, Captain. The device is mainly organised around a code, or alphabet, and its native functions involve storing and moving information. Such calculating ability as it has is unbelievably convoluted, deriving from its crude ability to count — which is clearly a minor (and undesirable) side-effect of the coincidental mathematical basis of binary code inherent in the 8-bit electronic alphabet chosen.'

Anybody who doubts that the computer is basically unsuited to arithmetic has a nasty surprise awaiting him or her when the time comes to sit down and make it work out square roots — or even multiply.

The reason we humans have chosen such a clumsy machine to do our maths for us is that we ourselves are even worse at it. Or more accurately, what we are bad at is precision.

Long before precision became important in trade and bargaining it was important to illiterate races, who evolved the arts of poetry — rhyme, alliteration — and drama or mythology, in an attempt to ensure that what was remembered remained as close as possible to what happened. And when writing was adopted, it was the precision of record that attracted us; likewise the printing press was not so much an attempt to give everybody a novel to read, as to make sure they all read the same novel — or tract, or whatever.

Now the computer is with us, it is replacing the printing press, not the calculator.

Yet it was the calculator that we were after when we invented it. That, and the automaton — we wanted a calculator that knew what to calculate, starting with the automatic loom, automatic flute player, automatic telephone exchange.

The earliest stored program device was, of course, not a computer, but — the music box. This was strictly Read Only Memory — because it was the hardest part of the machine to make, the idea of having two tunes to play on the same set of pipes or strings would be as silly as having two diamonds to wear in a brass ring.

The earliest digital calculator was Babbage's difference engine, and for anybody who wants its full history — and that of the digital computer itself in full detail — I can do no better than recommend Professor Brian Randell's 'The Origins of Digital Computers' (Springer-Verlag) ISBN 3-540- (and 0-387 06169-X) as comprehensive and compulsive reading

Between Babbage and Von Neumann — which is where we are today — there are many refinements, adjustments to new technology, and gradual slips forward.

And some strange things did appear, in the list of tabulators, calculators, totalisers and Strowger telephone exchanges that Randell summarises. My own favourite is a reprint of an IBM 'learned presentation' by John Sheldon and Liston Tatum in 1951 on 'The IBM Card Programmed Electronic Calculator.' What I particularly like is the authors' descriptions of the use of this machine to perform calculations needed for tracking guided missiles.

They say: 'Formerly, thousands of pictures from each of many cameras were turned over to a crew of computers . . . it took two weeks . . . now this is done on the IBM card-programmed electronic calculator in about eight hours.' IBM putting computers out of work — what a way to start!

The Second World War and the study of ballistics provided most of the real incentive to develop the electronic circuitry which was finally available to Von Neumann, the Princeton mathematician who conceived the full stored program concept, finally implemented by the EDVAC group following the successful construction and operation of ENIAC.

Essentially, the computer printed on the silicon of a microprocessor chip is Von Neumann's concept of having an arithmetic/logic unit, talking through input/output to program and data in memory and to

peripheral devices such as printers and card or tape readers.

The history of the computer since Von Neumann has been almost entirely the history of technology. The large and energy consuming (not to say temperamental) thermionic vacuum tube gave way to the smaller semiconductor diode and the more complex but more powerful transistor switch. A very much bigger computer could be built for the same money, and in the same space.

Then it was noticed that the transistor was actually a very small area of physical/chemical solid state interaction on a piece of germanium, or silicon. Two or three transistors could be built on the same tiny piece of semiconductor material, and connected to each other, for the same cost that one transistor could be built.

The semiconductor makers went out and found that their customers were buying innumerable transistors and putting them together in the same standard way, as gates — electronic circuits which produced a 'one' or 'zero' (actually, I prefer Von Neumann's concept of 'everything or nothing') according to the combination of ones or zeroes fed in to them.

And because the cost of a computer dropped heavily, the semiconductor makers sold considerably more gates than they had sold transistors, because so many more computers were sold.

The process continued as the ability to get more transistors on a single component improved. Shift registers could be produced as single components — so they were. Dividers, adders, the list went on. Unbelievably complex components appeared, until the semiconductor makers realised that the very complexity of the components they were making was starting to limit their application. What was wanted was the 'universal component' — something which would do whatever you wanted, and then change itself to do whatever you wanted next.

At this point, a company called Datapoint approached Intel, and asked for a design of processor/terminal controller circuit which would, in effect, be a computer. Intel rather over-optimistically said the job could be done and produced the 8008 — which was no good for the task, it could not be reliably built, and it was not powerful enough for a disappointed Datapoint. Intel scaled it down to the 4004, which played innumberable games of TV ping-poing, and up to the 8080 when they got smart enough.

The 8080 and its wretched instruction set were upgraded by Motorola into the 6800 and by Zilog into the Z80, and the 6800 was upgraded by MOS Technology into the 6502. These are the machines we amateurs have to live with today. Others will come before Von Neumann's child outlives its potential.

Yet that potential is nearly ended. It was extended by the ignorance of semiconductor designers who decided to imitate computer memory with transistors, rather than looking at the whole field of computer architecture and saying 'what have we here?'

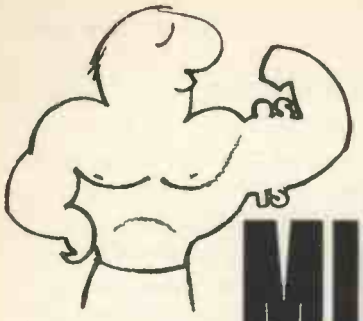
What we have here is a computer which, when asked for the one faulty item in 10,000, has to check all 10,000 to see which it is. How much better to have a computer which merely says to its component parts: 'Any faulty items report here at once'.

The ability to give memory that processing power has existed for something like five years already. *Associative* processing has been largely ignored because the markets for supplying cheaper forms of existing Von Neumann components were easier to satisfy — but the limitations of a sequential processor have nearly been reached.

But not by us, the private users of computers. We are only on the edge of a revolution which will make the printing press, the telephone and the motor car look like minor items on a shopping list, as the population gets 'on-line'. And from here on, the history of the computer will be the history of society, not just of calculators.

COMPUTER CLOCK: *Approximately 1500 BC, the abacus *1643 — Pascal invents mechanical calculator for his father, who was (naturally) a tax accountant. *1822-1834 — Babbage designs but never completes his analytical engine, introducing for the first time the ideas of storing data, controlling the computer through a specific program of instructions, and printing the results. Babbage was assisted and supported by Lady Lovelace. *Dr Herman Hollerith develops a punched card system for the US Census Bureau, in time to deal with the 1890 census. *1937 — Harvard professor Howard Aiken begins development of his Mark 1 electromechanical computer. Funded by IBM, it was completed in 1944. * 1939-46 — Dr John Mauchly and J. Prosper Eckart create the ENIAC at the Moore School of the U. of Pennsylvania. Used by the US Army. ENIAC stands for Electronic Numerical Integrator and Calculator. *1945 — Von Neumann for the first time suggests the use of binary patterns to incorporate data and instructions, as well as the internal storage of these in the computer. *1946-52 — Von Neumann and Goldstine develop the EDVAC (Electronic Discrete Variable Automatic Computer). But beaten in the race by a group at Cambridge who in 1949 brought out the EDSAC (the DS stands for Delay Storage). *1951 — Eckert and Mauchly sell the UNIVAC — 1 to the US Census Bureau (UNIVAC — Universal Automatic Computer). *1953 — IBM enters the field with the 701, the beginning of the era of IBM dominance. But UNIVAC hangs on. *1958 — computers begin using transistor circuits. Control Data Corporation begins competing. *1963 — the 360 series of IBM computers strengthen IBM's grip on the market. *The first of CDC's superfast computers, the 6600, shows that IBM doesn't have a monopoly on size and speed. * 1964 — First microcomputer, the PDP 6, by Digital Equipment Corp. * 1970 — CDC 7600 and IBM 360/195.

Also, Data General's Nova, a sixteen bit mini. Intel decides to market microprocessor designed for but not used by Datapoint as the Intel 8008. This might help to explain why the 8008 hasn't got such a brilliant instruction set — it wasn't specifically designed for personal computing. * January 1975 — The start of personal computing. *Popular Electronics,* a US magazine, publishes the design of the Altair 8800, based on the Intel 8080 Central Processing Unit. The rest is present history.



A MIGHTY MICROMITE

Tim Moore

The 77-68, a simple start to a large system

Introduction

Many people in this country have a considerable interest in, and a great desire to learn about and experiment with microprocessors. Unlike in the USA there is comparatively little spare spending money available for such luxuries, so there was a need for a low cost microprocessor system which could be built up cheaply and gradually into a complete and worthwhile system. The vast variety of microprocessors available cause a great deal of confusion to the newcomer to the field, but a few critical questions can rapidly clear this situation:—

1) Will it still be manufactured in five years' time?
2) Can it easily support a 'Basic' interpreter?
3) Is Software (Assemblers, Basic interpreters etc) readily available in the UK?
4) Is the price affordable?

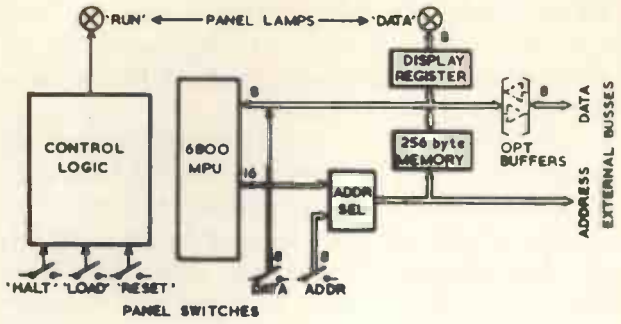
This immediately narrows the field down to the Z-80/8080 and the 6800. Of the two families the 6800 is simpler to begin with and better supported with software in the UK. The Z-80 is a very powerful but also very sophisticated and complex microprocessor, more suited to the experienced than the beginner.

The Simplest System

To make a 6800 microprocessor work, various additional items are needed eg:

1) Memory to store the program (which is to be executed).
2) A method of loading the memory and inputting data.
3) A method of outputting data.
4) All the necessary timing signals for the 6800.

These requirements can be satisfied with 256 bytes of memory, toggle switches, LED's, and a handful of TTL logic.



THE SIMPLEST SYSTEM

Operating Instructions:

Memory Change

Loading the memory with a program is accomplished by setting the 'Halt' switch which effectively turns off the 6800 and isolates it from the data and address bus. The 6800 achieves this by being able to make its data bus interface go high impedance. Also setting the halt switch causes X8 and X9 to select the address switches rather than the 6800 as the address inputs to the memory IC's X17 and X18.

Pushing the 'load' switch then causes whatever information that has been set into the data switches to be loaded into memory.

The contents of the memory can be inspected by setting up the address switches and observing the LED's.

Input/Output

The microprocessor, when it is executing a program, first of all selects a memory location and then either reads that byte of program or writes information to it. If location 'FF' hexadecimal (1111 1111) is selected and information read, the data switches are selected and whatever information they contain loaded into the 6800. Similarly if the 6800 writes into location 'FF' then the information is displayed via the LED's.

Running a Program

A 'Reset' causes the 6800 to jump to memory location 'FF' and there find the starting address of the program it is about to execute. If it is in the 'Halt' state it will then wait to be released with its starting address safely held in one of its internal registers.

Thus if the following program is loaded.

Memory Location

00	7C	INC
01	FF	
02	FF	
03	20	BRA
04	FB	

and the data switches set to zero, the 'reset' button

pushed and the halt switch set to run, then the instruction 7C is acted upon by the 6800 which asks it to go to memory location (FF) FF (only FF is acted upon because of the limited amount of memory present), reads it, (eg the switches) and then after incrementing it by one, writes it back into the same location (eg the LED's). So whatever the data switches are set to will, in a slightly modified form, appear on the LED's. The next instruction causes the 6800 to branch back to the starting address and repeat the operation in an endless loop.

Improvements on the basic system

Toggle switches and LED's are fine for demonstrating the binary nature of a microcomputer but a little tedious when 256 bytes of program have to be entered. The first improvement is to remove the LED's and replace them by two seven-segment LED displays and thus read Hexadecimal code directly. The second improvement is to build a hexadecimal keyboard that automatically advances the address switches and depresses the 'load' switch.

Constructional details

The printed circuit board is 8" by 8" and has a 78-way gold plated 0.1" pitch edge connector. This size was chosen because it is supported by several UK manufacturers. This means that card frames, prototyping boards and edge connectors are readily available at a reasonable price. As there are only 24 IC's it is feasible to hand-wire this circuit together, if a suitable prototyping board is already available.

If a reasonably deep junk box is available to the constructor, the cost of this machine could be kept to under £50.00.

Back-up Support

Microprocessors are complex devices and do take quite a lot of understanding before anything can be done with them. To help with this problem a manual was written which describes 77-68 in far greater depth, the 6800 instruction set, and gives a range of software which can be used. A user group has been formed which is supported by a quarterly newsletter. This was considered vital because it enables users who live locally to help each other commission their machines and chase out any faults that may have been introduced during construction. In practice this has worked very well and the user group now holds all sorts of software and hardware extension information for distribution among its members.

The Expanded System

To be able to run a 'Basic' interpreter, it is necessary to attach a visual display unit or teletype-writer to the computer and to have at least 4 kilobytes of memory.

This is achieved by regarding the main board as simply the central processor board of a large system. All the major control lines of the 6800 have been brought out to the edge connector and both the data and address busses have been fully buffered. A 4 kilobyte RAM printed circuit board and a 'Soft' monitor board which interfaces the system to a VDU etc. Design information already exists for interfacing the 'Mikbug' monitoring ROM with a single step feature and various other additions are expected such as a single card VDU and a Dynamic RAM board.

The method of connecting these boards together is simplicity itself: attach the appropriate number of edge connectors to the card frame and then lay a piece of vero board across all the edge connector pins, connecting all the pin 1's together, and so on. This enables a very cheap, fast and reliable back plane to be constructed. This will be appreciated by anybody who has ever wired up a large backplane — it takes a long time.

Conclusions

77-68 is a 'live' project, every month brings news of another extension to this system. All the parts are readily obtainable and if any problems are encountered there is a choice of a hundred or so other users who can give impartial advice and help.

Summary

* A 6800 based system for home construction.
* Low cost approx. £50.00 with help from a junk box.
* Can be fully expanded to a 'mainframe' microcomputer.
* Fully supported with a user group.
* Can be used with a simple switch interface or a Visual Display Unit.

(**PCW** We hope to feature another Mighty Micromite in the next issue of Personal Computer World. **PCW**)

77-68 Components List

Logic

3 off	7400	X4, X14, 24	1 off	7495	X2
1 off	7402	X19	2 off	74125	X22, X23
1 off	7404	X1	2 off	74157	X8, X9
1 off	7410	X3	*2 off	81LS97	X15, X16
2 off	7428	X5, X6	2 off	2112	X17, X18
1 off	7430	X7	1 off	6800	X13
2 off	7432	X11, X12			
1 off	7474	X10			
2 off	74LS75	X20,X21			

* optional buffers

Resistors

20 off	4.7k	¼w
*2 off	1k	¼w
14 off	430	¼w
1 off	270	¼w
2 off	220	¼w
2 off	22	¼w

Capacitors

3 off	33uF	6.3V tant
8 off	0.1uF	ceramic
1 off	33pF	

LED's 9 off

Crystal 5 MHz

Switches

17 subminiature toggle switches
single pole 2 way S3-19
1 push single pole n.o. S1
1 push single pole c.o. S2

*option

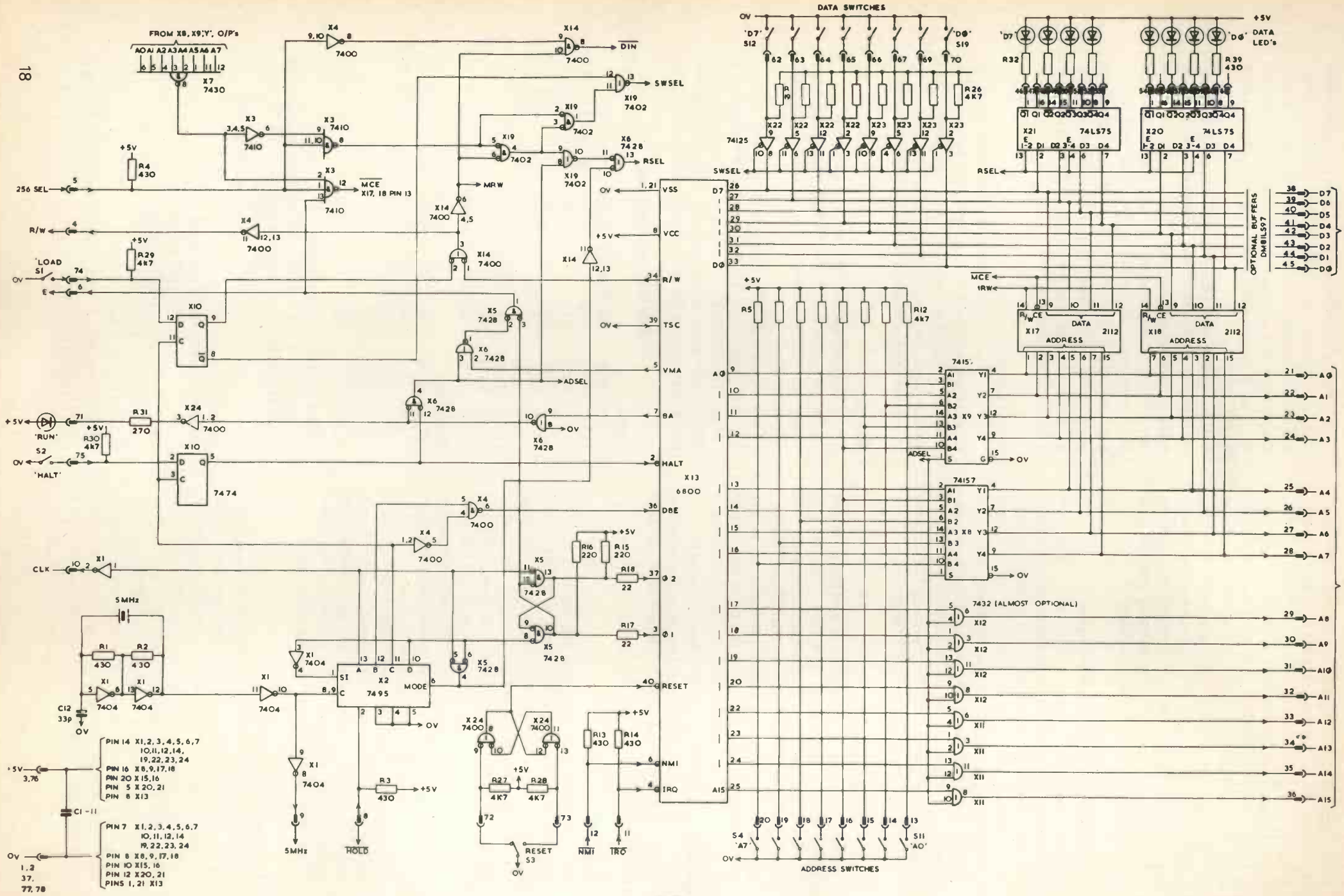
Low profile sockets 2 off 16 pin
1 off 40 pin

3 feet 22 s.w.g. wire and sleeving

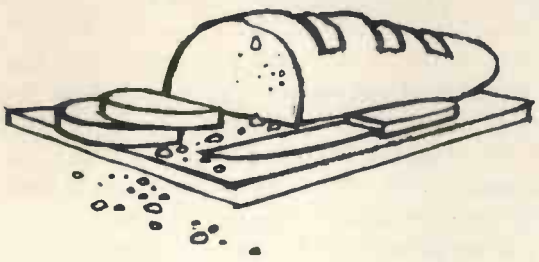
1 foot 8 way ribbon cable

Printed Circuit Board

Edge Connector, 77 way + polarising key 0.1" single sided.



How Not To Crumb Up Your Breadboard



A. T. Richter

This introduction is for those readers who do not have any previous practical experience in electronics. It will help them to follow the assembly instructions sent along with any kit. It lists some tools (with an emphasis on the use of a soldering-iron), and also gives some hints about available constructional aids for more advanced projects, and how to achieve the combination of maximum flexibility and compact design.

Apart from the kit which consists of loose components, there are a small number of tools as well as test equipment for the later stages of the assembly which will have to be purchased and can serve as a basis for a growing workshop.

Sidecutters (110 mm), pointed pliers (115 mm), a set of insulated screwdrivers (3.5x100, 4x100, 5.5x150, 6.5x200 mm), wire stripper, a good soldering-iron (20-30W, preferably with two or three exchangeable solder tips of different diameters), and solder. But ordinary solder can be damaging — electronic 5-core solder is better. The use of tweezers as a soldering help is not essential, but with a little practice results in better soldering and prevents the fingers from getting burned by hot wires, pins, solder etc.

A workbench, good lighting and a few handy mains plugs cut down accidents. The source of hardware errors can be traced and found when employing a logic probe that indicates the state (1 or 0) of any particular gate. Other test equipment such as: a multimeter (digital), double power supply (if using TTL circuits) with short-circuit protection and automatic cut-out, a function generator, and an oscilloscope decrease the time spent in testing and de-bugging — this reduces costs and opens immediate possibilities for new projects and the development of new ideas.

Experience shows that many enthusiasts go through all the trouble of building, testing, de-bugging, and get the machine to work perfectly or even add the latest improvements, but leave the circuits mounted on a little breadboard or loosely spread over the kitchen table, where it is fair game for the open air, dust, humidity and domestic animals. All this is impracticable and should be avoided. The electronics industry provides us with a great variety of mechanical parts which make it possible to build professional units with a minimum of tools, time and money.

I should like to consider the way the soldering-iron should be used, since badly soldered joints are the main source of invisible errors. These sometimes appear long after the entire kit has been assembled and tested. Non-conducting soldered joints, bad connections, bits of solder producing short circuits, and wires coming off make one wonder whether the components are faulty or not.

If the surface of a soldered joint is not smooth and shiny it needs resoldering! Preheat the pin and the surrounding copper for about three seconds, then add just enough solder for one soldered joint and remove the soldering-iron after another three seconds. The solder should now flow back without sticking on, then harden properly.

For easy maintenance, checking and testing while still in the assembly stage, it is of importance to be able to exchange or disconnect the logic boards within seconds without having to unsolder the whole lot. I therefore suggest the well-known 0.1 Veroboard or the re-usable Blob Board, together with edge connectors which can be soldered onto another board, saving you "Christmas-tree-wiring". The same applies to integrated circuits: once they are soldered direct onto the board it is a destructive job trying to get them out again, whereas the use of low profile sockets reduces the risk and effort considerably. There are also many types of ribbon cable and connectors which increase flexibility, and simplify the wiring. Also available is mounting hardware such as: PC board guides, spacers, and stand offs which save space for larger circuitry can be installed compactly into any suitable prepunched cabinet (from Amatek).

Integrated circuits are highly sensitive devices which can be destroyed simply through touch! Clock ICs, calculator chips, MPUs, have a very small current consumption of less that 10^{-10} W per gate. For that reason ICs are delivered on conducting foam, polystyrol or aluminium foil, shortcircuiting any static electricity. They should not be separated from their protection until the kit is completely assembled.

YOURS TO COMMAND

K. S. Borland

Britain's Own Microcomputer — The NASCOM I

The electronic hobbyist in the UK has been left out in the cold. His spending power is generally far less than that of his American counterpart. The fall-out of products or peripherals from our highly specialised electronics industry is minimal.

It was in this climate that Lynx Electronics decided to go into the micro business. Lynx Electronics is the hobbyist subsidiary of North American Semiconductor, known in the UK as NASCO. John A. Marshall, Chairman of the North American Semiconductor group, is frequently in the USA; particularly at the NAS American headquarters in California. The west coast is a major area of electronics development and has a well organised hobbyist industry. With members of the NAS operation he was able to visit amateur clubs and see at first hand the standard to which they have risen.

The American hobbyist scene has developed a long way. New technology, now predominantly MPU — in fact, whole systems — are available; and with these there is a wealth of peripherals. The clubs are not only well organised and well attended but have become forums where the small and medium, and even the large American electronics manufacturers, feel the need to be represented. Most colleges and universities that have an electronics department run a computer club. As nearly everyone who deals with either hardware or software is a member of a club it is very important to the American electronics industry that products are made readily available and that new developments are rapidly broadcast. What do we have in the UK? We would suggest, virtually nothing. This is possibly unfair to those who are trying very hard to start, such as the Amateur Computer Club. But there are very few facilities available, and as the British electronic industry does not seem to have heard of amateurs, it must be hard going.

It was against this background that the NASCOM I evolved. The project has been to produce an advanced technology kit and put it within the U.K. mass purchasing market. The design had four major concepts.

We set out to:

Firstly, produce a complete microprocessor system that is of intelligent use to the home user and in basic price around £200.

Secondly, use the best available mixture of products on the market, within our price range, and with it to produce the maximum possible system. There is, of course, an advantage here that an independent design has over a manufacturer limited by his own product range.

Thirdly, that whatever product was used the object must be to obtain the greatest possible control by software.

Fourthly, to design a system that would offer the user major future expansion. The design must be standard enough to offer a competent engineer the opportunity of adding his own expansion. Also Lynx must be prepared to offer an expanding range so that the software enthusiast can expand his system.

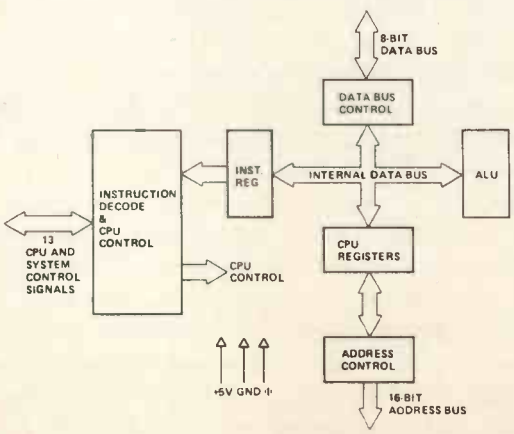
If it was possible to produce a microcomputer like this and the British amateur was ready to buy, then it might be possible to start a national NASCOM club. As well as being ideal for the amateur it would solve many educational problems by creating a low cost microcomputer. This could open up computer clubs in colleges all over Britain and the Continent.

This project started in the summer of '77 and NASCO commissioned Shelton Instruments Ltd. in London to design the NASCOM I.

Shelton Instruments specialise in the design of control equipment using microprocessors. They are capable of design around most available processors. A feature of the NASCOM I design is the trade-off between the hardware and the software.

To the great relief of us all the project is a complete success and the NASCOM I became reality. The design team had a free choice of microprocessors and decided on the Mostek Z80. Apart from the help that was freely given by Mostek, the main reasons for choosing the Z80 were:-

1. It is a third generation, 8 bit, LSI, CPU chip.
2. It is designed to require the minimum of hardware interface.
3. It has an easy to learn instruction set. This instruction set however is powerful enough to give flexible and efficient memory usage.
4. It is rapidly becoming a popular device both in the professional market and for the home computing enthusiast.
5. There is a fair amount of software already written for the Z80 for home applictions.
6. The Z80 will execute all 8080 software.
7. There are multiple sources for the Z80 family.
8. It has a simple power supply requirement.



Z-80 CPU BLOCK DIAGRAM

At the centre of all microprocessor systems is a Central Processing Unit (CPU). This manipulates data as directed by the instructions stored in the memory (program). It can be thought of as containing three sections:-

1. An arithmetic and logic unit (ALU) which processes data, eg adds two numbers together, combines two numbers on the basis of a logical AND and so on.
2. A number of registers which are simply single word stores, some of which operate as pairs giving 16 bit capability.
3. Control and timing circuits to synchronise the various minute steps that need to be taken to implement the program.

Information is transferred to and from the computer system via the input/output circuits.

The CPU processes data as parallel rows of bits and in many cases the input/output data is transferred in that form. For an 8 bit system the input/output bus consists simply of eight parallel wires.

All systems use some form of clock as a basic timing reference for instruction executions, memory and input/output operations.

In order to form a system the CPU must be linked to the other component parts. In general the connections can be divided into three groups. The data bus, the address bus and a number of control lines. The address bus is used to select an address in memory or to select some other external location. The number of address bits is not directly related to the word length used *in* the processor. The actual number of bits does determine the number of different locations that can be addressed. Eight bits give 256 different addresses whereas 16 bits give 65536 addresses. The function of the control lines include data strobes, an address strobe, reset, interrupt and flag lines.

A microprocessor system is illustrated in Fig 2. The control system is present in the ROM (read only memory) and the program and data are stored in the RAM (random access memory).

And so to the NASCOM I. In simple terms it is a computer kit, created by a hobbyist company, for use by hobbyists. Instead of selling the home user new equipment the design started with what *was available*: a television and a cassette deck. Any computer system needs a VDU so the basic NASCOM I has an interface to a domestic TV through the aerial socket. Similarly an ordinary audio cassette is used for storing and loading programs and data on standard magnetic cassette tape.

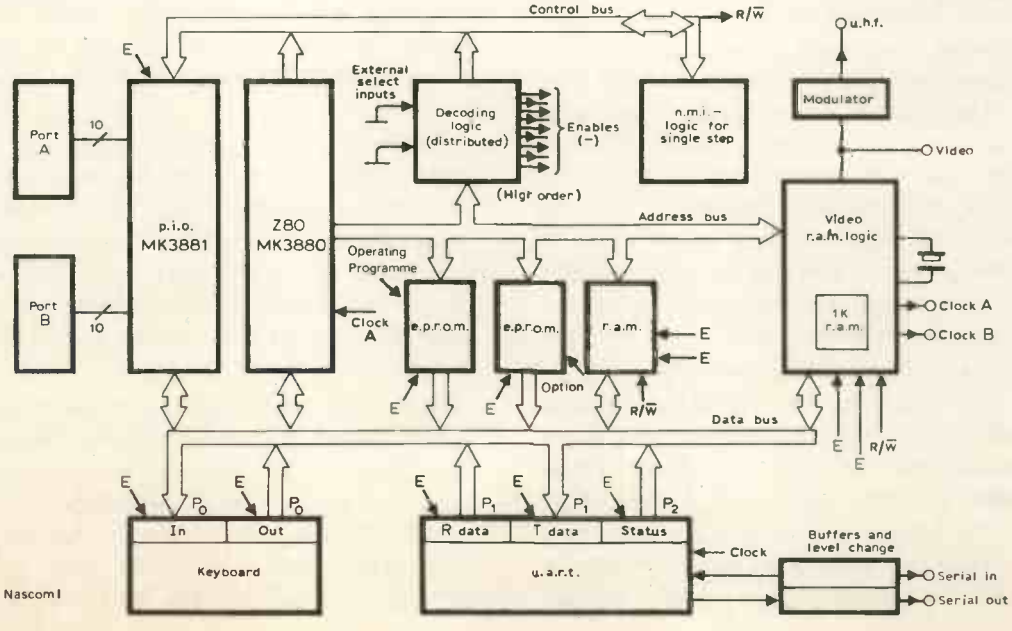
The Z80 CPU is connected to the memory and the input/output areas through a three-bus system.

There is a 1K x 8 EPROM which has been pre-programmed with a monitor program. It comes into operation immediately power is supplied to the system. This allows the user to utilise his kit when he has assembled it.

It has been obvious from the design stage, when considering expansion, that even a hex keyboard (0-9, A-F) would not suffice. So a Full 'QWERTY' keyboard is supplied. Data and programs are entered via the keyboard and the monitor program will interpret and execute the required operations. The user may enter his program into a RAM memory for future use under control of the monitor program.

The monitor has these main functions: Enter information into the memory. Tabulate the contents of the addressed memory on the television screen. Store memory on cassette tape. Load memory from cassette tape. Start the program from any given memory address. Stop program at a pre-determined point. When the program is stopped at a breakpoint the monitor program automatically copies the internal CPU registers into the RAM. They can *then* be examined by displaying those particular RAM addresses on the television. When a user program is started the internal registers of the CPU are loaded from this area. This allows starting data to be pre-set by the programmes.

The basic NASCOM I has a 2K x 8 static RAM. All of this is available to the user. However, to run the television as a VDU it is necessary to use 1K as a character store. Bit serial data is passed to and from



the cassette recorder via a serial interface. The keyboard is controlled by the monitor program. There are few I/O circuits in this system and therefore there is no need to have an address decoder. The individual port addresses are such that address bus bits can select the right peripheral.

To keep the memory address decoding simple the memory is arranged in two rows of 1K x 8. This means that to address 1024 address locations needs 10 address bits which are common to both rows. To select the correct row the EPROM needs three more lines. These are obtained through suitable decoding circuits.

The basic NASCOM I is in HEXA DECIMAL with the monitor program accepting the HEX codes. However, Lynx have been lucky enough to associate themselves with a software house, namely Starbase. Many of the software orientated hobbyists are already using some form of high level language. This posed the problem of what software to produce first. It was already decided to produce a BASIC tape, but many people want an assembler. However, the decision has been to produce a TINY BASIC for a 4K RAM. The first one hundred will be supplied free to those who have bought an expansion board or enough RAM. They will be asked to report on the acceptability and then it will be committed to ROM. After the BASIC is in operation an assembler will be investigated. Various hobbyists have already intimated that they will be writing an assembler so it may be available fairly quickly through the NASCOM Club.

So what can the NASCOM I do for the personal user? What software back-up will be available? The questions are so open ended. For example, there will be about 45% of the hobbyists who are competent engineers, and another 45% will be programmers. They will have different problems. Those who are horrified at assembling the kit will be offered the NASCOM I made up. There will be an assembly surcharge and kits will be made against orders. For those who may have software problems, the solution is less obvious. As the project progresses the library of tapes will be ever-expanding, and as the computer is for the hobbyist there is no doubt that the range will be enormous. But the hobbyist will not necessarily want written programs but help with writing his own. Lynx do not propose to teach programming, but help could come from two quarters. Firstly, and officially, Starbase is there to help. That is their occupation. They will be happy to receive problem programs and suggest solutions. For this there will be a charge. The other possibility is the NASCOM Club, of which more later.

However, these problems themselves generate an interest in the hobbyist. As a starter, many engineers will be learning programming and many programmers will start dabbling in hardware. Many are competent in both fields, but a very large proportion of the letters received by Lynx after their Wembley seminar were from buyers or potential buyers whose immediate excitement was to expand into a new medium: one which is a unique combination of entertainment and usefulness.

There are so many ideas and areas to explore. Control in the home. Heating, cooking, security. Games, both adult and junior, come in many shapes and forms. The games industry, specially in America, is getting ever more ambitious. From simpler games like Life and Hangman, to games of Startrek and Space War. The even more ambitious will attempt Chess. Of course, the more complex the games the larger the RAM necessary. So Lynx will be looking forward to many complex game players.

Not only are the games fun to play, they also give practical experience in writing programs.

How about writing an electronic diary? A home computer could remind you, day-to-day. It could look after your finances. Remind you of payment days.

There are great possibilities as a teaching aid. The NASCOM I is capable of graphics. The only graphics produced so far are very simple, but with the time the hobbyist has, many complex graphics will soon appear.

To aid with amateur radio. Photography offers many ideas, such as colour analysers and filter choice.

The initial NASCOM I has its command EPROM, and 2K of RAM. Of this 2K there is 1K devoted to the TV display. This leaves about 1K free. As a starter this is quite sufficient. Writing programs in HEX will limit the complexity but allow the hobbyist to get into the Z80 user language. From then on user needs will be guided by applications. Lynx has foreseen the need for expansion and have already produced their starter accessories.

As the only possible expansion on the main board is a 1K EPROM, the first need is for an expansion board. There is no ideal board. Many specific boards generate problems of supply and this increases the cost. However, there must be choice. So the expansion board has been designed for choice.

Even on a multi-option board there is argument as to its options. Lynx has designed its expansion board for three main options. The board itself will be through-plated and will include a decoder, an address multiplex logic and the CAS and RAS strobes.

The first and most obvious option is to increase the RAM available to the user. So there is an option for 4K or 8K of dynamic RAM. This should be enough immediate expansion for most users. However, some may want to expand very fast into much higher capacity. Although RAM boards are not immediately available they are planned for early 1978 and will probably offer 16K per board.

As the user advances to the stage of needing more options, many will have specialised their use into one main area. The monitor program will be in continual use and therefore it seems reasonable to commit it to a ROM. As a second option on the first expansion board Lynx have decided to offer up to 4K of EPROM. These are sold in 1K blocks and together

with the 1K EPROM option on the main board offers 5K. This should suffice to accept more advanced programs.

The third option is the most difficult. What is going to be linked to the NASCOM I? The system already offers a spare parallel I/O on the main board. But this will not suffice for very long so Lynx have added the option of another PIO kit to the expansion board.

Having reached this stage let us look at cost. As has already been intimated the NASCOM I micro-computer costs £197.50. Assuming that the user has a TV set then all that is needed to start is a power supply. Many hobbyists will have a power supply available but if this is committed then Lynx are able to supply one at a cost of £24.50. The other option is extra sockets for the TTL. The main components are socketed but many users will prefer to socket all the logic. This costs £4.90. This means that even with extras the hobbyist is offered a Z80 based micro-computer for the grand total of £226.90 (plus VAT). This is what Lynx set out to achieve. Although there are many methods of operation, choice of modem, format of display etc that will no doubt be discussed, and some criticism will be right, Lynx would ask everyone to consider if it would cost even fractionally more. There are obviously more possibilities available at £250.00 and even more at £300.

So we have a kit working out at £227. Let us now consider expansion. The obvious expansion is in RAM. The maximum at present is 8K of dynamic RAM. This will cost £97. So we now have a system with 9K of RAM at a cost of £324. Together with all the power and control already offered in the NASCOM I, with 9K of RAM it is unbeatable.

With our 8K extra RAM it is likely that the program to control 8K of data is past the simple stage. The maximum EPROM available is 5K at a cost of £87.50. The system by now probably needs various extra peripherals. So let's add a PIO kit at £11.50.

We now have an expanded system which includes a NASCOM I, 8K of dynamic RAM, 5K of EPROM and two PIO's in addition to the TV and cassette player. At a total cost of £423 for a very powerful computer system.

Lynx will have the capability of loading the PROMS as the users complete their programs. However, it is the aim of every hobbyist to be self sufficient. So Lynx offer an EPROM Programmer. This will program 2708 type EPROMS. The programmer will plug into the NASCOM I bus. All the timing, address generation and data handling is performed by the NASCOM I using a special program. The program voltage is generated on the card from a +12v supply. It uses TTL parts and the price includes a special zero insertion force socket. In kit form Lynx have managed to offer this at £55.

To go with the programmer Lynx offer a kit to make up an EPROM Eraser at £32.50. The whole project already offers more for less and it has only just started.

We have discussed briefly the climate in which Lynx decided to produce the NASCOM I. What has been produced was based around a price and the chance of expansion. The NASCOM I will have an immediate effect in that, at the time of writing this article, Lynx have sold 400 kits in the two weeks since the launch. The media have shown great interest and, although Lynx would like universal approval, there will no doubt be varying comment once they have a chance to use a NASCOM I.

Comment and information exchange is the other side of this project. It may be naive, impracticable or even arrogant to attempt to start another computer club with no more facilities to offer and certainly less expertise than those already in existence. But we intend to try.

This is not started without thought. There do seem to be certain advantages in this situation. Starting a club does of course revolve around the success of the NASCOM I. But after the seminar where 500 people gave it the thumbs up, and as Lynx has received more than two thousand enquiries, it would seem to be well on its way. Further, Lynx has had hundreds of enquiries from educational establishments from sixth form colleges to universities. This area of sales has not even started yet although twenty or so colleges have ordered one or more computers.

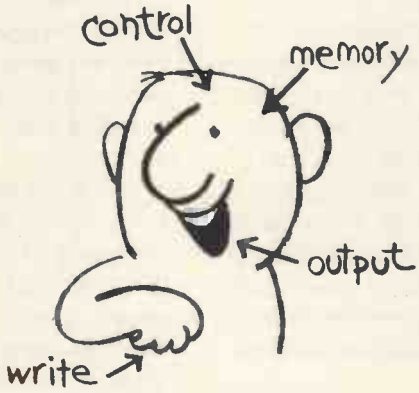
The NASCOM I is the ideal teaching machine as it incorporates the Z80. But this is not the main interest as far as a club is concerned. The sale of the NASCOM I to the general public will, of course be national, and so will the sale to colleges. It would seem reasonable to suggest that most colleges would start clubs especially if a fair proportion of the members had their own compatible hardware. There are certainly clubs in some colleges already. Southampton and High Wycombe spring to mind. But as far as we know there is no connection between them.

This is where we see the possibility of the NASCOM club. Each member will have access to the other members and, when and if they get started, each local club would have knowledge of the others throughout the country. Nothing new in any of this, but is is not being done at present, and in a country like the UK it is not only possible but quite practical.

For all the enthusiasm that Lynx people have, this still is entirely in the hands of the UK hobbyist. By March 1978 hobbyists will have kits spread throughout the UK and could by the end of 1978 number in thousands.

THE LITTLE SYMPOSIUM THAT GREW

LYNX HOME MICROCOMPUTER SYMPOSIUM



The Happy Symposiumite

Mike Dennis

The future of personal computing may well look back on Saturday, 26th November 1977 as the day on which home computing truly got off the ground in the UK. This was the day that Lynx Electronics held their Home Microcomputer Symposium in the plush surroundings of the Wembley Conference Centre. Past seminars have tended to be aimed at the professional and often with a price tag to match; but here was a brave new venture deliberately aimed at the home user and with an entrance price of only £3.50. Six speakers were expected, and with the launch of their NASCOM I, it promised to be an interesting day. So if you were unlucky enough not to be there, here are some of the impressions that I came away with.

It started with a very sketchy 'Introduction to Home Computing'. I must admit that I found "cost equivalent die size" and "random logic gate area" all rather irrelevant at this point. So did, I suspect, the majority of the audience if the number of heads bent studying the remaining lecture notes was anything to go by! I preferred the second lecture, called "Introduction to Hardware", with its section on 'What is a microprocessor?'. On the whole this was very eloquently presented by J. Ayres; however, the pace was alarming! Diagrams flashed up before our eyes — often too dim and more frequently out of focus. We belted through timing waveforms and tri-state buffers to arrive at a 'practical system' which — surprise, surprise — closely resembled NASCOM I.

The third speaker was Phil Pitman who introduced us to the software of the Z80. This could almost be a day's lecture in itself, and it is to Phil's credit that he attempted to treat such a big subject in the time available. I feel, though, that the order of presentation could have been a little tighter, for at times we jumped about like a berserk stack pointer. For instance, did the stack pointer really overwrite new data on top of that just written? No, of course not, but I think that a newcomer might easily have been confused.

In Lynx's defence it should be said that it was only four months since the inception of NASCOM I. In that short time, Lynx have had it designed by outside consultants, started the manufacturing and organised this conference. So if perhaps events on Saturday were a little ragged round the edges one can understand why. If Lynx could be accused of anything, it would be on a charge of underestimating the phenomenal interest that such a project would generate! For example, despite a very low advertising profile, nearly six hundred people turned up when only four hundred had been catered for.

Be that as it may, we had now arrived at the demonstration of NASCOM I, and it was here that the lack of time was most acutely felt. I fear that Lynx may have lost a few potential customers as a result. What we were presented with were two prototypes surrounded by the usual 'birds nest' of wires, instead of a fully fledged production model. Another disappointment was the inability to display either of the prototypes onto the TV monitor screens dangling above our heads. This was not due to any fault in the NASCOMs but due to the fact that

whereas NASCOM talks to domestic receivers by UHF, these monitors would have preferred a line feed, I suspect. What we did see on the screens on the stage were a few desultory runs of the Game of Life before we came to the draw for the winning ticket.

This draw was included in the ticket price and the prize was a brand-new NASCOM I kit. It was a masterstroke to use a NASCOM to choose the number of the winning ticket, but it was unfortunate that the programme as entered would only choose numbers without a matching ticket! A quick play with the keyboard and some modifications to the programme cleared that 'bug' but what were the numbers? We never really found out, much as we would have liked to, for hovering in the wings was the spectre of the Centre chef with his steaming ladle to say that lunch was ready! Still, the magic moment was upon us and the Go button was pressed. Utter confusion, as the answer was still in hexadecimal and many frantic calculations were made. To no avail, as the winner wasn't there. The eternal cry of *redraw* finally brought the morning session to an end, and this time the lucky winner claimed his prize.

I wandered around during the break, as at £4.50 the price of the lunch was outside my budget — besides, you can nearly buy ¼K of RAM for that! I asked a few people their views so far. Alistair Nicoll, who had travelled down from Nuneaton the previous day, said, "I found the organisation of the lectures a bit disappointing. I would have thought that one lecturer would have been better as I felt that some lectures tended to merge into others." I asked him about the kit itself and whether he would buy one. "No, I wouldn't buy one as I have already built my own system. However, I liked the use of the interrupt facility to assist single-steps," he replied. I next asked Neil Harrison of Swindon what he thought of the morning. "I found the general level of the lectures far too simple and thought that the NASCOM could have been demonstrated far more comprehensively," he said. In fact, during my walk I heard many people voicing this as we returned for the afternoon session.

Already running late, the session began with a talk by Tony Rundle on some uses for the microprocessor. This is a very difficult topic to cover, as the microprocessor has often been called "the tool looking for a job"! We, on this magazine, feel that this has been a neglected area in the past and so shall be concentrating on it in future issues. Anyway, Tony trooped out the old chestnut of central heating control * but also a number of other interesting potential applications such as photographic computation and exposure control.Tony is preparing a Tiny Basic for use with NASCOM and we hope to try it out when it arrives (big hint, Lynx!).

The final lecture of the day, presented by Phil Cooke was on interfacing to the real world. To my mind this was the most interesting lecture of the day. Here is another dark area for the microprocessor user. It deserves a little more illumination in the future. One of the ideas Phil Cooke produced was a very cheap but effective Digital to Analogue convertor; and then went on to add a PIO, comparator and some software to form an A to D converter very simply. Another point that he made was the ability of the processor to "talk-to" non-intelligent keyboards with a suitable programme. This system is used by NASCOM and certainly works very well. But the end drew nigh, and with only a passing reference to the interrupts we were into question time.

All the lecturers were on the stage and chaired by Gerry Tysoe. The questions from the floor were very wide-ranging, both in content and depth, and clearly showed the differing abilities and interests of the audience and the difficulties that the lecturers must have had in pitching their talks at the correct level. What emerged, though, were some additional facts about the NASCOM. For example, it possesses virtually no buffers on the busses. However, there are plans afoot for an expansion board which presumably will have its *inputs* buffered instead. This board will be able to take several options such as dynamic RAM and more EPROM. An EPROM programmer is also in the pipeline. Personally, I believe that the place to buffer is with the CPU, but then — of course — up will go the price.

There was much concern regarding the company's attitudes to suspect boards once assembled, for earlier on in the day we had been exhorted to 'throw away our 'scope and our AVO's'. Some of us were of the opinion that perhaps we might need them later on! Lynx's basic policy is that they will look at the boards. They will then replace faulty components free of charge. Where there has been the odd wiring error etc. then they will put it right for a small fee. Mass destruction with a plumber's blowlamp will be open to negotiation. This all sound very reasonable to me. One or two people asked whether NASCOM would interface with any of the development systems; which surprised me. If you can afford a development system then why are you thinking about NASCOM? I know that the true price of a NASCOM is £240 when you include the power supply and the nasty Chancellor's rake-off, but these other systems cost thousands. There were many other questions and clearly we could have carried on long into the night; but it was time to go.

In conclusion, a worthwhile day even though for some people it may not have turned out quite the way that they had anticipated. I feel that Lynx should consider whether perhaps just one or two lecturers would present the subjects more succinctly than the six they used then. I also think that they could perhaps benefit from a questionnaire to future 'delegates' as to the knowledge of the subject that they possess. A more comprehensive demonstration of NASCOM would also help. I think that Lynx have got a winner with NASCOM. It seems a shame that, at the last count, the promotional back-up did not quite do it justice.

* Does anyone in Europe *really* use one for this purpose?

THE ELEGANT MINMON

Neil Harrison

Z80 MINIMUM MONITOR

This 256 byte monitor was written to provide the minimum memory manipulation and program execution facilities for Z80 machine code programming. While lacking many of the frills of more elaborate monitors such as Zapple or Mikbug, it is ideal for the small system user who wants to get away from those leds and switches.

The Monitor needs an ASCII terminal device of some variety (Teletype, VDU etc.) and includes simple character input/output using Z80 I/O ports. It is almost certain that the user will have to modify these routines to suit his own system: some ideas on this are given later. By limiting the size of the monitor to 256 bytes the entire program can be put into a cheap 1702 EPROM, providing a resident monitor at minimum cost. With this in mind the program has been assembled to load at locations 0000H to 00FFH so that the monitor routines will be entered whenever the Z80 CPU is reset. The program can be used elsewhere in memory provided that references to absolute addresses are adjusted to suit. The addresses which must be changed are marked on the listing by a '*'.

Though more functions can be crammed into 256 bytes this Monitor is particularly "user proof", catching common errors like non hexadecimal input and writing to non existent memory.

Figure 1 is a fully assembled listing of the Minimum Monitor using standard Zilog mnemonics. Plenty of comment and explanation has been included in the listing so that with a little knowledge of the Z80 instruction set no further explanation should be needed here.

Using the Monitor.

When the monitor is running and ready to accept a command a '>' character is printed at the beginning of a new line on the terminal. There are two valid commands: 'E' to examine and alter memory, and 'G' to go to a memory location and start executing. Any other letter will simply restart the monitor. All addresses must be typed in as four hex characters and data as two hex characters.

The Examine command takes the following form:-

```
>Eaaaa
 -----
aaaa dd nn
     --
```

User input is underlined. 'aaaa' is a four character hexadecimal address: 'dd' is the contents of that location in hex. 'nn' is the user reply and will be one of the following:-

1) Two valid hex characters will be used as new data for that memory location, overwriting the original contents.

2) A single space character will print the next address and its contents.

3) A single '-' character will print the previous address and contents.

4) Any other character will restart the Monitor.

Example:-

```
>E0100                          (Examine data at address 100)
0100 CD (space)                 (Examine next location)
0101 23 (space)                    "       "       "
0102 02 03                      (Change 02 to 03)
0103 AF -                       (Examine previous loc.)
0102 03 -                          "       "       "
0101 23 -                          "       "       "
0100 CD (carriage return)       (Restart Monitor)
>                               (Awaiting a new command)
```

The Go to command forces the Z80 to start executing a program at some address specified by the user. After the address is entered, the monitor waits for a 'carriage return' character, allowing the user to abort the command if a mistake has been made.

```
>Gaaaa (carriage return)
 ----- ----------------
```

User input is underlined. 'aaaa' is a four character hex address at which execution is to start. The Monitor waits for a 'carriage return' before jumping to the user program: any character other than CR will restart the Monitor.

Errors

An asterisk (*) will be printed and the Monitor restarted if any of the following occur:-

1) A non-hexadecimal character found in an address or data input.

2) If the data entered into a location does not read back correctly. This checks for memory errors and attempts to write to nonexistent memory.

3) If a character other than 'carriage return' is typed after the address in the 'G' command.

Example:-

```
>E0140*                         ('O' isn't a hex character!)
>                               (Awaiting a new command)
```

Subroutines

One of the more desirable features of any monitor program is that the subroutines that it employs should be available for use in user programs. The following table is a list of all the subroutines in the Minimum Monitor, their entry addresses and what they do. Some of these are modified versions of equivalent routines for the 8080. The originals can be found in the AMSAT 8080 Debug Monitor, BYTE September 1976. The PRINT subroutine is not used by this particular version of the monitor and can be omitted if extra room is needed for I/O routines.

NAME	ADDRESS	FUNCTION
CRLF	0061	Types carriage return/line feed.
GETE	00E5	Gets a character from the keyboard and echoes it. Returns with the character in the A register.
GXB	008A	Gets two hex characters from the keyboard and returns them as a byte in A. If a non hex character is found the Carry flag is set and the character returned in A.
GXN	0074	Gets a hex character from the keyboard and returns it in the 4 least significant bits of A. If the character is not valid hex then returns with Carry set and the character in A.
GXW	009C	Gets 4 hex characters from the keyboard and returns a 16 bit number in the HL register pair. If a non hex character is typed then returns with Carry set and the bad character in A.
PRINT	00D8	Prints the ASCII message starting at the address in HL. Terminates on a Null character (00H).
SPACE	006C	Prints a single space.
TXB	0084	Prints the contents of A as two hex characters.
TXN	00BD	Prints the 4 least significant bits of A as a hex character.
TXW	00CD	Prints the contents of HL as 4 hex characters.

I/O Routines

To interface the Monitor to a particular system two subroutines are required, one to print the ASCII character in the A register and another to get a character from the console keyboard returning it in the A register. These routines should preserve all registers except the A register itself.

Simple routines to control I/O through Z80 ports 0 and 1 are included in the listing as examples. These routines assume that the console device is interfaced to the system as follows:-

Data input and output occurs through port 1.

Two bits in port 0 are used for device status:-

Bit 0 is a 1 whenever a keyboard character has been struck and indicates that data is ready to be collected.

Bit 7 is a 1 whenever the console output is ready to accept a character from the CPU.

Now what about modification for a different set up? If the system uses different I/O ports it is simple enough to change the existing references to ports 0 and 1 to those required. The bytes following the I/O instructions, currently NOPs, can be used to complement the data or status if the interface provides an inverted signal. If the device is memory rather than I/O addressed, then the entire IN or OUT instruction plus the NOP can be replaced by a memory load instruction. The character output routine, CHROUT, might become:-

```
CHROUT: PUSH    AF                      ;SAVE A & FLAGS
CHR01:  LD      A,(status address)      ;GET STATUS FROM MEMORY

        AND     80                      ;TEST THE READY BIT
        JR      Z,CHR01                 ;LOOP IF NOT READY
        POP     AF                      ;GET BACK A & FLAGS
        LD      (data address),A        ;OUTPUT THE CHARACTER
        RET                             ;RETURN TO CALLER
```

Some devices need quite complex I/O subroutines, far larger than can be fitted into the Minimum Monitor. A typical example of this is a 'memory mapped' TV display which may require several tens of bytes to put the character in the correct position on screen. There is no alternative but to put the routine into another ROM and replace the code at CHROUT with a jump to the TV output routine. The RET instruction at the end of the display subroutine will then return to the calling program.

Programmable I/O chips are becoming increasingly popular: however they do have to be 'set up' by the CPU for a particular application before they can be used. Eleven bytes are allocated for this at the beginning of the monitor so that any such chips are initialised every time the Z80 is reset.

Expansion

In time the user will probably want to expand the facilities of the Minimum Monitor. This is easily done by changing the conditional Jump instruction at 004B to jump to the address of the new routine rather than the start of the monitor. When the new routine is finished it should jump back to BEGIN ready for the next command. Any number of extra commands can be added in this way, limited only by the number of characters on the keyboard.

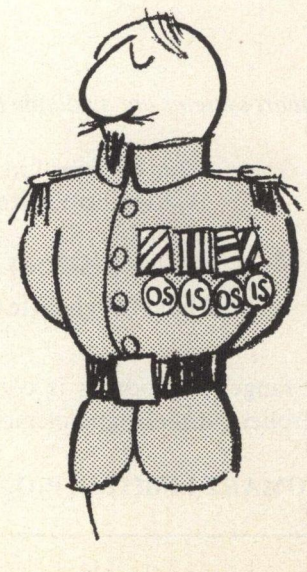
Neil Harrison Dec 77.

```
0000                    ;-------------------------------------------------
0000                    ;
0000                    ;       MINIMUM MONITOR PROGRAM FOR THE Z80
0000                    ;
0000                    ;       NEIL HARRISON           DEC 77
0000                    ;
0000                    ;-------------------------------------------------
0000                    ;       DEFINITIONS:-
0000                    CR      =       0D              ;CARRIAGE RETURN
0000                    LF      =       0A              ;LINE FEED
0000                    SPA     =       20              ;SPACE
0000                    STACK   =       3FF             ;STACK POSITION FOR 1K SYSTEM
0000                    ;
0000                    .       =       0000
0000                    ;
0000 00                 START:  NOP                     ;HERE'S SOME SPARE SPACE
0001 00                         NOP                     ;THAT YOU CAN USE TO
0002 00                         NOP                     ;INITIALISE I/O DEVICES
0003 00                         NOP                     ;IF NECESSARY.
0004 00                         NOP                     ;
0005 00                         NOP                     ;
0006 00                         NOP                     ;
0007 00                         NOP                     ;
0008 00                         NOP                     ;
0009 00                         NOP                     ;
000A 00                         NOP                     ;
000B 31 FF 03           BEGIN:  LD      SP,STACK        ;SET UP STACK POINTER.
000E CD 61 00  *                CALL    CRLF            ;NEW LINE ON CONSOLE
0011 3E 3E                      LD      A,'>'           ;PUT PROMPT INTO A
0013 CD E8 00  *                CALL    CHROUT          ;PRINT IT
0016 CD E5 00  *                CALL    GETE            ;GET A COMMAND LETTER INTO A
0019 FE 45                      CP      'E'             ;IS IT AN 'E'
001B 20 2C                      JR      NZ,GOTO         ;IF NOT SKIP TO GOTO
001D CD 9C 00  *                CALL    GXW             ;GET ADDRESS FOR EXAMINE INTO HL
0020 38 38                      JR      C,ERROR         ;ERROR IF CARRY SET
0022 CD 61 00  *        ADDR:   CALL    CRLF            ;NEW LINE
0025 CD CD 00  *                CALL    TXW             ;TYPE ADDRESS IN HEX
0028 CD 6C 00  *                CALL    SPACE           ;PRINT SPACE
002B 7E                         LD      A,(HL)          ;GET DATA AT CURRENT ADDR.
002C CD B4 00  *                CALL    TXB             ;PRINT IT IN HEX
002F CD 6C 00  *                CALL    SPACE           ;PRINT SPACE
0032 CD 8A 00  *                CALL    GXB             ;GET NEW DATA OR COMMAND CHAR
0035 38 07                      JR      C,EXAM1         ;IF CARRY SET ITS A COMMAND
0037 77                         LD      (HL),A          ;PUT NEW DATA INTO MEMORY
0038 BE                         CP      (HL)            ;CHECK THAT IT GOT THERE OK
0039 20 1F                      JR      NZ,ERROR        ;IF NOT THEN ERROR
003B 23                 NEXT:   INC     HL              ;INC ADDRESS COUNTER
003C 18 E4                      JR      ADDR            ;AND GO ROUND AGAIN
003E FE 20              EXAM1:  CP      SPA             ;IS IT SPACE?
0040 28 F9                      JR      Z,NEXT          ;IF SO DO NEXT ADDRESS
0042 FE 2D                      CP      2D              ;IS IT A '-'?
0044 20 C5                      JR      NZ,BEGIN        ;IF NOT THEN RESTART MONITOR
0046 2B                         DEC     HL              ;IF SO DO PREVIOUS ADDRESS
0047 18 D9                      JR      ADDR            ;GO ROUND AGAIN
0049 FE 47              GOTO:   CP      'G'             ;IS IT A 'G'?
004B C2 0B 00  *                JP      NZ,BEGIN        ;IF NOT THEN RESTART MONITOR
004E CD 9C 00  *                CALL    GXW             ;GET GOTO ADDRESS INTO HL
0051 38 07                      JR      C,ERROR         ;ERROR IF CARRY SET
0053 E5                         PUSH    HL              ;PUT GOTO ADDRESS ON STACK
0054 CD F5 00  *                CALL    CHRIN           ;WAIT FOR A 'CR'
0057 FE 0D                      CP      CR              ;IS IT 'CR'?
0059 C8                         RET     Z               ;"RETURN" TO GOTO ADDR IF SO
005A 3E 2A              ERROR:  LD      A,'*'           ;'*' MEANS ERROR
005C CD E8 00  *                CALL    CHROUT          ;PRINT IT
005F 18 AA                      JR      BEGIN           ;RESTART MONITOR
                        ;
0000                    ;       SUBROUTINES FROM HERE ON
0000                    ;
0000                    ;       'CRLF' PRINTS CARRIAGE RETURN/LINE FEED ON
0000                    ;       CONSOLE DEVICE.
0000                    ;
0061 3E 0D              CRLF:   LD      A,CR            ;CARRIAGE RETURN
0063 CD E8 00  *                CALL    CHROUT          ;PRINT IT
0066 3E 0A                      LD      A,LF            ;LINE FEED
0068 CD E8 00  *                CALL    CHROUT          ;PRINT IT
006B C9                         RET
                        ;
0000                    ;       'SPACE' PRINTS A SINGLE ' ' ON THE CONSOLE
0000                    ;
006C F5                 SPACE:  PUSH    AF              ;SAVE A
006D 3E 20                      LD      A,SPA           ;PUT A SPACE IN A
006F CD E8 00  *                CALL    CHROUT          ;PRINT IT
0072 F1                         POP AF                  ;RESTORE A
0073 C9                         RET
                        ;
0000                    ;       'GXN' GETS A HEX ASCII CHARACTER FROM THE CONSOLE
0000                    ;       AND RETURNS THE HEX VALUE IN THE 4 LSBS OF THE
0000                    ;       ACCUMULATOR. IF AN ILLEGAL CHARACTER IS FOUND THE
0000                    ;       CARRY BIT IS SET AND THE ILLEGAL CHAR RETURNED IN A.
0000                    ;
0074 CD E5 00  *        GXN:    CALL    GETE            ;GET A CHAR (ECHOED)
0077 FE 30                      CP      30              ;IS IT LESS THAN '0'
0079 D8                         RET     C               ;IF SO RETURN WITH CARRY SET
007A FE 3A                      CP      3A              ;IS IT LESS THAN '9'+1
007C 38 09                      JR      C,GX1           ;IF SO THEN OK
007E FE 41                      CP      41              ;IS IT LESS THAN 'A'
0080 D8                         RET     C               ;IF SO RETURN WITH CARRY SET
0081 FE 47                      CP      47              ;IS IT LESS THAN 'G'
0083 3F                         CCF                     ;COMPLEMENT CARRY
0084 D8                         RET     C               ;IF GREATER THAN 'F' RETURN
0085 D6 07                      SUB     7               ;ADJUST HEX LETTERS
0087 D6 30              GX1:    SUB     30              ;ADJUST HEX NUMBERS
0089 C9                         RET
                        ;
0000                    ;       'GXB' RETURNS 2 HEX ASCII CHARACTERS AS A BYTE
0000                    ;       IN A. IF A NON HEX CHARACTER IS FOUND THE CARRY BIT
0000                    ;       IS SET AND THE ILLEGAL CHAR RETURNED IN A.
0000                    ;
008A CD 74 00  *        GXB:    CALL    GXN             ;GET A HEX NIBBLE
008D D8                         RET     C               ;RETURN IF ERROR
008E C5                         PUSH    BC              ;SAVE B & C
008F 07                         RLCA                    ;ROTATE NIBBLE
0090 07                         RLCA                    ; INTO TOP
0091 07                         RLCA                    ;  HALF OF
0092 07                         RLCA                    ;    ACCUMULATOR.
0093 47                         LD      B,A             ;PUT RESULT IN B
0094 CD 74 00  *                CALL    GXN             ;GET ANOTHER NIBBLE
0097 38 01                      JR      C,GX2           ;ABORT IF ERROR
0099 80                         ADD     B               ;ADD IN FIRST NIBBLE
009A C1                 GX2:    POP     BC              ;RESTORE B & C
009B C9                         RET
                        ;
0000                    ;       'GXW' RETURNS WITH 4 HEX ASCII CHARACTERS AS
0000                    ;       A WORD IN HL. IF AN NON HEX CHARACTER IS FOUND
0000                    ;       THE CARRY BIT IS SET AND THE ILLEGAL CHAR RETURNED
0000                    ;       IN A.
0000                    ;
009C B7                 GXW:    OR      A               ;CARRY = 0
009D F5                         PUSH    AF              ;SAVE A & FLAGS
009E CD 8A 00  *                CALL    GXB             ;GET HIGH BYTE
00A1 67                         LD      H,A             ;PUT IT IN H
00A2 30 04                      JR      NC,GX3          ;JUMP IF OK
00A4 F1                         POP     AF              ;RESTORE A & FLAGS
00A5 7C                         LD      A,H             ;RETURN WITH BAD CHAR IN A
00A6 37                         SCF                     ;SET CARRY TO INDICATE ERROR
00A7 C9                         RET
00A8 CD 8A 00  *        GX3:    CALL    GXB             ;GET LOW BYTE
00AB 6F                         LD      L,A             ;PUT IT IN L
00AC 30 04                      JR      NC,GX4          ;JUMP IF OK
00AE F1                         POP     AF              ;RESTORE A & FLAGS
00AF 7D                         LD      A,L             ;RETURN WITH BAD CHAR IN A
00B0 37                         SCF                     ;SET CARRY TO INDICATE ERROR
00B1 C9                         RET
00B2 F1                 GX4:    POP AF                  ;RESTORE A & FLAGS
00B3 C9                         RET
                        ;
0000                    ;       'TXB' PRINTS THE CONTENTS OF THE ACCUMULATOR
0000                    ;       AS TWO HEX CHARACTERS ON THE CONSOLE DEVICE.
0000                    ;
00B4 F5                 TXB:    PUSH    AF              ;SAVE A & FLAGS
00B5 0F                         RRCA                    ;ROTATE TOP NIBBLE
00B6 0F                         RRCA                    ; INTO BOTTOM
00B7 0F                         RRCA                    ;  NIBBLE OF
00B8 0F                         RRCA                    ;    ACCUMULATOR
00B9 CD BD 00  *                CALL    TXN             ;PRINT IT
00BC F1                         POP     AF              ;RESTORE A & FLAGS
                        ;
0000                    ;       'TXN' PRINTS THE 4 LSBS OF THE ACCUMULATOR
0000                    ;       AS A HEX CHARACTER ON THE CONSOLE DEVICE.
0000                    ;
00BD F5                 TXN:    PUSH    AF              ;SAVE A & FLAGS
00BE E6 0F                      AND     0F              ;LOSE TOP 4 BITS
00C0 FE 0A                      CP      0A              ;IS IT A NUMBER?
00C2 38 02                      JR      C,TX1           ;IF SO JUMP
00C4 C6 07                      ADD     7               ;IF NOT ADD OFFSET FOR LETTERS
00C6 C6 30              TX1:    ADD     30              ;CONVERT TO ASCII
00C8 CD E8 00  *                CALL    CHROUT          ;PRINT IT
00CB F1                         POP     AF              ;RESTORE A & FLAGS
00CC C9                         RET
                        ;
0000                    ;       'TXW' PRINTS THE CONTENTS OF THE HL REGS AS
0000                    ;       4 HEX CHARACTERS ON THE CONSOLE DEVICE.
0000                    ;
00CD F5                 TXW:    PUSH    AF              ;SAVE A & FLAGS
00CE 7C                         LD      A,H             ;PUT HIGH BYTE OF WORD INTO A
00CF CD B4 00  *                CALL    TXB             ;PRINT IT
00D2 7D                         LD      A,L             ;PUT LOW BYTE OF WORD INTO A
00D3 CD B4 00  *                CALL    TXB             ;PRINT IT
00D6 F1                         POP     AF              ;RESTORE A & FLAGS
00D7 C9                         RET
                        ;
0000                    ;       'PRINT' OUTPUTS A MESSAGE TO THE CONSOLE
0000                    ;       DEVICE. ON ENTRY HL PONTS TO THE FIRST
0000                    ;       CHARACTER OF THE MESSAGE. A NULL (00)
0000                    ;       TERMINATES THE MESSAGE STRING.
0000                    ;
00D8 F5                 PRINT:  PUSH    AF              ;SAVE A & FLAGS
00D9 7E                 PRIN1:  LD      A,(HL)          ;GET A CHARACTER
00DA B7                         OR      A               ;IS IT A 0 (NULL)
00DB 28 06                      JR      Z,PRIN2         ;IF SO THEN END OF MESSAGE
00DD CD E8 00  *                CALL    CHROUT          ;PRINT THE CHARACTER
00E0 23                         INC     HL              ;BUMP POINTER FOR NEXT CHAR
00E1 18 F6                      JR      PRIN1           ;GO ROUND AGAIN
00E3 F1                 PRIN2:  POP     AF              ;RESTORE A & FLAGS
00E4 C9                         RET
                        ;
0000                    ;       'GETE' GETS A CHARACTER FROM THE CONSOLE
0000                    ;       AND ECHOES IT.
0000                    ;
00E5 CD F5 00  *        GETE:   CALL    CHRIN           ;GET A CHAR FROM KEYBOARD
                        ;
0000                    ;       TYPICAL CHARACTER OUTPUT ROUTINE USING PORTS 0 AND 1.
0000                    ;       BIT 7 OF PORT 0 IS THE 'READY' BIT, IT MUST GO HIGH
0000                    ;       WHEN THE OUTPUT DEVICE IS READY FOR A CHARACTER.
0000                    ;       EXPECTS AN ASCII CHAR IN A ON ENTRY.
0000                    ;
00E8 F5                 CHROUT: PUSH    AF              ;SAVE A & FLAGS
00E9 DB 00              CHRO1:  IN      A,(0)           ;GET CONSOLE STATUS
00EB 00                         NOP                     ;PADDING
00EC E6 80                      AND     80              ;TEST READY BIT
00EE 28 F9                      JR      Z,CHRO1         ;LOOP IF NOT READY
00F0 F1                         POP     AF              ;GET A & FLAGS BACK
00F1 D3 01                      OUT     (1),A           ;PRINT IT
00F3 00                         NOP                     ;PADDING
00F4 C9                         RET
                        ;
0000                    ;       TYPICAL CHARACTER INPUT ROUTINE USING PORTS
0000                    ;       0 AND 1. BIT 0 OF PORT 0 MUST GO HIGH WHEN A
0000                    ;       KEYBOARD CHARACTER HAS BEEN STRUCK. RETURNS
0000                    ;       THE ASCII CHARACTER IN A.
0000                    ;
00F5 DB 00              CHRIN:  IN      A,(0)           ;GET CONSOLE STATUS (PORT 0)
00F7 00                         NOP                     ;PADDING
00F8 E6 01                      AND     01              ;TEST DATA AVAILABLE BIT
00FA 28 F9                      JR      Z,CHRIN         ;LOOP IF NOT READY
00FC DB 01                      IN      A,(1)           ;GET THE CHARACTER (PORT 1)
00FE 00                         NOP                     ;PADDING
00FF C9                         RET
00FF 00                         .END
```

SYMBOL TABLE

```
ADDR    = 0022          BEGIN   = 000B          CHRIN   = 00F5
CHRO1   = 00E9          CHROUT  = 00E8          CR      = 000D
CRLF    = 0061          ERROR   = 005A          EXAM1   = 003E
GETE    = 00E5          GOTO    = 0049          GX1     = 00B7
GX2     = 009A          GX3     = 00A8          GX4     = 00B2
GXB     = 008A          GXN     = 0074          GXW     = 009C
LF      = 000A          NEXT    = 003B          PRIN1   = 00D9
PRIN2   = 00E3          PRINT   = 00D8          SPA     = 0020
SPACE   = 006C          STACK   = 03FF          START   = 0000
TX1     = 00C6          TXB     = 00B4          TXN     = 00BD
TXW     = 00CD
```

0 ERRORS.



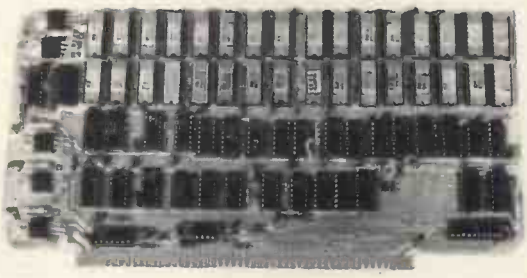
comart

SPECIALISTS IN MICROCOMPUTERS

Cromemco Z2 Processor

* Power of Z80 Processor Card with Power-on-jump
* S100 Bus 21 slot Backplane to configure the largest system
* 30 Amp Power Supply

  Z2 = Z80 + S100
* Range of Analogue & Digital Interfaces
* Powerful Disc based Software



Dynabyte Memory

16K Dynamic for Altair, Imsai, Poly 88.

16K Static with memory backing facilities, and individual 4K boundary location and write protect.

32K Static for large system economy.

Assembled, tested and 1 year's warranty at list prices.

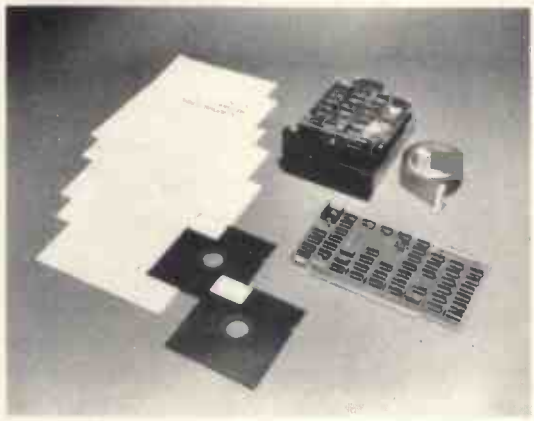
North Star Mini-diskette System

With operating system, File & String handling

BASIC

For all 8080 & Z80 S100 Bus Systems.

For greater calculation speed use the Hardware Floating Point Board.



Comart systems are available by direct mail order or through local suppliers:-

Xitan Systems Ltd.,
31, Elphinstone Road,
Highcliffe,
Dorset, BH23 5LL
Phone: Highcliffe 77126

Computer Bits Ltd.,
41, Vincent Street,
Yeovil,
Somerset.
Phone: Yeovil (0936) 26522

Our range of products is constantly being expanded - write or phone for catalogue of latest controllers, diskette and memory systems and options.

COMART LIMITED, P.O. BOX 2, ST. NEOTS, CAMBS. PE19 4NY TEL. 0480 74356

A Computer That Means Business

D. Jackman

In a world of ever rising costs and a general fall in value for money, it is refreshing to write about a product that represents excellent value. The expanding world of Microprocessors has provided the springboard for greater advances and wider choice of computer equipment for the Small Business Systems and Applications.

It is possible to provide complete systems for Small Business applications very economically, by providing the workhorse Computer tool for a very wide range of tasks, covering many different types of business and operations.

Advancement can only be made possible by equipment suppliers recognising the needs of Small Business, and then taking steps to produce equipment that is economical, reliable, flexible and fully supported and backed up by excellent software and system support.

CASU Electronics Limited, a totally independent British Company, has made every effort to produce a Small Business System that is in keeping with the details outlined above. The specification for this System is called SUPER C IN SMALL BUSINESS APPLICATIONS and incorporates the following design criteria:-

1(a) Hardware
- (i) Modular Construction
- (ii) De-centralized Power Supply System
- (iii) High Speed CPU operation (250nSec)
- (iv) High Speed Memory (64K Bytes)
- (v) Floppy Diskette Mass Storage (½ Mega-byte)
- (vi) Priority Interrupt Structure (Z80 Mode 2)
- (vii) Multipurpose Intelligent Controller Design
- (viii) D.M.A. Video Controller
- (ix) Multipurpose System Controller

1(b) Software
- (i) Powerful Monitor and Debug Program
- (ii) TEXT EDITOR + RE-LOCATABLE ASSEMBLER
- (iii) EXTENDED DISC BASIC
- (iv) FORTRAN IV (ANSI IV)
- (v) Application Packages
- (vi) Specialized BASIC Programs
- (vii) Powerful DISC Operating System

After the system design specification was completed CASU investigated the availability of a well defined System BUS Structure and decided to standardize on the 'S100' bus structure; this provides an excellent and well defined bus structure. The S100 bus system is a Multi-channel 100 pin connector system. It was decided at an early stage of the design to allow system enhancement and expansion above and beyond the basic system; this will allow Small Business Systems to expand and adapt to changing requirements.

Twenty-two full S100 channels are available on the SUPER C chassis. In basic SUPER C systems only 10 channels are utilized, leaving 12 channels free. Another important factor is packaging. To provide a choice, SUPER C systems can be Desk Top, Rack Mounting or special low level pedestal Desk fitting. Once again flexibility is the keyword.

With the Bus Structure, Power Supply, Chassis and Front Panel completely defined, the next task was to design a range of S100 compatible controllers

that would form the backbone of the SUPER C System. The following controllers were specified:-

(i) Z80 Controlled CPU System
(ii) Dynamic RAM in 16K byte blocks. This provides 64K bytes of directly addressable Memory. Special 'bank select' features allow up to 128K bytes of Memory to be specified.
(iii) Specially designed System Control Board that provides:-
  (a) FRONT PANEL INPUT/OUTPUT
  (b) SERIAL DATA INPUT/OUTPUT (RS232 on Current Loop)
  (c) PROGRAMMABLE REAL TIME CLOCK INTERVAL TIMER
  (d) BOOTSTRAP EPROM
  (e) SOFTWARE INTERRUPT FEATURE
(iv) General Purpose I/O Controller that can be optioned for a wide range of parallel devices such as Printers, Punches, Reader, etc.
(v) Floppy Diskette and Intelligent Controller with Disk Operating System
(vi) Card Reader Controller
(vii) High Speed Paper Tape Reader Controller 350/700 characters/second
(viii) Intelligent Communications Controller

CASU have chosen the ZILOG Z80 Microprocessor for the following reasons:-

(a) The Z80 is fully software compatible with the popular 8080A CPU.
(b) The Z80 component set is superior in both SOFTWARE and HARDWARE capabilities.
(c) The Z80 can execute 158 different instructions types including all 78 of the 8080 CPU.
(d) The Z80 has WAIT, HOLD and HALT capability, this allows easy and effective control for DIRECT MEMORY ACCESS or for operation with slow MEMORY systems.
(e) The Z80 has very powerful INTERRUPT CAPABILITY, and can operate under three different levels of Interrupt. The SUPER C system employs mode 2 level interrupts; in this mode 128 levels of priority interrupt can be handled.
(f) The Z80 can operate at 250nSec clock speed. SUPER C system operates at full speed on all devices. This enables the very best execution times on all instructions.

For ease of implementation and maximum impact, Small Business Systems require high speed Random Access storage media for both Program storage and Data Base Material. SUPER C Systems incorporate a FLOPPY DISKETTE Sub-System that provides ½ Million bytes of storage, access to this media at very high speed. Complementing the speed of access CASU offers a powerful DISC OPERATING SYSTEM that is a completely free-standing

Intelligent Sub-System. This sub-system provides a conversation type of interface between the SUPER C Software and the Floppy Diskette.

This reduces the software effort required to implement a Small Business System, and reduces main memory requirements normally associated with DISC Operating Systems.

SUPER C Systems offer the following DISC commands:-

FORMAT, COPY, DELETE, OPEN FILE, CLOSE FILE, PACK, POSITION, DIRECTORY, SAVE, LOAD, READ, WRITE, ERASE, MERGE, NAME, INPUT, OUTPUT AND MODE

The full Disc Command set is supplied with all SUPER C Systems, this is important for customer-generated program requirements.

SUPER C Systems incorporate the latest technology, the fastest DISKETTE System available, coupled with the finest peripheral controllers available. This is taking advantage of the latest devices available, developing the hardware around such devices, and presenting to the Small Business Sector of the market a complete package.

Having specified the hardware requirements the final development phase required software specifications to be completed, so that the most effective system could be produced. CASU have taken the Software Agency for a range of Software packages designed for Z80 based processors. CASU have modified and enhanced these packages to be completely complementary with SUPER C Systems, providing a complete business system.

Super C Systems represent a high degree of development effort — this effort could have been wasted if the product was not properly supported. SUPER C is therefore supported at all levels:-

(a) Design
(b) System
(c) Field Engineering

This provides a 'total care concept' for the Small Business System.

Software support

CASU offers fully supported and documented Software packages, together with special programs to customer specifications.

Small business systems can take advantage of the powerful DISKETTE based BASIC AND FORTRAN Packages, both packages are fully compatible with the Super C Systems and the full range of peripherals offered.

SUPER C is, therefore, the ideal tool for many small business systems and can provide a flexible and cost effective solution for many companies currently relying on time sharing systems or larger computer Complexes over telephone line communications.

Outlined below are some ideas for typical SUPER C Systems:

(a) **Single Program Mode**

In certain applications it is advantageous for the system to operate in one fixed mode and the system should always enter this mode on reset or power restart. One classic example of this is a requirement for BASIC only. SUPER C can be supplied with a bootstrap program that will call the BASIC Program from the diskette media. Once the BASIC program has been loaded it will be executed and be instantly ready for operation.

(b) **Multi-Program Mode**

SUPER C can be supplied with a special bootstrap feature that will call a directory from the diskette media. This will allow the Manager/Engineer/Operator/Clerk to select from a suite of programs; this provides a high degree of flexibility essential in a small system that must be adaptable to a multi-task situation.

(c) **EPROM Based Program**

SUPER C Systems can be supplied with EPROM based prorams if the need arises.

Conclusions

SUPER C Systems have been specially designed for the small business, recognising the needs of the small business. Care has been taken to provide flexibility coupled with cost effectiveness. Cost is very important and SUPER C is a worker system that is very productive

Costing

The Basic SUPER C System costing is *£6,950.00.* This buys the following equipment fully tested:-

(i) Full chassis with 22 channel sub frame
(ii) Power Supply Unit capable of running 22 controllers
(iii) Front Panel + controller + Bootstrap
(iv) Z80 (CPU) + Automatic Power Restart
(v) 32K bytes of Dynamic RAM
(vii) Printer Controller + High Speed Printer, 70 character/second
(vii) ½ Mega-byte Floppy Diskette + Controller+ Diskette Operating System
(viii) Full set of cables for all devices.

Future Developments

It is CASU Limited's intention to produce other products to enhance our product line. Special controllers can be produced to customer specification.

Further details of SUPER C Systems can be obtained from:- CASU Electronics Ltd., Ferndown, Northwood Hills, Middlesex, England. Northwood 28994.

It's time computers came to the community

THE GATES OF REASON

Michael Whitney

AN INTRODUCTION TO COMPUTER LOGIC

Part 1: Binary logic devices

It will be as well to say at the start, for the benefit of those readers who do not realize the fact already, that I cannot hope, in a short series of articles, to give a thorough description of any real modern computer. However, there is a great interest in the questions of how a mere machine can be designed to carry out the arithmetical and logical operations which are needed for the tasks a computer is called upon to perform; and it should be possible to describe enough typical circuits — arithmetical, logical and control — to give the reader a good insight into the logic of a complete machine. Many elementary texts on the subjects either fail to show how the simple circuits they describe are linked to form a complete machine; or they carry the matter to a higher degree of complexity than is required by a reader wishing only to gain an understanding of the principles involved. I hope to avoid these pitfalls.

Without some acquaintance with the binary number system, it is impossible to go very far in understanding computer logic; so I will deal with this topic first. You will find it helpful to have pencil and paper by you as you read.

THE BINARY NUMBER SYSTEM

"M. JOURDAIN: What? When I say 'Nicole, bring my slippers, and give me my nightcap', that's prose?
PROFESSOR: Indeed it is.
M. JOURDAIN: Good Lord! I've been speaking prose for more than forty years, and I had no idea."

MOLIÈRE: *Le Bourgeois Gentilhomme*

M. Jourdain's astonishment is often shared by people who find out that they have been doing arithmetic in decimal all their lives, and that other ways of doing it exist. In fact, an infinite number of systems is possible; a number system can have *any* whole number as its base; and the binary system differs from the decimal only in that it uses the base 2 instead of the base 10.

What does this mean in practice? First, the decimal system needs *ten* symbols — we use the digits 1 to 9, and the cypher, 0; the binary system needs only *two* — the binary digit 1 (with the same meaning as the decimal digit 1), and, again, the cypher, 0. Second, if we select any decimal integer — say 6 — and put a zero after it — making 60 in this case — then we have multiplied the original number by *ten*. Doing the same with a binary integer multiplies it by *two* — 10 is the binary equivalent of decimal 2. *In all other respects, the binary system is identical to the decimal system.* Here is a table containing the first twelve positive integers in the two systems: the binary numbers in the left-hand column have the same values as the decimal numbers on the same line in the right-hand column:

BINARY	DECIMAL
1	1
10	2
11	3
100	4
101	5
110	6
111	7
1000	8
1001	9
1010	10
1011	11
1100	12

Study the binary numbers in this table: look for the pattern of change that takes place as the numbers increase. Note, for example, that, when 1 is added to binary 111, the result is 1000. To understand how this occurs, compare what happens when 1 is added to decimal 999; we add 1 to the right-most digit, giving 10; we write down the 0, and 'carry' the 1 to the next column on the left; the same thing happens in the second and third columns, so that we end up with three zeros, the final 1 being carried into the fourth column, giving decimal 1000. Adding 1 to binary 111 operates in the same way: we add a 1 to the right-most column, giving decimal 2; but there is no 2 in the binary system — 1 is the highest valued digit. The binary result is 10, and we must therefore write 0 in this position in the result, and *carry* 1 to the next column on the left; and so on. Select a few other binary numbers from the table, and try adding 1, following the same procedure; check that the result in each case is the next number in the table.

The following binary addition is almost as straightforward as adding 1: try it:

```
  110
+  11
-----
=  ?
-----
```

The correct answer is 1001 (in decimal, 6+3=9). When you have got it right, try:

```
  111
+  11
-----
=  ?
-----
```

This is only slightly more tricky. The right-most column is straightforward: 1+1=0, *carry* 1. But the next column on the left is 1+1, plus the 'carried' 1; what do we do here? 1+1+1= decimal 3, or binary 11; so we write 1, and carry 1, just as in decimal. Complete the calculation — the correct answer is 1010 (in decimal, 7+3=10).

You have now carried out every possible operation required for adding together any two binary integers. Table 1.1 sets out the eight possible combinations which can arise in any single column in the addition of two binary numbers; or,

Input			Output	
X	Y	Ci	Co	R
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Table 1.1 Bit addition — input-output table

to put it another way, in the addition of two binary digits *and* a 'carry'. It is worth looking at this table closely — there will be a number of similar tables later in this series. The title for the table introduces the term 'bit'; this is simply a contraction of the words 'binary digit'. 'Bit' = 'binary digit'. The rest of the table is fairly easy to understand. There are three columns under the heading 'Input': the X column is the first, or 'upper', of the two bits being added; the Y column is the second, or 'lower', bit; and the Ci, or 'carry-in', column is the carry forward from the previous bit-addition. Under 'Output', the R column gives the result, or sum; and the Co column is the 'carry-out', ie, the carry-forward required. Each line of the table gives one of the eight possible combinations of input bit-values, together with the result and carry-forward required for that combination. For example, the *sixth* row of the table indicates that, if the upper bit is 1, the lower 0, and 1 has been brought forward from the previous column's addition, then a 0 should be written in the result, and 1 should be *carried forward*.

All modern digital computers operate in binary. Some of them, it is true, appear to operate in decimal; but they are in fact using a special form of binary-*coded* decimal which allows the fundamental operations to be in binary (I hope to say more of this in a later article). This universal use of binary may seem surprising at first sight, since binary integers are obviously longer than their decimal equivalents. Would this not suggest that circuitry for binary operations would be more complicated than for decimal? Not necessarily. Look again at Table 1.1 — the input-output table for a bit-adder — and imagine what a similar table for a digit-adder would be like. It would have to allow for ten possible values of X — the digits 2 to 9, as well as 0 and 1; and, for each of these ten X-values, it would have to allow for the same ten possible values for Y, making a total of 100 possible combinations. In the complete table, each combination would have to appear twice — once with a 'carry' (Co) value of 0, and once again with a 'carry' value of 1. The decimal digit addition table would therefore have to provide for 200 possible input combinations, as compared with only 8 for the bit addition table; we would therefore expect any circuit designed to add two bits together to be simpler than a circuit to add two decimal digits, and this will go some way to compensate for the fact that the equivalent binary numbers will have more digits.

Given this fact, there are very good practical reasons for using binary rather than decimal. To perform operations on digits, we must have a way of indicating their values, and, in the decimal system, we would need ten separate recognizable indications. For example, we might decide that the digit 1 shoud lbe represented by 1 volt on the signal line, 2 by two volts, 7 by seven volts, and so on. It would not be impossible technically to employ such a system. But our circuits must be absolutely reliable in operation. In the course of time, electronic components deteriorate, and this can cause voltage fluctuations — with results which, obviously, would not lead to absolute reliability in this type of circuit. These problems can be circumvented by proper circuit design, but only at the cost of adding complication.

Circuitry designed for binary operations, on the other hand, needs to provide for only two separate identifiable signals, and we can choose, for example, that a pulse of more than three volts will mean a binary 1, whilst *any* other pulse, or no pulse at all, will mean zero. We can then design our circuits to put out a six-volt pulse for a 1, and no pulse for a zero, so leaving a large margin of safety to allow for component deterioration. The need for additional complexity in the circuits merely to make them reliable is, in this way, much reduced. For these reasons, binary operation, in one form or another, is the universal choice of circuit designers.

LOGIC GATES

" ' Contrariwise,' continued Tweedledee, 'if it was so, it might be; and if it were so, it would be: but as it isn't, it ain't. That's logic.' "

Computer logic is not quite as complicated as it may seem. The sheer quantity of circuitry within a typical machine appears to belie this; but the fact is that the bulk of it is made up of a few simple patterns, repeated a large number of times. These small patterns are made up themselves from only half-a-dozen or so different kinds of device. I am going to describe three kinds of elementary device — the inverter, or NOT gate, the AND gate, and the OR gate — and then show how they can be connected

together to form some of the simple patterns, or *logic networks,* of which the circuitry of a computer is largely composed.

The clock As well as the signal input and output connections, the devices I am about to describe require a special *timing* signal. This signal is the same for every device, and is supplied by the *clock* (sometimes called a *mill)* down the timing, or T, line The signal consists of a sequence of voltage pulses, each pulse lasting exactly the same length of time, and being followed by an equal interval during which the line is at zero volts. This signal may be provided by having a steady voltage supply with an electronically-controlled switch in it, which goes *on* for one time interval and then *off* for the next, and so on. In practice, the time-interval will be very short — less than a millionth of a second in a modern computer — but its actual duration need not concern us for the moment.

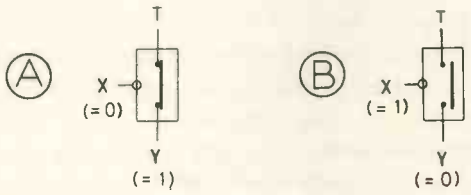


Fig. 1.1

The NOT gate, or inverter Examine Fig. 1.1A and 1.1B. The diagrams show a box with three lines leading into it — the T line (which carries the clock-pulse described above), the X line (which carries the incoming signal) and the Y line (which takes the outgoing signal). Inside the box, the T and Y lines are connected together, through a switch — the switch is in the 'on' position in Fig. 1.1A, and is 'off' in Fig. 1.1B. When the switch is on, a timing pulse arriving on the T line will pass through the device onto the output Y line; when the switch is off, no pulse will go out. The position of the switch depends on the voltage appearing on the input X line: if this is at zero volts, the switch remains on, as in diagram 1.1A. If, however, a voltage pulse appears on the X line, similar to the one on the timing line and simultaneously with it, then the switch will be 'pushed' off, and held off as long as the pulse is present, and the timing pulse will be prevented from reaching the Y line. Fig. 1.1B shows the device as it is during the presence of a pulse on the X line.

It will be seen that, for each pulse arriving on the T line, there are two possibilities with this device: either a pulse arrives simultaneously on the X line, preventing any signal on the Y line; or no signal arrives on the X line, so permitting a pulse to go out on the Y line. Pulse in = no pulse out; no pulse in = pulse out.

We want our device to manipulate *bits,* so we must decide how we are to represent 0 and 1 on the signal lines. There are, in fact, two options available, but let us agree that:

if a pulse occurs on a signal line simultaneously with a timing pulse on the T line, we will regard the signal line as carrying a 1;

if the signal line has no pulse simultaneous with the timing pulse, we will regard it as carrying a zero.

With this convention agreed, we can now describe this device in a more useful way, as follows:

if a 1 is input, a zero will be output;

if a zero is input, a 1 will be output.

Table 1.2 represents this in the format of an input-output table. This device is called an *inverter,* for obvious reasons;

Input	Output
X	Y
0	1
1	0

Table 1.2 NOT gate (or inverter) — input-output table

it can alternatively be called a *NOT gate,* since, with only two possible signal values, we can say that it puts out a 1 only when a 1 is NOT input.

Fig. 1.1

Fig. 1.1C shows how the NOT gate will be marked on logic network diagrams later in this series. Note that the T line connection is not shown; this is because this line performs only a timing and pulse-supply function, and affects the logical performance of the gate no more than does the power-supply line for the transistor in the gate.

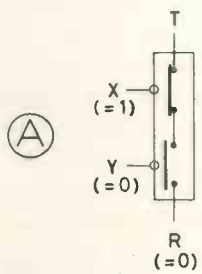


Fig. 1.2

The AND gate Fig. 1.2A shows a device in some ways resembling the NOT gate. However, it has *two* input lines — shown as X and Y — and one output line — R, for the result. The T line is also connected. There are two switches in this device, operated by the signals on the input lines as before, but with one important difference: in the absence of a pulse on an input line, the switch stays *off;* in the presence of a pulse, it is pushed *on*. In the diagram, the X switch is shown as with a 1 input (ie, a pulse present), the switch therefore being on; the Y line is shown with a

zero input (no pulse), and the switch is therefore off. It will be clear from the

Input		Output
X	Y	R
0	0	0
0	1	0
1	0	0
1	1	1

Table 1.3 AND gate — input-output table

diagram that the timing pulse will reach the output R line only when *both* switches are on; that is, when a pulse (ie, a 1) is present at *both* inputs X and Y. Table 1.3 is the input-output table for this device, which is called an *AND gate,* since it puts out a 1 *only* when 1's are input at both X AND Y.

One further point about AND gates: they can have more than two input lines. Where this is the case, the operating rule is that the gate puts out a 1 when *all* the input lines carry 1's; in all other cases, zero is output. Figs. 1.2B and 1.2C shows the symbols that will be used in this series for two- and three-input AND gates, respectively; notice once again that the T line connections are not shown.

Fig. 1.2

The OR gate The OR gate is shown

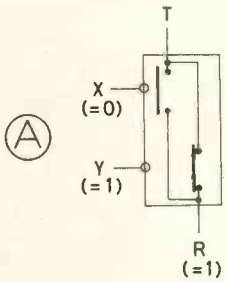


Fig. 1.3

diagrammatically in Fig. 1.3A. The two switches in this device are of the same type as those in the AND gate — that is, they are of the 'push for on' variety, a 1 at the corresponding input closing the switch. In the diagram, the gate is shown with a zero in the X input line and a 1 on the Y line. It will be seen that, if *either* switch is on, the output R line will receive the timing pulse and will therefore output a 1; only if both switches are off — ie, a zero on both input lines — will a zero be output by this device. Table 1.4 is the input-output table for a

Input		Output
X	Y	R
0	0	0
0	1	1
1	0	1
1	1	1

Table 1.4 OR gate — input-output table

two-input OR gate; note that this type of gate, too, can have more than two input lines, and Figs 1.3B and 1.3C show the symbols which will be used in logic network diagrams for a two-input and a three-

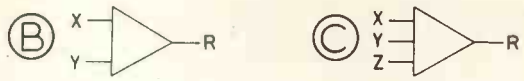


Fig. 1.3

input OR gate, respectively. Compare these with the symbols for the AND gate in Fig 1.2B and 1.2C. The name of this gate reflects its action in putting out a 1 on the R line when a 1 appears on either the X OR the Y input line; where the gate has more than two input lines, the rule is that it will output a 1 when a 1 is present on *any* input line.

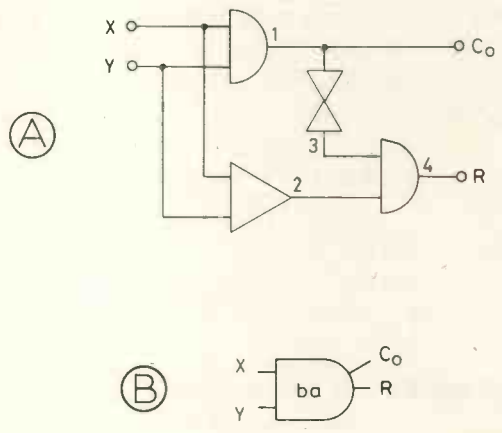


Fig. 1.4

ADDING NETWORKS

" 'Write that down,' the King said to the jury, and the jury eagerly wrote down all three dates on their slates, and then added them up, and reduced the answer to shillings and pence."
LEWIS CARROLL: *Alice in Wonderland*

With the three logic gates I have described (together, it must be understood, with the clock) we can construct our first network. Since binary addition is an operation required by any computer, this is an obvious place to begin. We will start by considering the addition of a pair of bits only — that is, one column of a two-number binary addition. Table 1.1 shows the input-output conditions which would be required of a *full* bit-addition network, with *three* inputs (don't forget the 'carry') and two outputs. But first, we will look at a slightly simpler network called a 'half-adder'.

The half-adder The half-adder differs from the full adder in that it makes no provision for a brought-forward carry. Thus, there are only four possible different input combinations; these are shown in Table 1.5, together with the result, and 'carry out', required for each one.

Fig 1.4A is the diagram of one possible logic network which will give the results required by Table 1.5. The network

Input		Output	
X	Y	Co	R
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Table 1.5 Half-adder — input-output table

includes four gates, which are numbered on the diagram; numbers 1 and 4 are AND gates, 2 is an OR gate, and 3 is a NOT gate. Notice that the input X and Y lines are connected to both gates 1 and 2; the output line from gate 1 provides the 'carry-*out*' signal, and is also led into NOT gate 3; into gate 4 are fed the signals from gates 2 and 3, and its output provides the result, R.

To check the output of any network, it is helpful to have an input-output table with a column for every gate in the network. Table 1.6 is such a table for the half-adder network. The output columns have not been filled in — I leave

Input		Output			
X	Y	1	2	3	4
0	0				
0	1				
1	0				
1	1				

Table 1.6 Half-adder network (Fig 1.4A) — input-output table

this task for the reader's amusement. Here are some hints:

Column 1 is the output of AND gate 1, which has X and Y as input data; look at table 1.3 if you need to, but it should be easy to remember that an AND gate puts out a 1 if, and only if, both its input lines receive 1's.

Column 2 is the output of OR gate 2, which also has X and Y as its input data; look at table 1.4 if you need to — you will find that an OR gate puts out a 1 if either (or both) of its input lines receives a 1.

Column 3 is the output of NOT gate 3; this gate receives as input the output from *gate 1* (entered in column 1); since it is an inverter, it simply converts 0 into 1, and vice-versa.

Column 4 is the output signal from AND gate 4, which receives in the signals from *gates 2 and 3* (entered in columns 2 and 3).

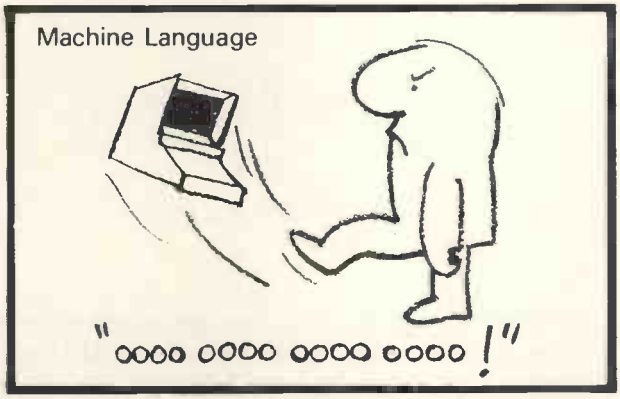
When you have completed all columns of the table, you can check whether your results are correct. You can see from Fig 1.4A that the 'carry' result is taken from gate 1 — so column 1 of table 1.6 should have exactly the same entries as the Co column of table 1.5. The sum, R, is taken from gate 4, so column 4 of Table 1.6 should contain the same entries as the R column of Table 1.5 If these columns agree, you have proved that the network will have the desired effect: if they do not, you've made a mistake somewhere.

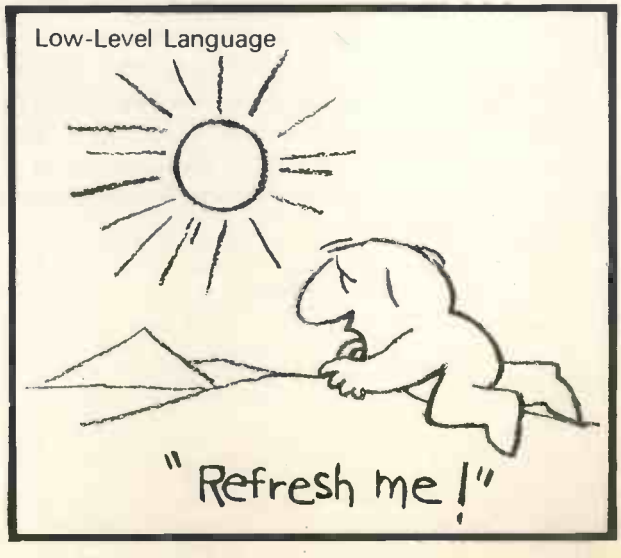
One point which should be noted about this circuit is that the addition is performed with a single timing-pulse — the appearance of the results on the output lines can be regarded, for the time being at least, as simultaneous with the arrival of the input signals. The significance of this fact will become apparent when we go on to consider control circuits.

NEXT ISSUE

In part 2, I will be describing the full bit adder — that is, providing for a *carried-forward* bit — and will go on to show how two complete binary numbers can be added together using one or many full bit adders. You may care, in the meantime, to try your hand at the design of a full bit adder network, using the input-output table given in Table 1.1 — the network I shall describe will use two half-adders and an OR gate, but many other combinations are possible.

PCW In tandem with each of Michael Whitney's articles, beginning from Vol 1 No.2, there will be a practical project. **PCW**





THE GINGERBREAD MAN'S COMPUTER

Colin Chatfield

The birth of any new concept is an exciting event, and even more so when you have been able to be a part of it. Just a year ago I sat at the keyboard of a typewriter which was linked to a computer little larger than a lunch box and realised immediately that there was a future in micro-computing for me.

As I run a business for a charity, arranging holidays for one-parent families, anything that could increase efficiency was of interest to me. But I had to convince the directors that £2000 spent now could enable us to continue operating with less staff, forced on us through lack of funds. They agreed, if I could raise the money.

Whilst trying to do this I investigated what hardware was available and was pleasantly surprised to find quite a large range. In fact the choice was bewildering, confusing and perplexing! All of it came from the USA and three systems were available through UK dealers. Unfortunately the information was not exactly in lay English so I spent three months sorting it out. In the meantime the money was found and I looked into printout devices. These are most important and difficult to find in good condition at an economic price. The solution is now only a little easier but by taking a risk I found a unit which needed conversion for £300 which would suit our requirements. Nobody wanted or was able to do the job though; and without a hard copy, the system, we had by now ordered, was virtually useless.

The units of the system were self-selected as nobody knew what to advise me to buy. In order to save money I decided to build the units myself from a kit, although the 4500 plus solder joints were a bit daunting. However, I had a go and despite one or two small solder bridges, which could have destroyed some of the integrated circuits, the system worked.

I had decided on a central computer unit with 20 thousand words of working memory and a disc memory system capable of storing programs and data records. The disc unit came completely built and tested. Still having no working printer, I acquired a visual display terminal which I find invaluable as it is connected to a standard TV and I can develop programs silently and without wasting paper and use it for data input. Now we have the teletypewriter working and the full system is in use.

Our first task was the recording of the issue of ¼ million raffle tickets to 400 Groups in our sister organisation, Gingerbread, and a stock record system for the articles we sell. The next project was the recording of over 900 bookings for our main holiday of the season for 4500 people which involves the receipt of deposits, the issue of an invoice and subsequent receipt of final payments, automatic searches or listings of the data made alphabetically. New work is being found regularly for the 'GHOST' (as we call it) which stands for Gingerbread Holidays Own Systems Terminal.

Despite the fact that we over spent our budget by £800, because of the problem associated with the teletypewriter and are still looking for funds to clear the account, we are very pleased that for the equivalent of my own salary for a year we have the capabilities of a larger staff.

The programming language used is BASIC and I was fortunate in having a good knowledge of it and only had to overcome the variations in the dialect used. Now we are building up a library of useful programs of commercial interest not only to ourselves but to others. As we could see the problems involved for the lay businessman our user service, set up under the name Micro-aid, may be of interest to readers. A number of young people

interested in programming have got together to help solve specific problems in BASIC for small commercial users and others. With a few contacts in the industry this service can advise potential users, and assist them to put a micro computer system into use. We know the problems and have overcome them and are happy to help others.

The System we have is:

Processor Board based on Motorola 6800 chip.

20K of Random Access Memory with an access time of 500nS.

3 Serial interface boards to:-

1) 8" Floppy disc memory unit capable of storing 315K bytes of memory with sequential and random access.

2) Visual display terminal with typewriter keyboard for input of information which is connected to a 20" UHF TV.

3) Ultronic standard typewriter with paper tape reader and punch converted for us by Micro-aid.

The hardware was supplied by Computer Workshop Ltd., 174 Ifield Rd, London SW10 9AG

Total cost £2800.

If you are interested in the subject of this article you can contact Colin Chatfield and Micro-aid by writing direct to the addresses below:

Colin Chatfield,
Secretary,
Gingerbread Holidays,
Lloyds Bank Chambers,
Camborne,
Cornwall.

Micro-aid,
Lloyds Bank Chambers,
Camborne,
Cornwall.

V. S. Nicola and G. Percival

FLOWCHART YOUR WAY OUT OF TROUBLE

We often find ourselves in a position where we want to solve a problem but are unable to because we don't know how. If we are really intent on solving it, we may go to a text book, or a manual on the subject, and study the method of how to solve our problem. Or, we might ask someone — and nowadays, something — else how to do it. But every time a computer is 'asked' to solve a problem the computer has to be 'taught' how to do it — strange as this may seem to the inexperienced.

This teaching involves giving the computer a sequence of commands or instructions (very like a recipe to a cook), and the data it is to work on, and directing the computer to the beginning of the 'recipe'. The computer then starts to follow the instructions faithfully and, if they are correct, and the data is not erroneous, the solution will be produced. How useful the solution will be is a different matter. The sequence of commands or instructions, which the computer follows, is called a program.

Since the only things the computer knows about the problem that we have presented it with is the program (or recipe), and the data (or ingredients), we must cover every eventuality that might arise during the processing of the data to a solution by the program. For example, if the program is designed to find the real square root of a number, the program must make sure the computer checks the sign of the numbers it is to work on. This is because the square root of a negative number cannot be expressed in real terms, and any result other than an error message would be incorrect. If the program did not tell the computer to check for a negative number, *and* go to an error routine if it found one, then if the number was fed in as data, the computer would attempt to process it, and blindly produce an erroneous answer. Thus the program must test for all possibilities, or errors will occur — errors that may be labelled by the layman as 'computer errors', but which are in fact program (and since the programmer is human — human) errors.

From the preceding discussion, we can see that if we expect the computer to give a good, accurate and reliable result, we must load it with a precise program which will be able to cope with any possible data (including obvious erroneous data). To do this, we must, therefore, be able to formulate a strategy for solving the problem precisely, then translate the strategy into a program. The strategy which guarantees either a solution (correct), or indicates an insoluble problem, in a finite number of steps, is known as an algorithm.

The algorithm, then, is like a recipe, in that it is a list of steps to be followed to find the result. But though an algorithm is what we want to produce to tell the computer what to do, we still don't know how to find it. The tool we use to express the algorithm in easy-to-understand terms, is the *flow chart*.

The flow chart is a very powerful aid, as it expresses information flow of a time-dependent nature visually. When we write down formulae, or a list of instructions, we don't easily interpret them as a set of discrete, time dependent steps, but rather see them as a whole. The computer, being a time-state machine, only 'sees' each instruction as it executes

it, so that to devise an algorithm for a computer, we need an aid which will represent the information flow in a time-dependent manner. The flow chart does this by displaying each logical time-state separately and in graphical form which the brain finds easy to interpret. I said 'logical' time state, as the flow chart does not normally express each machine instruction, but rather each logical step.

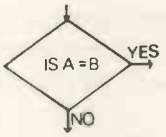
The word flow chart, in fact means a chart (ie graph) which gives the 'flow' of information, and so is not peculiar to the computer industry, but is used by a wide range of people who want to represent time-dependent operations such as chemical production, or mechanical assembly, to name but a couple of its uses. The reason it is so much used in the computer field is because computers are designed to handle vast quantities of information processing, and they do this in a time-dependent manner.

A flow chart then, is a set of symbols representing operations on information, which are connected by arrows to indicate the *path* the information takes through the flowchart to produce the result. As used to describe algorithms for computers, flowcharts use a variety of symbols to represent different types of operation such as input, output, decision, or assignment (ie arithmetic or logical operations). These symbols may vary, but a representative set is described below:—

The beginning, or entry point to the algorithm.

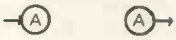
The end of the algorithm.

A step to be performed. May be any arithmetic or logical process, except decision.

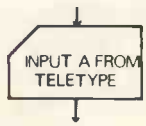


Decision. If the answer to the question in the box is yes, the yes exit is taken; otherwise take the no exit.

Connections; Where two or more lines merge.

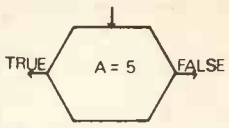


Connection. When it is not possible to continue a flowchart immediately after the preceding box, a labelled circle may be used.



INPUT

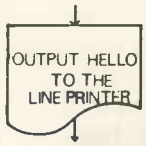
The box will be labelled with the source, and possibly a variable to read it into.



DECISION

This box is sometimes used in place of the diamond shaped box. It generally contains a proposition which is either true or false.

Circles are also used to indicate the entry to, and exit from, a subroutine.



OUTPUT

Output data to an output device.

SUBROUTINE

A subroutine is a common step, which is described in full in another part of the flowchart. The box contains the subroutine name.

This box is used to indicate the values passed to the subroutine, and those returned. If no values are changed by the subroutine, the box is not needed.

Now we have defined the symbols, it is time to see how they are used. We shall therefore take a problem we wish to solve, and produce a flowchart for it. The problem we have chosen is taken from the 'Basic Pontoon' article, and is to produce an algorithm to fill an 'array' of 52 cells with the numbers 1 to 13, selected at random, but no more than 4 of each number. The flowchart is reproduced and works as follows:—

First, we need a 'pointer' into the array, so that we know which cell we are looking at. This pointer will be the number of the cell we want to look at. We shall use as the pointer variable 'A', and set it to 52, so that it will point to the top of the array. We then pick a number at random, in the range of 1 to 13, but before we put it into the cell, we must check to make sure that there are not four similar numbers *already* in the cell 'A'.

You may think that this can't be so, as this is the

first number we are putting into the array, but we will be 'looping' to reuse that section of code again, and it is easier to use a *common* set of code, than make a special case of the first four numbers. This is an important principle in programming, as it makes repetitive procedures very easy; it is only necessary to use some form of loop counter' to detect when the code inside the loop has been executed the required number of times, and to return control to the start of the procedure at the end of each pass, until that condition is met. Here, we will use the variable A as both a pointer into the next cell, and as a loop counter, which will count A *down* on each pass, and *exit* when it reaches 1, the last cell.

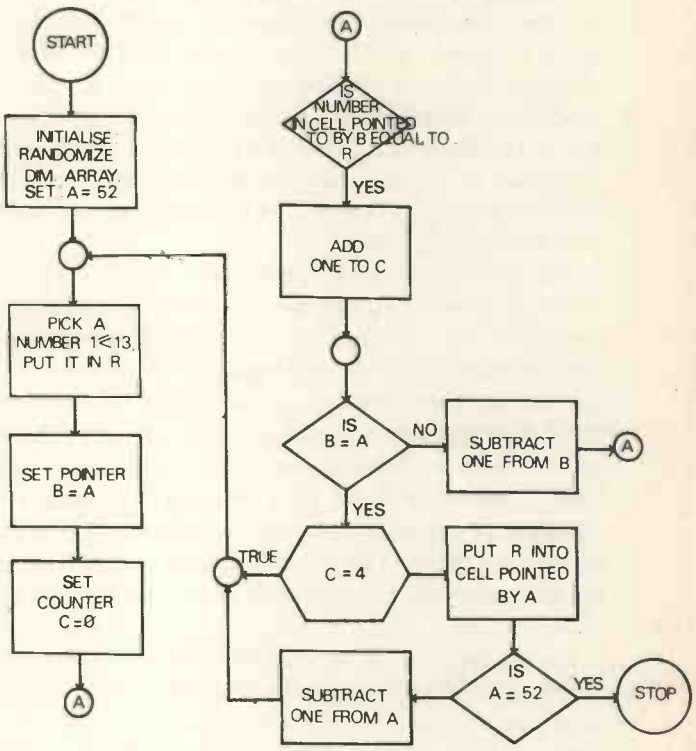
If it didn't matter how many of each number 'R' were the same, we could simply put 'R' in the cell pointed to by 'A', decrement (subtract one from) A, test for 'loop complete' (A=1) and jump back if not. However, we need to ensure that there are no more than four (actually, exactly four, as four times thirteen equals fifty-two) of the same number in the array. To do this, we will need to search the array, to ensure we haven't already got four of the number 'R'. We do this by using the *second* pointer, 'B', to point to the cells we wish to test, and start it off at the first cell — in this case, No 52. We also reset a counter, C, to zero. As you can see from the flowchart, we then read the number in cell 52 (first pass, it will be undefined), and test if it is equal to 'R'. If it is, we add one to 'C', then look to see if we have searched all the cells filled so far. If not, we don't add one to 'C', but go straight to see if we have searched all the cells. We know if we have done this, because A and B will then be the *same,* and we can continue on to test 'C'. If not, though, we decrement B, and loop back to read *another* cell. When we have tested *all* the cells, we check if C=4; if it does, we can't use R, and so go back for another 'R'; if not, though, R is useable, and we put it into the cell pointed to by 'A', before testing if A is one (loop finished), and if not, decrementing 'A' before looping back.

All this may sound complicated, but in fact it turns out to be quite simple when programmed. Using the high level language 'BASIC', the looping logic can be condensed to two statements. For example, the outer loop, counting A down from 52 to 1, becomes 'FOR A=52 TO 1 STEP — 1", which sets 'A' to its initial value 52 then counts it down −1 (the step value) until it reaches 1 (the 'to' value). The actual decrementing occurs when a 'NEXT A' statement is found, which tests if A is one, before performing the decrement, and returning to the statement following the 'FOR ———', unless A is one, in which case control falls through to the statement following the next.

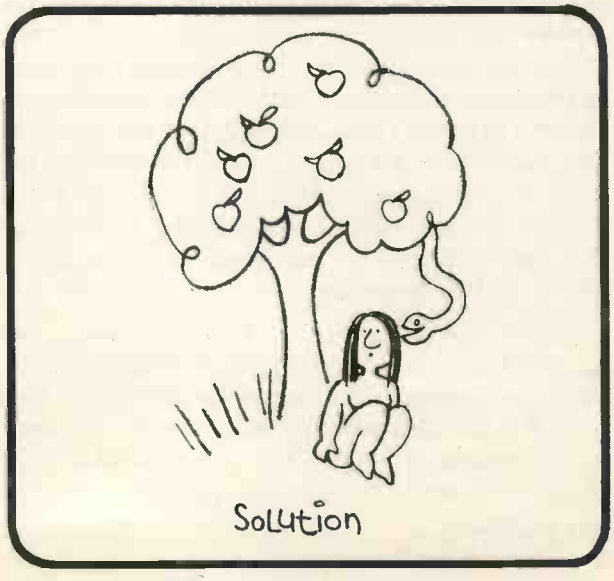
All high level languages have provision for such loops (often called 'do loops'), but if you are programming in machine code, you would probably have to use the expanded version given in the flowchart.

As you can see, the flowchart explains the program much better than words, and during development is the easiest way to put ideas onto paper. Of course, there is no need, when developing the flowchart, to use standard symbols. It is easier to develop a kind of 'shorthand', as you will probably be redrawing the thing many times before it is finalised. However, when it is finished, the use of standard symbols makes it easily transferable to any other interested person — and that is the other main function of the flowchart — explaining in universal terms, your program to others.

Some programmers look down upon flowcharts, taking a kind of perverse pride in forcing their minds to jump around burdened with mundane detail. But of course, any program which is not clearly documented exists in a vacuum; and like any living



thing in a vacuum, is as good as dead. It is worth our while to accept the discipline and help of this programming aid which makes it possible for us to flowchart our way out of trouble.



Basic Pontoon

Guilbert Percival

The last few years has seen the introduction of computer hardware, in the form of microprocessors, at a price cheap enough for the hobbyist to buy for use at home. However, the mystique which has surrounded computers since their invention makes their operation and use the province of only a chosen few — the programmers. The aim of this article is to remove some of the mystery from the programming of a computer, by following the development of a program to enable the computer to play the card game pontoon. While the complete program listing will be given to enable anyone with a microprocessor capable of accepting a program in standard Basic to enter and run the program, emphasis in the text will be more on the process involved in going from an idea to a working program, than the program itself.

Just as in driving a car, certain skills must be learnt before programming can become easy or enjoyable. The difficulty is that computer programming is remote from life, while cars have become so much a part of our lives, that we have already learnt such basic skills as judging speed and distance long before we try to drive. Although we may not realise it, we have similarly learnt the skills necessary to flowchart a program when performing such tasks as learning to play a new game. The only difference is that we did not formalise the process by giving it such titles as 'flowcharting'.

The computer is a machine which executes instructions given to it by the programmer, which it does literally, acting on the instructions exactly as they are given. Thus the programmer must cover every possibility, and such formalities as flowcharting have been evolved as an aid. This is no different in essence from the case of a game, where the rules are set out in clearly defined, unambiguous steps, so that there can be no disagreement from the players. It is why I have chosen a game to illustrate the procedures I use when programming. The steps of the game have already been defined, and it merely remains to convert them to a language the computer can understand. Of course, when we examine the steps for input to the computer, we may find that they are not as unambiguous or logical as we would like, but this can be corrected with the formal methods of the flowchart.

When performing a familiar task, we unconsciously split it up into three main sections. When confronted with an unfamiliar problem, it pays to look at the steps we used on the familiar problem, then try to apply them to the unfamiliar one. I will therefore formally define the steps I would use on any problem:-

1) Understand the problem.

This may seem obvious; in the context of the game pontoon we may take that to mean 'define the rules'. However, as we intend to play the game with a computer, we must define the rules to that end. We must therefore define the inputs we expect to feed into the computer, and the outputs we would like it to generate as a result.

2) Devise a plan.

Once we have defined the inputs, and hence have an idea of the data representation, we can devise a plan to give us the outputs. We already have the rules, so if we can break them up into their simplest steps, and define them in terms of the data representation in the computer we should be able to input those steps as program instructions to the computer. By writing out the steps in the form of a flowchart, we put them into a convenient form for coding as program instructions.

3) Carry out the plan.

Having defined the problem in steps the machine can use, we may try it in the computer to see if it works. Often it helps, when confronted with a complex problem, to break it into smaller steps, and program each step separately. That way, progress is more obvious, and there is less danger that one erroneous step will upset other correct steps in the program.

There is one more step, which may not be used in familiar problems, and that is *review.*

4) Review the program.

Although the program may work, it may not be very efficient, or there may be 'bugs' which only show up in special cases. Therefore, it should be tested with all possible inputs, and inspected in the light of hindsight, to see if there is a better way to do it.

My solution to the problem.

1) Define the problem.

The problem I have set is to write a 'BASIC' program to play pontoon.

The rules I shall use for the purposes of this program are:-

a. The player and the house (computer) are each dealt two cards. The cards are 2 through 10, jack, queen, king, ace. The suit is unimportant; the cards face value, with jack through king worth 10, and ace worth 1 or 11 at the player's choice.

b To win, the player must exceed the house score, without exceeding 21. To do this he may call for additional cards. When he is satisfied, the house may draw cards until it is satisfied, when the scores are compared and both hands shown. If either player or house exceeds 21, he busts, and loses.

c There are several special cases. If the player gets two aces first off, he has the option to 'split',

drawing further cards to each ace and forming two hands, of which he takes the best. If either player or house gets 21 in two cards, it is called pontoon, and the cards are shuffled. Normally, that player would take the bank, but to keep the program within 2K bytes, I did not implement this. Pontoon is the highest hand, followed by 5 cards under 21 (5 card trick), then 21 in more than two cards, and so on down.

The program must therefore keep some representation of cards, selecting them by a random number process, and be able to display their values on a VDU. It must assign these 'cards' to either a player, or a house total, being able to recognise aces, so as to assign 1 or 11, whichever is more appropriate, and be capable of recognising two aces for a split. It must give the player the option to twist or pass, in which case it must check its own score to see if it wants more cards. Lastly, it must check to see who won, type out the score, test if pontoon, then either shuffle the 'cards', or put the discards on the bottom of the pack, and restart for the next round.

2) Devise a plan.

Having defined the problem in words, I proceed to draw a flowchart. This is not in fact difficult.

I proceed as follows:-

a. Draw a box labelled 'start'. This might seem trivial, but it gets things going.

b. Look at the rules. What should I do first? Well, if I was playing with cards, I would shuffle them. So I must decide how I will represent the cards, then write a routine to shuffle them.

As I have 13 different cards, I can represent them with the numbers from 1 to 13. I can generate them as I need them, using the 'RND' statement, but I could not be sure that I would not get more than 4 of each card, when playing enough rounds to use all the pack. Hence, I will use an array which I will dimension to 52 elements, and fill with random numbers from 1 to 13, with exactly four of each number. As it can be written separately, I will write the pack filling routine first, testing and debugging it before continuing with the program. Its flowchart is given elsewhere as an example in an article specifically about flowcharting by Victor Nicola and myself.

Now, I have got to the start of my rules. I therefore draw another block in my flowchart saying 'initialise', by which I mean shuffle the pack and set any counters to their starting values. I then consult the rules. They say 'deal two cards each to the player, and the house in turn'. I therefore draw four boxes on my flowchart labelled 'deal player', 'deal house', 'deal player', 'deal house'. Then back to the rules. These state that the player may call for additional cards, and that if he has two aces, he may split. So let's draw a decision box, and label it 'has player two aces?'. If not, we must ask the player if pass or twist, so from the 'no' output of the 'two aces' box, we draw another decision box labelled 'pass or twist?'. Similarly, we follow every possibility, filling in our flowchart until we have no dead ends. It can be tedious, but it's not difficult. Depending on the computer language you are writing in, you can fit more or less in each 'box', but if in doubt, it is best to give each branch of control its own 'decision box'. Self-contained blocks with only one input and output need only one box, especially if these will be implemented as subroutines — in which case, the subroutine need only be detailed once. An example of the latter is the box labelled 'get player/house card'. This turned out to be quite a complex routine, but while it contains many 'conditional if' statements, it is self-contained so that though its first occurrence in the flow may be fully documented, its later occurrence is contained in one box.

Once the flowchart is complete, the program may be coded by simply writing out the lines of code needed to implement each box, then crossing off that box on the flowchart. The 'BASIC' statement numbers may be put next to the input or output to each box as it is completed, so that looping back can be done easily. Looping forward can only be done by taking a guess at the statement number you are going to, using that number, and ensuring that when you get there, the right statement is inserted. To assist in this, it is common practice to number 'BASIC' statements in increments of ten. Then additional statements can be added easily. When the program is complete and running, the 'renumber' command can be used to tidy up the line numbers.

As you write out the code, you may find yourself using similar sets of code in different parts of the program. If this occurs, it is possible a subroutine would be more efficient. In 'BASIC', a 'GOSUB' statement does not pass any parameters into the subroutine, which can only use the same variables each time it is called. Thus a section of program which performs a common operation on different variables at different places in a program, cannot normally use a subroutine. For example, in the case of the 'get a card, add it to the totals, type its identity, and test if it's an ace' routines, which are essentially the same for player and house, the variables into which the totals are entered are different. Where the routine is long, and the variables few, it is possible to swap the variables before entry to the subroutine. However, in this case, a much better alternative was possible. The variables were made into an array (X), which could be accessed by a subscript. It was then only necessary to use one variable (C), plus appropriate offsets in the sub-routine, to make 'C' a 'switch', changing all references from player to house as required.

There are other similar tricks, which unfortunately only come with practice, that may be used to reduce the amount of code required. The best advice I can give is 'if in doubt, try it'. The worst that can happen is that it doesn't work, and at best it may save a lot of code, and hence memory space. As another example, I originally had the pack shuffling routine stepping 'A' from 1 to 52. I then used 'A' as a variable, which needed to be initialised at 1. I was

able to save one line of code by rearranging the pack shuffling routine to step from 52 back down to 1. Thus, at the exit from the pack shuffling routine I had 'A' already at 1. This may seem trivial, but if you have a microprocessor with limited memory, it could be important.

3) Carry out the plan

Having successfuly entered the program, it must be debugged. This means not only making sure you don't get any error messages, but also trying it with every possible set of inputs to make sure it has no hidden bugs. For example, the program as I first coded it, did not recognise pontoon, or a 5 card trick in a split, as I had forgotten to save the 'cards dealt total' in X5(C). This and several other difficulties was resolved as part of step 4, *review*.

4) Review the plan.

Although the program worked well, it had a few difficulties, such as that outlined above when splitting. Also, it slightly exceeded my target of 2K bytes. By looking at the program as a whole, it was possible to rearrange it to cut down storage requirements, and correct several ills. One of the corrections came as a compromise; the ability to print out the value of all the house cards, including the first two. Initially, the first two cards were dealt alternately, but with the house cards not displayed to the player. However, this made it difficult to display them later when the house showed its hand, without special logic to go back in the pack to pick out the cards. This difficulty was solved, and the code simplified, by taking all the player's cards first, then the house's. This method is not strictly correct, but as random numbers are being used, should make no practical difference — except when a player busts, in which case, the house takes no cards, instead of the two it should take.

In conclusion, the program listing, a separate 'remarks' listing (kept separate to minimise storage), and a flowchart is provided to help readers enter and modify the program on their own system and to their own tastes. There are many avenues for expansion for those with enough memory, as the program given does not allow betting, or multiple players. Also, the bank could be given to a player after pontoon, by changing the advantage, and order of dealing.

In order to make the program as universal as possible, I have not used any advanced 'BASIC' statements, such as multiple 'if . . . then if . . . then . . . (etc)' to ensure that the program would run 'as is' on any 'BASIC' interpreter. However, it could be shortened by their use, which I leave to the more experienced reader. Similarly, I have not spent time discussing the 'BASIC' statements I have used, as there are many excellent texts available from any public library, suitable for the beginner.

As the program was designed to run in 2K bytes of 'scratch' area in a microprocessor based system, some compromises were necessary. Even so, minor difficulties may be experienced. The most likely problems on a 'micro basic' interpreter are:-

1) No 'for next' loop.

In this case, the 'let', and 'if . . . then' statements may be used. For example, in the shuffling routine:-

60 for A=52 to 1 step −1 becomes 60 let A=53
(note one more for negative step)
65 let A=A−1
(or + 1 for positive step)

70 . . . 70 . . .
etc . . .
150 next A 150 if A <> then go to 65

Note that one more statement is needed, and also that the initial value is one less (one more in magnitude for negative steps)

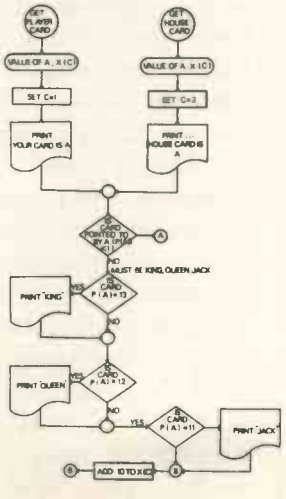
2) Integer only arithmetic. The only problem here is the 'RND' statement. This should be modified to produce at line 70, a random number between 1 and 13 inclusive. At line 540, a multiplier may be needed on the right hand side of the inequality. As given, the random number is in the range 0 to 1 with 6 significant figures.

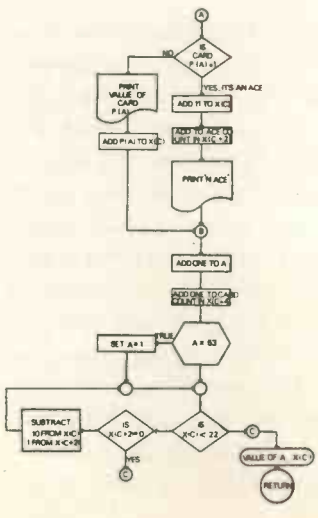
3) Some interpreters expect an 'end' statement as the highest numbered statement. If necessary, this may be added as '1100 end'.

Suggested books on 'BASIC' programming. See the mathematics or language section, under 510.7834.

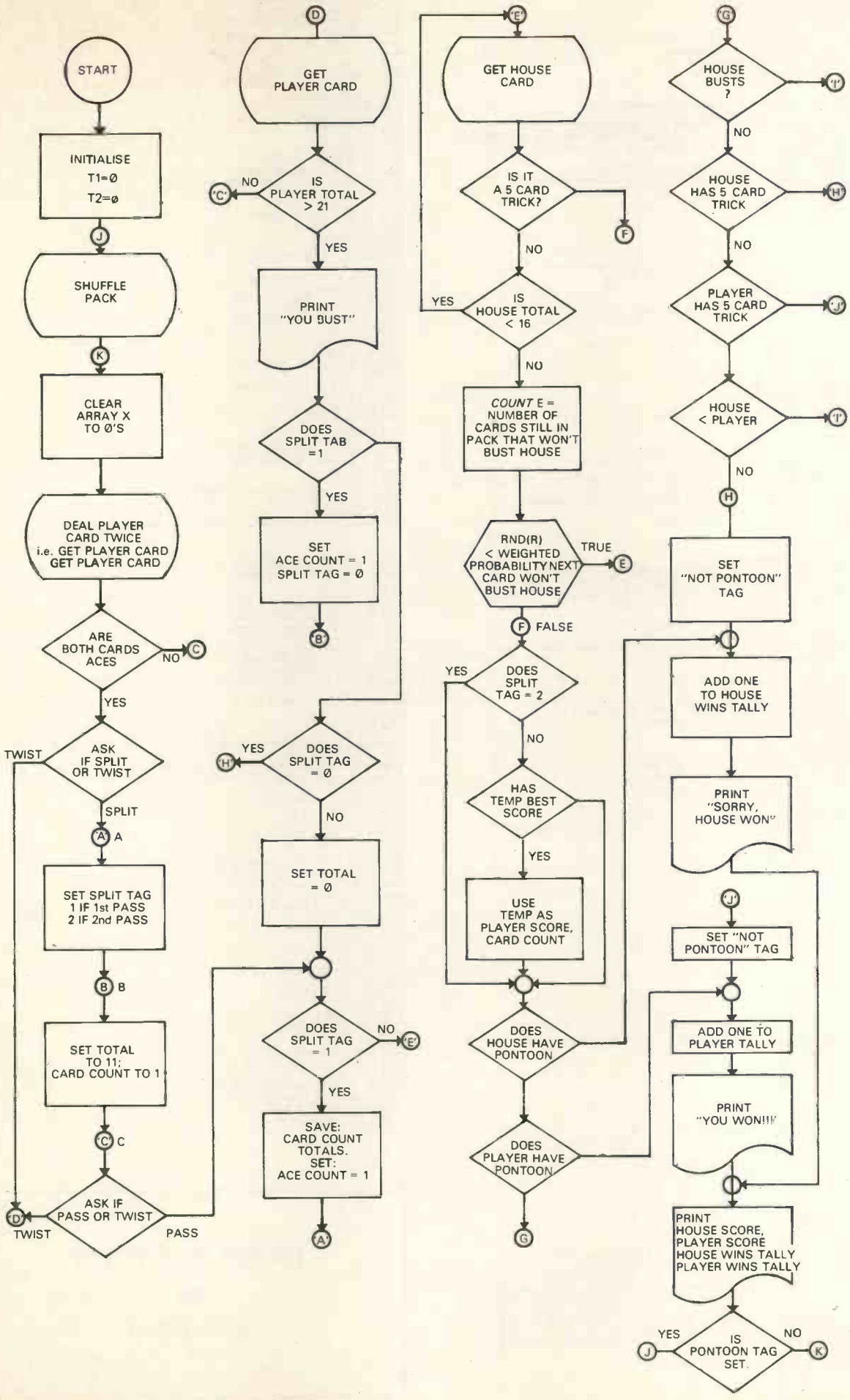
1) 'Interactive Computing with BASIC' by Donald M. Monro.

2) 'Elementary BASIC with Applications' by Mario V. Farina.





FLOWCHART FOR "PONTOON"



```
0001 REM THIS IS A PROGRAMME TO PLAY PONTOON.
0002 REM ARRAY X IS USED FOR:-
0003 REM X[2] IS HOUSE TOTAL
0004 REM X[3] IS PLAYER ACE COUNT
0005 REM X[4] IS HOUSE ACE COUNT
0006 REM X[5] IS PLAYER CARD COUNT
0007 REM X[6] IS HOUSE CARD COUNT
0008 REM X[7] IS PLAYER SPLIT TAG.
0009 REM F IS TEMP FOR X[1] ; G IS TEMP FOR X[5], IF SPLIT
0010 REM DIMINSION PACK P, TALLY COUNTER X
0020 REM T1 WILL BE PLAYER WON GAMES
0030 REM T2 WILL BE HOUSE WON GAMES
0040 REM
0050 REM
0060 REM SET UP LOOP TO FILL ARRAY P WITH RANDOM NUMBERS
0070 REM GET RANDOM NUMBER BETWEEN 1 AND 13
0080 REM C IS USED TO COUNT NO. OF SAME FACE VALUE CARDS
0090 REM SET UP LOOP TO SEARCH PACK TO ENSURE NO MORE THAN 4 OF EACH CARD
0100 REM IF RAND. NO. <> CARD IN P[B], DONT ADD TO COUNTER C
0110 REM ADD ONE TO C
0120 REM LOOP TILL PACK CHECKED (AS FAR AS IT IS FULL)
0130 REM IF 4 OF CARD R, TRY ANOTHER R TILL ONE IS FOUND WHERE C<>4
0140 REM PUT R IN PACK
0150 REM LOOP TILL PACK FULL. A IS NOW =1
0160 REM CLEAR ALL ARRAY X
0170 REM
0180 REM AT END, B=7
0190 REM GET PLAYER CARD
0200 REM GET PLAYER CARD
0210 REM X[3] IS PLAYER  ACE COUNT. IF NO ACES, GO ASK IF PASS OR TWIST
0220 REM X[1] IS PLAYER SCORE. WILL BE 12 IF 2 ACES.
0230 REM 2 ACES. ASK IF SPLIT.
0240 REM
0250 REM TWIST? THEN 320 TO GET A PLAYER CARD
0260 REM X[7] IS SPLIT TAG. IF 1, THEN 1ST PASS; IF 2, THEN 2ND.
0270 REM SET X[1] TO TOTAL FOR 1 ACE
0280 REM SET X[5] CARD TOTAL TO 1
0290 REM ASK IF PASS OR TWIST
0300 REM
0310 REM IF PASS, THEN 400
0320 REM TWIST: GET PLAYER CARD
0330 REM TEST IF PLAYER BUST
0340 REM
0350 REM IF NOT 1ST, PASS OF SPLIT,THEN 380
0360 REM SET PLAYER ACE COUNT TO 1
0370 REM RESET SPLIT TAG
0380 REM GO LOOP FOR 2ND PASS
0390 REM IF NO SPLIT, THEN PLAYER BUST; GO TO PLAYER LOST.
0400 REM ZERO PLAYER TOTAL
0410 REM EITHER PASS, OR BUST ON 2ND PASS OF SPLIT. IF LATTER, THEN HOUSE TURN
0420 REM F IS TEMP, USED TO STORE 1ST PASS (SPLIT) SCORE.
0430 REM G IS TEMP, USED TO STORE 1ST PASS (SPLIT) CARD NUMBERTOTAL.
0440 REM SET ACE COUNT 1 FOR 2ND PASS
0450 REM GO SET SPLIT TAG = 2, DO 2ND PASS

0460 REM GET (MORE) HOUSE CARD(S).
0470 REM X[6] IS NO. OF HOUSE CARDS. IF 5 CARD TRICK, THEN SIT.
0480 REM X[2] IS HOUSE SCORE. IF <16, THEN LOOP FOR ANOTHER CARD
0490 REM E WILL BE NO. OF CARDS STILL IN PACK THAT WOULD NOT BUST HOUSE
0500 REM SET UP LOOP TO COUNT CARDS STILL IN PACK.
0510 REM TEST IF CARD WOULD BUST HOUSE
0520 REM WOULDNT BUST HOUSE, SO ADD TO TOTAL IN E
0530 REM LOOP TILL ALL CHECKED
0540 REM COMPARE RAND. NO. TO WEIGHTED PROBABILITY WANTED CARD IS NEXT
0550 REM HOUSE FINISHED:- TEST IF SPLIT
0560 REM SPLIT, TO FIND BEST HAND
0570 REM 2ND HAND IN F WAS BEST; PUT IN X[1]
0580 REM ALSO PUT G IS X[5].
0590 REM HOUSE NOT 21? CANT BE PONTOON
0600 REM HOUSE 21; PONTOON IF 2 CARDS - THEN HOUSE WON
0610 REM PLAYER 21? CANT BE PLAYER PONTOON
0620 REM PLAYER 21; WON IF 2 CARDS
0630 REM HOUSE BUST
0640 REM HOUSE GOT 5 CARD TRICK? HOUSE WON
0650 REM PLAYER GOT 5 CARD TRICK? PLAYER WON
0660 REM PLAYER MORE THAN HOUSE? PLAYER WON
0670 REM SET NON-PONTOON TAG
0680 REM HOUSE WON. ENTER HERE IF PONTOON
0690 REM BUMP HOUSE GAMES WON
0700 REM GO PRINT SCORES
0710 REM SET NON PONTOON TAG; ENTER HERE IF PLAYER WON, NOT PONTOON
0720 REM ENTER HERE PLAYER WON, PONTOON
0730 REM BUMP PLAYER SCORE
0740 REM PRINT VALUE OF HANDS
0750 REM PRINT SCORES SO FAR
0760 REM
0770 REM IF NOT PONTOON, LOOP WITH SAME PACK
0780 REM PONTOON; SHUFFLE PACK
0790 REM ENTRY OF 'GET PLAYER CARD' ROUTINE
0800 REM
0810 REM
0820 REM ENTRY OF 'GET HOUSE CARD' ROUTINE
0830 REM
0840 REM IF NOT JACK, QUEEN, KING, THEN 930
0850 REM TEST IF KING
0860 REM
0870 REM TEST IF QUEEN
0880 REM
0890 REM TEST IF JACK
0900 REM
0910 REM ROYALTY: ADD 10 TO TOTAL
0920 REM NOW SKIP ACE/NUMBER CARD ROUTINE
0930 REM TEST IF ACE
0940 REM ADD 11 TO TOTAL
0950 REM BUMP ACE COUNT
0960 REM
0970 REM SKIP NUMBER CARD ROUTINE
0980 REM
0990 REM ADD VALUE OF CARD TO TOTAL

1000 REM BUMP A TO NEXT CARD
1010 REM BUMP CARD COUNT
1020 REM TEST IF PACK EMPTY
1030 REM YES; START AT BEGINNING OF PACK
1040 REM IF TOTAL NOT BUST, THEN RETURN
1050 REM TOTAL BUST; IF NO ACES, RETURN (BAD LUCK)
1060 REM ACES IN HAND, SO TAKE 10 FROM TOTAL
1070 REM TAKE ONE FROM ACE COUNT
1080 REM LOOP BACK IN CASE STILL BUST(PLAYER GOING FOR 5 CARD TRICK)
1090 REM                    (WITH LOTS OF ACES.          )
1100 REM LINE 540 DECIDES IF TWIST OR PASS. THOUGH IT USES THE PROBABILITY
1110 REM THAT THE NEXT CARD WILL IMPROVE IT'S TOTAL, IT IS NOT CHEATING
1120 REM ANY MORE THAN A PLAYER, HAVING KNOWLEDGE OF THE CARDS ALREADY DEALT
1130 REM - AND HENCE THOSE STILL IN THE PACK IS CHEATING. THE ACE COUNT
1140 REM X[4] IS USED TO INCREASE THE PROBABILITY OF TWISTING IF THERE ARE
1150 REM ANY ACES COUNTING AS 11 STILL IN THE TOTAL; IE THE HOUSE GOES FOR
1160 REM A 5 CARD TRICK. THE -.2 IN THE DIVISOR ENSURES THAT THE HOUSE WILL
1170 REM NOT TWIST ON LOW PROBABILITIES.
1180 REM WRITTEN ON 12/12/77 BY G. PERCIVAL.
```

```
0010 DIM P[52],X[7]
0020 LET T1=0
0030 LET T2=0
0040 RANDOMIZE
0050 PRINT "SHUFFLING PACK......"
0060 FOR A=52 TO 1 STEP -1
0070   LET R=INT((RND(R)*13)+1)
0080   LET C=0
0090   FOR B=A TO 52
0100     IF P[B]<>R THEN GOTO 0120
0110     LET C=C+1
0120   NEXT B
0130   IF C=4 THEN GOTO 0070
0140   LET P[A]=R
0150 NEXT A
0160 FOR B=1 TO 7
0170   LET X[B]=0
0180 NEXT B
0190 GOSUB 0790
0200 GOSUB 0790
0210 IF X[3]=0 THEN GOTO 0290
0220 IF X[1]<>12 THEN GOTO 0290
0230 PRINT "DO YOU WISH TO (1):- SPLIT; (2):- TWIST ";
0240 INPUT M
0250 IF M<>1 THEN GOTO 0320
0260 LET X[7]=X[7]+1
0270 LET X[1]=11
0280 LET X[5]=1
0290 PRINT "DO YOU WISH TO (1):- PASS; (2):- TWIST ";
0300 INPUT M
0310 IF M<>2 THEN GOTO 0410
0320 GOSUB 0790
0330 IF X[1]<22 THEN GOTO 0290
0340 PRINT "YOU BUST!"
0350 IF X[7]<>1 THEN GOTO 0390
0360 LET X[3]=1
0370 LET X[7]=0
0380 GOTO 0270
0390 IF X[7]=0 THEN GOTO 0670
0400 LET X[1]=0
0410 IF X[7]<>1 THEN GOTO 0460
0420 LET F=X[1]
0430 LET G=X[5]
0440 LET X[3]=1
0450 GOTO 0260
0460 GOSUB 0820
0470 IF X[6]=5 THEN GOTO 0550
0480 IF X[2]<16 THEN GOTO 0460
0490 LET E=0
0500 FOR D=A TO 52
0510   IF P[D]>21-X[2] THEN GOTO 0530
0520   LET E=E+1
0530 NEXT D
0540 IF RND(R)<((E*(X[4]*2+1))/(53-A)-.2) THEN GOTO 0460

0550 IF X[7]<>2 THEN GOTO 0590
0560 IF X[1]>F THEN GOTO 0590
0570 LET X[1]=F
0580 LET X[5]=G
0590 IF X[2]<>21 THEN GOTO 0610
0600 IF X[6]=2 THEN GOTO 0680
0610 IF X[1]<>21 THEN GOTO 0630
0620 IF X[5]=2 THEN GOTO 0720
0630 IF X[2]>21 THEN GOTO 0710
0640 IF X[6]=5 THEN GOTO 0670
0650 IF X[5]=5 THEN GOTO 0710
0660 IF X[1]>X[2] THEN GOTO 0710
0670 LET B=1
0680 PRINT "SORRY, HOUSE WON."
0690 LET T2=T2+1
0700 GOTO 0740
0710 LET B=1
0720 PRINT "YOU WON!!!"
0730 LET T1=T1+1
0740 PRINT "YOU GOT";X[1],,"HOUSE GOT";X[2]
0750 PRINT "YOUR WINS =";T1,,"HOUSE WINS =";T2
0760 PRINT
0770 IF B=1 THEN GOTO 0160
0780 GOTO 0840
0790 LET C=1
0800 PRINT "YOUR CARD IS A";
0810 GOTO 0840
0820 LET C=2
0830 PRINT ,,,"        HOUSE CARD IS A";
0840 IF P[A]<11 THEN GOTO 0930
0850 IF P[A]<>13 THEN GOTO 0870
0860 PRINT " KING."
0870 IF P[A]<>12 THEN GOTO 0890
0880 PRINT " QUEEN."
0890 IF P[A]<>11 THEN GOTO 0910
0900 PRINT " JACK."
0910 LET X[C]=X[C]+10
0920 GOTO 1000
0930 IF P[A]<>1 THEN GOTO 0980
0940 LET X[C]=X[C]+11
0950 LET X[C+2]=X[C+2]+1
0960 PRINT "N ACE."
0970 GOTO 1000
0980 PRINT P[A];"."
0990 LET X[C]=X[C]+P[A]
1000 LET A=A+1
1010 LET X[C+4]=X[C+4]+1
1020 IF A<>53 THEN GOTO 1040
1030 LET A=1
1040 IF X[C]<22 THEN GOTO 1090
1050 IF X[C+2]=0 THEN GOTO 1090
1060 LET X[C]=X[C]-10
1070 LET X[C+2]=X[C+2]-1
1080 GOTO 1040

1090 RETURN
```

BASIC PONTOON

by

G. Percival

Missionary Job

John M. Anderson

THE CONVERSION OF IBM 73 SERIES I/O WRITERS

Here are some notes that may be of interest to those intending to use these machines:

The IBM 'Selectric' typewriter was originally developed as an Input/Output device for use with the 360 series of computers. IBM found it possible to eliminate most of the problems caused by the previous methods of parallel transmission, in which any one of 56 (or more) characters was transmitted over as many separate circuits, in each direction! By introducing the two central concepts of 'Single Element' typing and 'Extended' Binary Coded Decimal' coding, the IBM designers were able to reduce the number of parallel circuits to only eighteen.

The IBM 73 Series of I/O writers (which consist of 731 and 735 models with respectively 11" and 15" paper width) were all built around the same mechanical system. We have taken the 735 I/O writer supplied to *Systemation* for use in the **Susie** computer as our example, but the basic mechanism is the same in all the many OEM machines. **Susie** writers are often found on the surplus market at the moment and most of them can easily be identified as they have a large S fixed in the centre of the front cover in place of the IBM sign. The 'Golf Ball' print head has all the capital letters on the Lower case hemisphere and then has 31 fractions on the Upper case hemisphere, which is easy to spot as it faces away from the paper. One point worth noting at this stage about these Susie typewriters is that the Carrier Return function key is not mechanically connected to select mechanism and thus relies on a signal from the controlling system to echo the C/R when it is needed. This can easily be modified by linking the key lever by a suitable wire strap to the latch — look at the mechanism, press the key and you will see what is needed to effect the conversion.

The 88 characters required to be printed are coded as six bits in one of two versions: — EBCD or Correspondence. There are differences between the coding of the printing characters in these two versions but the main difference that concerns us at the moment is that the function switches are wired differently. The following five functions are required in addition to the printing characters:— Carrier Return, Tabulate, Index (Linefeed), Backspace and Shift. Also found on most machines are Keyboard Lock, Red Ribbon Select and Black Ribbon Select. Because these functions take longer to perform than typing a single character, IBM recommend that closed loop operation should be used. This means that to ensure that no further operation is requested until the completion of that already in progress, status information is 'fed back' to the controlling system.

In either EBCD or Correspondence coding, the arrangement of the characters on the spherical type-head is used to implement the selection process. The head is divided into two hemispheres, namely Upper and Lower case. Each case is further divided into four rows and eleven columns. See figure 1 for details of correspondence coding used on Systemation head. The so called Home Column (indicated) is normally presented to the printing position of the next place in the line. The head, when at rest, also has the so called Zero tilt row in the printing position. By coding bits T1 and T2 as binary selections, four positions are available ie zero, one, two or three steps of tilt. By coding R1, R2 and R2A in 1-2-2 extended binary code, five positions are available ie one, two, three, four or five steps of rotation. By coding R5 as minus five steps of rotation (spring loaded to the centre) positive and negative rotations as shown in the figure are also obtained. Thus 88 characters are selected with only six bits of code.

NOT R5						R5						ROTATION
Ø	1	2	3	4	Ø	1	2	3	4	5		
*	11	9	Z	Ø	8	6	5	3	2	1	3	TILT
B	H	K	E	N	X	L	C	D	U	T	2	TILT
W	S	I	%	7	4	O	A	R	V	M	1	TILT
'	'	.	10	J	Y	G	P	Q	F	.	Ø	TILT

HOME

Figure 1.

The following special features are found on the I/O models but are not included in the office models:— The cycle shaft has two cams which are used to operate timing switches, the operation shaft has large heavy duty bearings, an electrical keyboard lock is provided, parity coding is built in and electrical shift is also included. Generally speaking the I/O model is built to take the load of continuous

operation and is mechanically more solid and heavier than the office models.

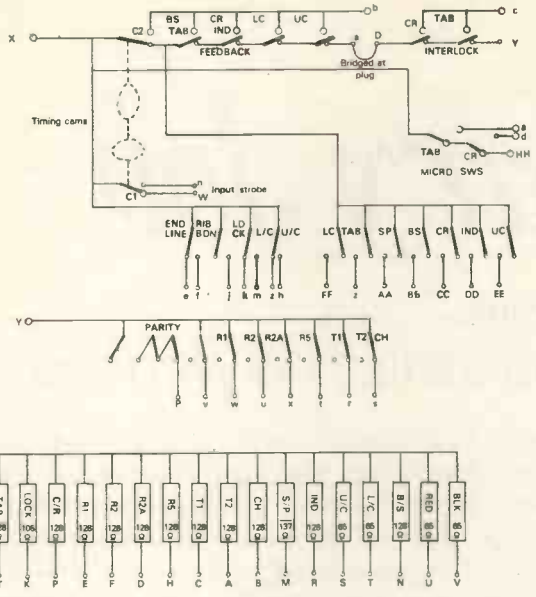


Figure 2.

Because of the previously stated requirement that the writer should be operated in a closed loop (asynchronous) mode, the controlling device must be able to detect the end of any function when printing. The feedback switches are thus used to signal to the controlling system that a function is in progress. In figure 2 all the switches are shown with their particular clutches latched (ie no function in progress) and the shift cam is shown latched in Lower case. Again this is the wiring of a Susie mechanism, which happens to be Correspondence coded. The interlock switches shown on the diagram are used to prevent multiple function selection. The Tab and C/R micro switches are fitted on the key bar and are used to supply further circuits switched by these two functions but electrically independent of the other switches. The so called Mode switches:— End Line, Ribbon (which is closed when red is selected), Lock and Case are used to determine the present status of those functions that can rest in either of two states. They may be interrogated at any time but care should be taken that they are only examined when the state is not changing. Each selection input contact closes when its key is depressed, and then the input strobe makes, sometime after the selection contacts — to gate the coded character onto the parallel wires and onward to the controlling system.

Parity contacts are included in the coding of each character to ensure that any invalid bit combination that might appear on the lines due to noise etc is rejected by the controlling system without errors being passed onward. Normally the coding is for Odd Parity, ie there will always be an odd number of 'ones' in the seven bit code of the character. Machines could have been coded with even parity to special order to it is best to check this out.

The contacts of the switches used on this mechanism are not recommended for use with switching currents of less than 10mA to prevent dirt buildup, nor for more than 40mA unsuppressed or more than 200mA suppressed due to arcing. Note

C	H	M	S	W	a	e	k	r	v	z	DD		
A	E	K	P	U	Y	c	h	n	t	x	BB	FF	
D	J	N	T	X	b	f	m	s	w	AA	EE		
B	F	L	R	V	Z	d	j	p	u	y	CC	HH	

50 way Amp plug seen from outside

Figure 3.

SOLENOID TYPE	RATED VOLTS	RATED OHMS	MAX. mA
LOCK	48	358	146
	24	105	240
SHIFT & RIBBON	48	240	217
	24	65	387
OTHERS	48	475	111
	24	128	197

Figure 4.

that these switches are liable to contact bounce and should be latched in the receiving system, ie connected to a bistable. Do not operate the switches at a higher voltage than 48vDC or unreliable operation will result.

The illustrated Susie mechanism uses 24v solenoids, but 48v solenoids are also found on Selectrics. The resistance and current ratings of both of these are shown in figure 4. The Keyboard Lock solenoids are designed for use on a 100% duty cycle, ie the current may flow continuously without the solenoid overheating, but all other solenoids are rated for 'normal operational needs when gated by the feedback and interlock contacts'. The solenoids should be suppressed with a diode, but Susie mechanisms should have had the diodes removed as the control electronics contains diodes on the circuit board. In this case your drive electronics must also provide diodes to do the same job. The only other component not shown on the diagram is the drive motor, which is normally of 240v, 50Hz Capacitor Start type, on UK machines; but US machines with 110v, 60Hz motors may also be found.

The connections on the Amp 50-way plug on the back of the typewriter are shown in figure 3, the letters correspond to those shown on the pins on the main circuit diagram, figure 2.

As this brief outline may have shown, the Selectric is a well proven mechanism which should provide a sound basis on which to construct a reliable, and high print-quality, hard copy output system for use with low cost computers. The driving electronics needed to control this mechanism may take any one of many forms depending on the degree to which the central processor can be dedicated to printer control. But that is another story . . .

313, Kingston Road, Ilford,
Essex, IG1 1PJ England,

L. ENTERPRISES

From the representatives in Europe . . . for America's leading Microcomputer magazines and books for the hobbyist, educationalist and professional alike . . . we bring you a little light browsing!

Reading maketh a full man ... Francis Bacon (1561-1626)

	UK Price	*O/Seas price where diff.*
From BYTE Publications Inc.		
Paperbytes:		
Tiny Assembler for 6800	£5.75	
Bar Code Loader for 6800,8080,Z80 & 6502	£1.75	
BYTE Magazine (Single Issue)	£2.00	
BYTE Magazine Subscription (One Year)	£15.00	
Best of BYTE Volume 1	£8.50	
From Creative Computing Press		
Best of Creative Computing Volume 1	£6.95	
Best of Creative Computing Volume 2	£6.95	
Creative Computing Magazine (Single Issue)	£1.60	
Creative Computing Magazine Subscription (One Year)	£8.50	£9.00
101 Basic Games (Revised & Reprinted Feb 78)	£5.50	
The Colossal Computer Cartoon Book	£3.95	
Computer-Rage (A new board game)	£6.95	
Artist and Computer	£3.95	
Three Binary Dice	£1.00	
From Everyone Else & Other Magazines		
TV Typewriter Cookbook by *Don Lancaster*	£7.95	
Magazine Storage Boxes (Holds 12)	£1.75	
Personal Computing Magazine (Single Back Issue)	£1.75	
Personal Computing Magazine Subscription (One Year)	£16.00	£17.00
Interface Age Magazine (Single Back Issue)	£2.00	
Interface Age Magazine Subscription (One Year)	£20.00	£20.50
Dr. Dobbs Journal (Single Back Issue)	£1.60	
Computer Music Journal (Single Back Issue)	£2.50	
Computer Music Journal Subscription (One Year)	£8.50	£9.00
Dr. Dobbs Journal Subscription (One Year)	£13.00	£13.50
Peoples Computers Magazine (Single Back Issue)	£1.50	
Peoples Computers Magazine Subscription (One Year)	£8.00	£8.50
ROM Magazine (Single Back Issue)	£1.75	
ROM Magazine Subscription (One Year)	£16.00	£17.00
From Kilobaud/73 Magazine		
Hobby Computers Are Here	£3.95	
New Hobby Computers	£3.95	
Kilobaud magazine (Single Back Copy)	£2.20	
Kilobaud magazine (Single Current Issue)	£2.00	
73 Magazine (Single Issue)	£2.00	
Kilobaud magazine Subscription (One Year)	£20.00	£21.00

	UK Price	*O/seas price where diff.*
From Adam Osborne Associates:		
INTRODUCTION TO MICROCOMPUTERS		
Volume 0 The Beginners Book	£5.95	
Volume 1 Basic Concepts	£5.95	
Volume 2 Some Real Products (Revised 1977)	£11.95	
6800 Programming for logic design	£5.95	
8080 Programming for logic design	£5.95	
Z80 Programming for logic design	£5.95	
8080A/8085 Assembly Language Programming	£5.95	
Some Common BASIC Programs	£5.95	
BUSINESS PROGRAMS IN BASIC		
Payroll with Cost Accounting	£9.95	
Accounts Payable and Accounts Receivable (avail Mar)	£9.95	
General Ledger (avail Mar)	£9.95	
From SCELBI Computer Consulting, Inc.		
8080 Software Gourmet Guide and Cookbook	£7.95	
6800 Software Gourmet Guide and Cookbook	£7.95	
8080 Programmers Pocket Guide	£2.25	
8080 Hex code card	£2.25	
8080 Octal code card	£2.25	
8080 Guide and one 8080 code card	£4.20	
8080 Guide and Both code cards	£6.00	
SCELBAL High Level Language for '8008/8080' Systems	£39.25	
SCELBAL String Handling Supplement	£8.00	
SCELBAL Extended Maths Supplement	£4.00	
8080 Standard Assembler (In book format)	£15.95	
Understanding Microcomputers & small computer systems	£7.95	
SCELBAL BYTE Primer	£9.95	
From Dymax Inc		
Instant BASIC by *Jerald R Brown*	£4.95	
Your Home Computer by *James White*	£4.95	
My Computer Likes Me .. when I speak BASIC by *Bob Albrecht*	£1.65	
Games With a Pocket Calculator by *Thiagarajan & Stolovitch*	£1.75	
Games, Tricks and Puzzles For A Hand Calculator by *Wallace Judd*	£2.49	
Calculators & Computers Magazine (Single Back Copy)	£1.60	
Calculators & Computers Magazine Subscription (One Year)	£10.00	£10.50
From Peoples Computer Company		
Reference Book of Personal & Home Computing	£4.95	
What To Do After You Hit Return	??.00	
Dr. Dobbs Journal Volume 1	£10.00	

HOW TO ORDER

Please note our prices include postage and packing. Make cheques, etc. payable to L. P. Enterprises. Payment in sterling only please.

Barclaycard Accepted.

Send to address above
Indicate Payment method:
My check, PO. IMO is enclosed
Charge to Barclaycard/Visa No..... exp date.....
Name.....
Address.....
Town..... County..... Postcode.....
Signature.....

All publications are published in U.S.A. and shipped air-freight by L.P. Enterprises. In unusual cases, processing may exceed 30 days. All orders prepaid.

You may photocopy this page if you wish to leave your magazine intact.

Trade Enquiries Welcome.

PCW OPEN PAGE

The Amateur Computer Club View

By Mike Lord

Homer G. Homebrew

I'd like to start by thanking Personal Computing World for giving this space to the ACC, and by wishing them every success. During 1977 I was constantly hearing people say that the hobby computer market in the UK was minimal. That it would never take off unless the general standard of living and level of personal disposable income in the UK improved, and unless our current prejudice against technology was reversed. However, it may be that previous marketing approaches were wrong, and, as illustrated by the runaway success of Lynx's NASCOM 1 system, there is a healthy UK market for sensibly designed products at minimum cost. The formula for success may well be to cut the frills, forget the gloss, and produce something that people can use at a price they can afford. Judging by the pre-publication information this is exactly what **PCW** are setting out to do. So I wish them well.

For those who are not aware of the ACC or its background, a few comments on where we came from and what we're doing now are appropriate.

During the mid 1960's, particularly in the USA, electronic engineers started to use digital computers as a design aid. They quickly discovered an aspect of computing that programmers and academics had known for some time, but had been keeping quiet about for fear of spoiling the august image of their professions; that computing is fun. 'An entire and complex universe under your complete control' is the best explanation of the fascination of computing that I've heard so far. Some of the more adventurous engineers decided that it would be nice to have their own computer so they could concentrate on optimising their version of LIFE without having to keep a look out for the Data Processing manager. At the same time, the first digital IC price war started, and suddenly people realised that they could afford to build their own mini computer. Also, large quantities of junk first and second generation machines began to appear in scrap yards, and the diligent searcher could find enough useable parts to reconstruct a complete system.

It soon became apparent that there was a need for some way of getting constructors in touch with each other. Computers are complex animals, and that core stack you bought for £10 is not much use unless, somehow, you can get the data for it, and the best source of information is usually another enthusiast. So, the Amateur Computer Society was started in the USA by Stephen Gray. He produced a newsletter which contained assorted articles plus, more importantly, hints, bits of advice and cries of help from members across the USA.

Some years later it was felt that a UK version might fill a need and following publicity in some amateur electronics magazines, the Amateur Computer Club was formed in 1973, along the lines of Stephen Gray's ACS.

It was soon realised that, as well as producing a newsletter and circulating it to members, the ACC could also organise meetings and visits for members, and these have become an important and popular part of the Club's activities. Also, again with the aim of helping people to get in touch with each other, a listing of members' names and addresses is sent to all members about a once year. The next list will include brief details of any computer equipment the member may have, or have access to, again with the aim of trying to help the person with a problem locate the person likely to have a solution.

As the Club has grown (at the time of writing we have about 900 members) local groups have formed within the ACC and specialist 'Libraries' have been set up by members to hold, for example, M6800 based system data and software. We have also devised a system for quickly spreading the word about, for example, a batch of cheap ASR33's that have been discovered in a junk yard in East Cheam.

Although primarily a UK Club, about 5% of our members live abroad, so we could claim to be represented worldwide. In an attempt to find out more about our members' interests, a recent survey asked the question 'Are you particularly interested in Software/Hardware'. 98% said 'both', which has

taught us something about designing questionnaires, but not much else. However, it seems that about 70% of our members at present are hardware oriented i.e. they know something about electronics and want to learn about programming, while the other 30% know about software and would like to get some hardware together. A small but growing proportion of members are interested in obtaining a computer system for use in their own business or profession; 'personal' rather than 'hobby' computer enthusiasts.

Of the various local groups within the ACC, the North West (Manchester) group has been holding well attended meetings at monthly intervals at the National Computing Centre, Oxford Rd. Members wishing to attend their meetings should check with Dave Wade (061 980 2755 home) or Ken Horton (061 799 0192 home).

If there are any ACC members interested in setting up a group in the Northampton area, would they contact Mr P. A. Gibson Daw, 479 Wellingborough Rd, Abington Park, Northampton.

Two 7768's made their debut at the Coventry Group meeting on 13th November. One of these was interfaced to a 'Meccano' X-Y plotter, and was busy writing messages and drawing patterns. This marriage of microprocessor and Meccano is an absolute 'natural' and we look forward to a fascinating progeny.

The newly formed London Group held its first meeting on 6th December last year — featuring the world's largest single board computer. This was a 6100 based machine entirely constructed on a single hand carved copper clad board measuring about 2'6" x 18". Three 7768's were also demonstrated, one of these containing the new 4K RAM and Soft Monitor boards. It is hoped that a meeting for Z80 enthusiasts will be held in mid February, but as details have not yet been finalised, would anyone wishing to attend please contact Mike Lord (address below), in early February.

Finally, for those not already members of the ACC, membership details and application forms may be obtained by sending a SAE to Mike Lord, 7 Dordells, Basildon, Essex. Tel: (0268) 411125 (weekends and after 7 pm on weekdays).

THE PERSONAL COMPUTER WORLD

★ COMPETITION ★ ★

We thought long and hard about the subject of our competition. It is obvious that the readers of **Personal Computer World** range from the beginner to those conversant with computing, and to be fair to everyone in the conventional sense would mean offering at least four 'first' prizes. So we came up with this idea:

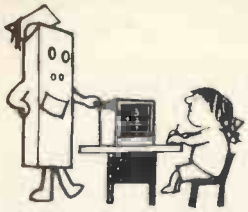
After reading through **Personal Computer World**, you might say: 'I could write a better program.' Or, 'I've designed and built a piece of hardware that would be just the thing for readers to construct.' Or, 'I think readers would like to learn about my particular difficulties in beginning computing.' Or, 'I'm a small businessman with a computer, and I've some advice to pass on.' Or, 'I'm in school, I've learned BASIC, and I'd like to explain to others how to go about learning it.' In fact, we all have ideas on what our magazine should contain.

So, to enter the competition, write the kind of article you would like to read in **Personal Computer World**. *Now here's the delicious part:* you may not win a prize, but the article may still be good enough to be published in a future issue. Don't forget to write clearly and simply. Even if you are writing about something highly technical, write a short introduction to give an idea of the subject.

The judges for the competition are: Neil Harrison, Mike Dennis, Charles Sweeten, K. S. Borland, M. N. Solomon, John Coll. The judges' decision is final. The names of the winners will be published in Vol. 1, No.3 of **Personal Computer World**.

Entries should not be more than 3,000 words long, lines double spaced with wide margins. School children without access to a typewriter may still submit entries, provided these are written in neat block letters. Closing date is 20th April 1978.

Details of prizes:
One First Prize:
A NASCOM 1
(Donated by Lynx Electronics)
Two Second Prizes:
SINCLAIR PROGRAMMABLE CALCULATORS
Twenty Third Prizes:
The SEPTEMBER '77 issue of 'SCIENTIFIC AMERICAN'. Brilliantly written articles on the microworld.



DO WE WANT OUR SCHOOLS TO BE PERSONAL OR TERMINAL?

Charles Sweeten

At the present time in schools, colleges and education authorities around the country, there are two main types of policy towards the provision of computer facilities. There are those who believe that the only way to provide hands-on experience at a reasonable cost is to make use of the expensive installations that are already going into the administrative departments of many councils, by connecting each school via a GPO line to a teletype. The other policy believes that as complete computers get cheaper, the only way to provide this experience is to install micro systems in each school. There are some advantages in each method and I would like to examine both in more detail.

The policy of centralising the computer system inevitably means that a large computer will be necessary at the centre. Such machines are expensive and it cannot be assumed that a machine of this size would have been justified without the necessity of providing this educational service. Suppose that this had the effect of increasing the purchase price from £400,000 to £600,000 in an area with 200 schools, then each school could be said to be benefiting to the extent of £1000. Equally, in an area of 10 schools, it is likely that the cost of providing them with a service will be at least £10,000. Each school would also need a teletype, and colleges of education or technology would need a large number of teletypes. So that the investment of capital for a school might typically be of the order of £2000 and for a college it would be much higher. This takes care of the capital expenditure. In order to maintain this equipment, maintenance of the teletypes will be needed and this is typically £100 p.a. plus an overhaul after every 1500 hours use costing £250. Maintenance of the central computer would be necessary on that part of the capital which was justified on educational grounds and the lowest rate at which this could be accounted would be 6%. Further, in order to maintain the system as an educational system it would be necessary to have a support staff. To what extent it is necessary to build up such a team must be debatable but some presence is undeniable. Hatfield has a team of about 13, and London has a sizeable team also. So we come out with a true annual expenditure of about £500 plus the support staff, allowing for paper and 200 hours phone time. Of course the school does not pay all of this but ultimately the Local Education Authority does.

What are the advantages and disadvantages of such a system? Anyone who has had the job of maintaining a system which is being used by hordes of enthusiastic but heavy handed amateurs trying to squeeze in one more hour than the budget will allow, will know that it is nice to have equipment built to the highest standards, maintenance contracts which guarantee a 24 hour service and resident experts on all aspects of software and hardware. This costs money, but to some extent it can be achieved in the centralised system. An education authority can build up a team of people to satisfy the needs of the schools and within that team there will be both continuity and expertise. One of the difficulties experienced by schools or colleges who try to 'go it alone' is that it is very nearly impossible to find a member of the staff who has the time or ability to develop support programs for use in teaching. The kind of programs that may be needed might include special software to operate an old IBM golfball typewriter or perhaps a database for the geography department. With the centralised system the latter demands, and many more, can certainly be satisfied. Inherent in the nature of the central system is the great storage capacity and processing power that is available. This allows the users access to a large library of programs with the minimum of difficulty, and to run programs which make heavy demands on processing time, such as some simulation programs. What is more, the results will be available to all the schools in the area, and there will be an expert available to explain it to each school. The very existence of this team makes the organisation of meetings, conferences and courses, to explain teaching methods and additions to the system, much easier than would be the case where schools are on their own. All this tends to cast the teacher in the school in a passive role where all he or she has to do is sit back and wait for the ideas and materials to roll in. The teacher who is anxious to experiment and contribute may be less motivated by such a system than would be the case if they had an actual computer in the school. But the real disadvantages arise from the limitations imposed by a telephone system. It is possible to have many telephone sockets for use in different areas and it is possible to use the phone for eight hours a day, but certainly most schools would find that the cost of this was beyond them.

The other method of satisfying the needs of an educational establishment is to have a complete micro system tailored to the needs of that establishment. The cost of such a system obviously depends on exactly what is required, but it is interesting to

look at a system that is to a large extent comparable with what is provided by a centralised system. Such a system within a school would need to provide programming access for about five hours a week and would need to provide storage of moderately large programs. It would need to be able to respond, and to provide results, as quickly as the centralised system. Such a system is of course possible, and in certain circumstances may even be quicker. We can buy a computer with 24K bytes of memory for £500. Add to this the cost of a cassette recorder and a VDU controller and a TV and we reach £900. If we are unable to find a tame electronics expert to put it all together, then this may increase the cost to £1200 to have it done for us. It is obvious that such a system will be different from that provided by a centralised system, but proponents of either system will always be able to find something that can be done on the system that they favour which cannot be done on the other. The main difference of course is that files on the micro system are contained on tape cassettes, rather than on high speed disks, and can only be retrieved by finding and loading the right cassette and replaying it at 30 cps. This is very much slower than disk retrieval but in many situations might be regarded as acceptable. But it is certainly an improvement on the paper tape handling that so often seems to be a feature of remote teletype operation. A further difference is that a VDU does not produce a printed output to use as a record of results or to use to examine programs away from the keyboard. This might be considered a distinct disadvantage and at present there is no satisfactory answer to this problem if it is a problem. Maintenance of the hardware is another question that has to be considered. A maintenance contract is not normally available on micro equipment at these sort of prices and any intending user must consider what can be done when a breakdown occurs. When used the same amount as a remote teletype, breakdowns will admittedly be few and far between but when they do occur, the offending part may be replaced without great cost. If an education authority can take a strong lead within its own area, most usefully through the work of a college of education, they may be able to organise a common specification for the micro systems within their area. This means that part of the work of the central support team, which remains just as necessary a part of the educational program as in the other type of system organisation, can be to provide expertise in case of breakdowns. Such a commitment to a specification does not mean a slavish acceptance of set system frozen at a particular stage of technology; for, provided that the basic processor chip is agreed, it should be possible to implement the developments that take place provided that these are also agreed by the area and provided that these are not proceeded with at too great a pace. In the Berkshire area work on this sort of agreed specification is being coordinated at Bulmershe College of Higher Education and it will be interesting to see if the schools in the area will follow it in practice and whether the LEA will fund the essential back-up in the form of centrally organised and permanent technical expertise allocated solely to helping the schools. The approach adopted at Bulmershe should go some way to overcoming the difficulties engendered as a result of using cheap equipment, which inevitably is more liable to fall to pieces under constant battering than is industry standard, expensive equipment. As far as development work in writing software and programs for educational use is concerned there is no reason to doubt the need for a support team as stated before. There might even be some advantage to be gained from the necessity for its members to visit schools and colleges rather than vice versa. It is equally important, as in any structure proposed, that institutions should be able to communicate efficiently with each other, and the ILEA is one area that springs to mind where the effectiveness of this can be seen. Advantages that the micro system undeniably has, that it is hard to envisage in a centralised system, are the active role that staff and students are encouraged to play, the extreme portability of the system, and the opportunity for anyone who wishes to see how it really works. There is some comfort to be found for at least some pupils in discovering that a computer can be turned off.

When comparing centralised and micro systems in general though, there is one enormous advantage that the micro or in-house system possesses and that is that it may be used to a far greater extent with no increase in cost. A statistic that is used too infrequently is the cost per pupil of hands-on experience. Quite often it will be found that a remote terminal is being used for about 30 pupils in a large school at a cost of £500 pa. It is equally easy to envisage a micro system where 100 pupils are exposed to the keyboard at a cost of £100 pa.

If one now compares what can actually be bought for the cost of a centralised system we get a different picture. For a further £800 on top of the £1200 already suggested, one can buy a dual floppy disk. This transforms the simple system into a relatively fast and powerful one which bears comparison with large centralised time sharing systems. The only area of doubt that remains is the inefficiency of some of the software available on some micros. In this respect it is interesting to see the surveys carried out by KILOBAUD magazine in the USA and by John Coll for MUSE in this country which show timings of some BASIC programs run on different machines using various BASIC interpreters. The results show conclusively that the efficiency of the software is of much greater importance than the efficiency of the hardware.

At this stage one should perhaps remember that there is another method of providing computing facilities for large numbers of pupils which has had great support for a number of years, and this is to mark cards and then process them in batch mode away from the students who have marked the cards. The advantage of this is that the bottleneck which

builds up at the keyboard does not exist and that the cost per pupil becomes extremely reasonable. This method could be used in conjunction with either of the systems discussed, but the cost of the equipment remains very high. There is also the problem of turn round time, where irritating minor faults can cause the program to be returned for correcting a number of times, and it can easily take two weeks to run a simple program. This loss of immediacy is a serious loss for the student. But, until a multiplicity of keyboards can be achieved, it may remain the only alternative in some cases.

For many schools this type of discussion appears to be building castles in the air. Education authorities are facing large cutbacks and are reluctant to spend money on 'luxuries like computers', but once they have accepted the principle of computer education there will eventually be some money available somewhere and it is as this phase develops that schools and colleges need to be in contact with those who have got through this stage. Many mistakes can be made and they are expensive and often not correctable in these hard-up times. With the technology developing so fast it is essential to gather the expertise which already exists in some schools and colleges. MUSE is a society that exists for this precise purpose and which is able to give advice to those who are just hopeful and waiting for their school to win the pools, as well as to those who have well-established systems, by keeping all interested institutions in touch with each other through meetings and a newsletter. MUSE is currently doing work on hardware, system software and classroom applications, and the MUSE standard for implementing micro systems is a useful guide for anyone contemplating such a move.

One LEA man with responsibility for computers in schools in an area was recently asked what that area was doing about computing and, with a straight face, he answered: 'nothing'. It is no use sitting back and doing nothing because the money is not here today. Plans take time to prepare and knowledge takes time to be acquired and agreement may have to be reached with other schools about a common specification.

We stand at a turning point in the history of computing in Secondary education. We have passed through the eras of visiting Data Processing departments to gaze at tape drives clicking round, and of carrying suitcases of marked cards to a friendly company five miles down the road, and we are now coming to the end, I think, of 'cheap' computing by telephone. What comes next is surely the personal computer in school and college and this will shortly be followed by the personal computer plugged into the TV at home in place of the present TV games. But one of the features of this area of education is that so many schools have to start at the beginning of this series of developments and work their way through. Perhaps it is because they start, like so many have done in the past, with some staff member developing an interest but having no knowledge whatsoever. Perhaps it is because LEAs distrust such new technology. Whatever it is, one wishes that those with any remote interest could short-circuit the process by meeting those who have reached the present state of the art (to coin a phrase).

So let me give a purely personal view of what I would do if I wanted to start computing in my school.

You should join CEG and MUSE immediately; they only cost £2 a year at present and this must be good value. You should pluck up your courage and attend as many meetings of both as you can; you won't understand what some people are saying but they are quite used to this, and if you can manage to ask someone what you should do, you will find them only too eager to help. In conjunction with someone who knows what they are doing in another school, you should prepare a specification for a micro system in terms of the hardware and software currently available, and you should analyse the capital and annual costs carefully. You should then ask your headteacher to present a special case to the LEA (or governors or whatever) for provision of funds outside the normal and you should contrast this unfavourably with the costs of a centralised system. If this fails, enquire diligently for any trust funds that may exist, as these are much more common than many imagine, and try to find a local industry or business that might be persuaded to help if someone else was doing the same thing. It is also surprising how often a parent in a large school is working in a computer-related industry and this may lead to a fruitful relationship. And if all that fails you should change school because by now someone will need you to do the same in a more favourable school.

If there is one simple message that sums up what I am saying, it is that the personal system in a school or college gives education the flexibility for the future at a price it can afford, and that all intending or existing users of any kind of system should join an organisation that keeps them in touch with what is going on in other schools and colleges.

If you are interested in joining MUSE
write to the Secretary at 18 South Road, Oundle, Peterborough.
Telephone: 08322 3430

DIRECT ADDRESSING: WHERE TO GET YOUR PERSONAL COMPUTER

John Coll

A LOOK AT HARDWARE AND SOFTWARE

Two problems face anyone attempting to review the available hardware and software: firstly there are so many systems available that a full comparison would lose its usefulness because of the sheer volume of information presented. Secondly, new products have been coming onto the market very rapidly and will continue to do so. Whatever is written will be out of date in six months. Given these constraints I have attempted to give a review of the market at present.

In choosing a computer system the following points need to be borne in mind:

1. *The Central Processor (CPU) that is used.*

There must be at least twenty-five to choose from, four seem to be in particularly common use.

a) the Intel 8080 — the 'industry standard'
b) the Zilog Z80 — a much improved version of the 8080 which will run all 8080 software
c) the Motorola M6800
d) the MOS Technology 6502 — which is similar to the M6800, better in some ways not so good in others.

Software written for one CPU will not run on another with the exception of 8080 software which will run on the Z80. Clearly any program written in BASIC will run on any machine that can itself run BASIC, though usually with minor changes. The same applies to FORTRAN and other high level languages. The choice of CPU determines amongst other things, the ease of assembly language programming (the M6800 is the easiest) and the speed of execution of complex programs (probably the Z80 has the edge there).

2 *The Price and Availability of Software.*

There are two factors here: do you want to buy 'off the shelf' software or are you prepared to do a bit of 'patching' to get it to work? If you want 'off the shelf' software then your main supplier will be the hardware supplier of your particular computer.

If yours is an Altair (8080 based) machine there is some very good software — but at high prices. For example, Extended BASIC will cost you £112 if you are buying 16K of Altair memory at the same time and £350 if not. SWTPC software for their 6800 system is £10 for almost everything. It is not as smooth as the Altair product but is adequate and improving. Research Machines (Z80) have some very good software available but it looks as if it will be quite expensive. *Note:* If you are prepared to 'patch' other software to suit your system then you have a wide range of sources. A lot of good software is available for the 8080, Z80 and M6800, less for the 6502.

3. *The Price and Availability of Hardware.*

In this field the Altair wins hands down at present. Not only can Altair supply a good range of add-on units but their bus (arrangement of connections on the mother board) has become a standard in the States. Numerous manufacturers have produced boards to plug into this bus and the resulting competition is reflected in the prices and facilities offered. None of the competition can offer such a range of boards. The establishment, by the IEEE, of an Instrument Standard Interface (IEEE-488) could well greatly help in this direction in Europe. Motorola are about to launch an IC designed to interface the 6800 to the IEEE-488 and Zilog will, no doubt, follow suit.

4 *The Pace of Development of Hardware and Software.*

Here one is considering the IC Manufacturers as well as the computer manufacturer. Software development is largely in the hands of independent software houses and they seem to be concentrating on the Z80 and M6800. Much very good Z80 software has been produced already, notably by T.D.L. in the States. There seems to be a slightly wider range of software for the 6800 but much of it lacks finesse at present.

Now a very brief and factual look at six systems, three available at present, and three to be released very soon — perhaps before February 1978.

1. *The Altair 8800b* is supplied by Compelec Electronics Ltd., 107 Kilburn Square, Kilburn High Road, London NW6 6PS. Tel. 01-328 1124. The basic unit includes CPU board, interface buffer board, case, power supply and front panel and sells for £860. The motherboard is to the S-100 standard and is, optionally, available with 18 slots. Altair offer the following accessories: 4K static RAM £227, 16K static RAM £575, 16K dynamic RAM £289, 2K PROM (board only) £92, Serial interface for teletypewriter £135, four parallel interfaces £95, Analogue/Digital-D/A £275, Process control board with eight relays £172, Real time clock £135, Line printer and control interface £1,100, Disk drive and controller £1,255, Minidisk and controller £842. Other manufacturers provide similar products and many others, particularly, graphics in black and white and colour. Kit prices are approximately 10% below those quoted. Altair software is good but expensive. The following prices assume that Altair memory is being bought at the same time: Disk BASIC (£150), two assemblers, monitor, text editor and debug (£56). A multi-user BASIC is available.

The Altair system is very well established and solidly constructed. It is a 'safe' buy.

2. *The South West Technical Products Corporation S6800* is manufactured, under licence, in the UK and sold by Computer Workshop, 174 Ifield Road, London SW10 9AG. Tel. 01-373 8571. The basic unit includes CPU, motherboard, interface to teletypewriter, case, power supply. 4K of RAM and the

MIKBUG ROM operating system and sells for £353. The motherboard can accept six general boards and eight interface boards. Computer Workshop offer the following accessories: 4K static RAM £107, 8K static RAM £240, 16K static RAM £480, Serial output board for teletypewriter etc. £37, Parallel output/input with transistor buffers £37, VDU £345, CUTS (Computer User's Tape Standard) interface £100, 40 column printer £250, Floppy disk and controller £950, Minifloppy disk and controller £636. Computer Workshop also supply the following software at £10 per tape: 3K BASIC, 4K BASIC, 8K BASIC, Editor and Assembler, Floating point unit, Co-resident assembler editor, Space Voyage game. A dis-assembler is available for £15 as well as a number of games. A multi user BASIC is available. New hardware and software is constantly being produced. The equipment is very good value for money and is well supported.

3. *Research Machine Ltd. 380Z* is manufactured by Research Machines Ltd., 209 Cowley Road, Oxford. Tel. Oxford (0865) 49791. The basic unit includes a Z80 CPU with a good monitor, a built in VDU and keyboard that will drive an unmodified domestic TV, the VDU has a graphics facility as standard. Interactive text editor, 5K BASIC, 2K BASIC, utilities and games software are also included free of charge. The price for the above depends on the amount of memory supplied, as follows: with 4K RAM £848, with 16K RAM £1,038. The construction differs from all the other computers reviewed here in that no motherboard is used. Instead a flexible cable with sockets is used. (See photograph). This, it is claimed, will be more reliable than a motherboard system. Certainly the internal design is very neat. The whole of the above system with up to 32K of RAM is accommodated on two small boards. It is very impressive. Additional software that is offered includes a Macro Assembler generating relocatable object code, an obsolute assembler, as well as Algol and Fortran compilers. TDL in the States sell some very good software for the Z80 including 8K BASIC $50, 12K BASIC $95, Monitor $25, Text Editor $35, Text Output Processor $35 and the Relocating Macro Assembler $50. The following items are under development by Research Machines: a mini floppy disk system, a multi user BASIC and a high resolution graphics unit.

Note: The TV display and cassette are not included in the above prices.

4. *Tandy Corporation TRS-80* is to be imported by Tandy Corporation, Bilston Road, Holyhead Road, Wednesbury, Staffs. WS10 7JN, though the computer is to be sold through their chain of 110 'High Street' shops. It is hoped that the machine will be available in the UK in February or March of 1978 and the cost of the basic unit is expected to be about £500. The basic system consists of four units, the keyboard-computer-VDU circuitry, a 12 inch monitor, a power supply and a cassette deck. The computer uses a Z80 microprocessor. Included *in* the computer is 'Level 1 BASIC' in ROM so there will never be any need to 'load' BASIC. The VDU offers a graphics mode as well as software-selectable 32 or 64 characters per line alphanumerics. Included in the basic price is 4K of dynamic RAM. The version of BASIC supplied is a very simple subset and does not include any trigonometric functions, square root, or renumber facility. A single array and two string variables (of maximum length 16) are allowed. It is a very basic BASIC.

Tandy anticipates that two printer interfaces, serial and parallel interfaces, floppy disk and telephone modem will follow in the very near future. It is interesting to note the software that is being offered with this system in the States; Blackjack and Backgammon are supplied free, a payroll routine for 15 people $19.95, an education set teaching addition, subtraction and multiplication complete with teacher's guide(!) $19.95, a Kitchen set including Menus, conversion tables, computer directory and message centre $4.95 and a personal finance set $14.95. Clearly every home should have one! Still it is Z80 based so all that nice software from TDL should run satisfactorily. The TRS-80 has, clearly, been very well thought out and looks as if it is designed to last.

5. *Heath Ltd. H8 and H11 computers* will be available from Heath (Gloucester) Ltd, Gloucester GL2 6EE, in April or May 1978. The H8 is based on the 8080A CPU and uses their 'exclusive' 50 pin bus. The basic unit incorporates CPU, power supply, bus and box and BASIC, assembler, editor and debug program. With the addition of a memory card and serial I/O card the price rises to $625, which, to my mind, puts it right out of the running, especially when you realise that that is a kit price unlike all the other prices quoted in this article.

The H11 is interesting because it uses an LSI-11 which runs all PDP-11 software. The basic H11 consists of an LSI-11, 4K of RAM, power supply and box and the following PDP-11 software: editor, relocatable assembler, link editor, absolute loader, debug, executive, relocatable dump, absolute dump, BASIC and 2 versions of FOCAL. The price is $1,295. Several peripherals are available, in the States, for both the H8 and H11.

6. *C.B.M. P.E.T. 2001.* from Commodore Business Machines, 446 Bath Road, Slough, Berks. Tel. Burnham (06286) 3224. I have tried very hard to obtain information from them without *any* success. This seems to me to be very bad public relations since they have publicly displayed the machine (for example in London in May 1977) on more than one occasion. To create an interest which they then totally fail to support is not a good sign for the future. However, the product is as follows: 4K RAM, 8K BASIC ROM, 4K operating system ROM, 9 inch integral VDU, integral audio cassette (though there is no indication to what standard). The cassette system incorporates a file management system both under the operating system and under BASIC. The VDU supports a limited graphic facility and is able to display reverse field characters (black on white). Perhaps the most interesting feature is the incorporation of an IEEE-488 instrument interface and it seems reasonable to assume that C.B.M. will produce peripherals to that standard. The computer uses a MOS Technology 6502 processor and it is interesting to note that MOS Technology were recently taken over by C.B.M. Clearly they are very heavily involved here. The price of the machine is expected to be about £600 and it is to be released 'sometime during 1978' probably very early in 1978. There is a dearth of software for the 6502 and Commodore will be under real pressure to produce a good range. Apparently they intend to produce programs for such things as video games and inventory control. Clearly the P.E.T. 2001 will have an enormous impact when it is released.

'The proof of the pudding is in the eating' and it is the eventual performance of the hardware/software *combination* which is of real interest. Apart from the cost aspect one is interested to know what facilities are offered and how fast the system is. Particularly with large scale simulations, speed is a vital factor. To give some strictly limited guidance on the comparative speed of a number of computer systems I have collected together the execution times for a set of benchmark programs run on a number of small computers. These should be treated with caution. They simply tell you how long the computer took to execute a particular set of routines. They say nothing about other facilities which may be offered — for example string handling. They reflect the situation today and this may well radically change with faster hardware and improved writing of the BASIC interpreter. None-the-less they are interesting. The first seven benchmarks were used in a series of tests carried out in the States and published in an article in the June 1977 issue of *Kilobaud*. The eighth benchmark has been introduced to test the trancendental functions of the various interpreters. Unusually poor performance on this benchmark is a clear indication of the use of poor algorithms and is more a reflection on the programmer than on the machine.

Here are the benchmarks:

BM1.
```
300 PRINT 'S'
400 FOR K=1 To 1000
500 NEXT K
700 PRINT 'E'
800 END
```

BM2.
```
300 PRINT 'S'
400 K=0
500 K=K+1
600 IF K<1000 THEN 500
700 PRINT 'E'
800 END
```

BM3.
```
300 PRINT 'S'
400 K=0
500 K=K+1
510 A=K/K*K+K-K
600 IF K<1000 THEN 500
700 PRINT 'E'
800 END
```

BM4.
```
300 PRINT 'S'
400 K=0
500 K=K+1
510 A=K/2*3+4-5
600 IF K<1000 THEN 500
700 PRINT 'E'
800 END
```

BM5.
```
300 PRINT 'S'
400 K=0
500 K=K+1
510 A=K/2*3+4-5
520 GOSUB 820
600 IF K<1000 THEN 500
700 PRINT 'E'
800 END
820 RETURN
```

BM6.
```
300 PRINT 'S'
400 K=0
430 DIM M(5)
500 K=K+1
510 A=K/2*3+4-5
520 GOSUB 820
530 FOR L=1 TO 5
540 NEXT L
600 IF K<1000 THEN 500
700 PRINT 'E'
800 END
820 RETURN
```

BM7.
```
300 PRINT 'S'
400 K=0
430 DIM M(5)
500 K=K+1
510 A=K/2*3+4-5
520 GOSUB 820
530 FOR L=1 TO 5
535 M(L)=A
540 NEXT L
```

```
600 IF K<1000 THEN 500
700 PRINT 'E'
800 END
820 RETURN

BM8.  300 PRINT 'S'
      400 K=0
      500 K=K+1
      530 A=K↑2
      540 B=LOG(K)
      550 C=SIN(K)
      600 IF K<100 THEN 500
      700 PRINT 'E'
      800 END
```

Two points emerge immediately from these results: firstly minicomputers do not enjoy a clear advantage in speed when compared with micro-computers and secondly there does not seem to be a clear leader in the microprocessors. The best Z80 time for BM7 and 8 was 37.8s achieved with a 4MHz clock, the best M6800 time was 53.7s with a 898kHz clock. It is reasonable to expect that the M6800 with a 2MHz clock (now possible) would perform in about 24s.

May I, in closing, make two requests. These results were put together at great speed during early October 1977. They are very preliminary and a lot more testing needs to be done and is under way. The other point is to ask you, if you have access to a mini or micro-computer, to run the benchmarks on your machine. *Before* you do that please write to me and let me explain *exactly* how they are to be run. They would be useless if they were not all run under exactly comparable conditions.

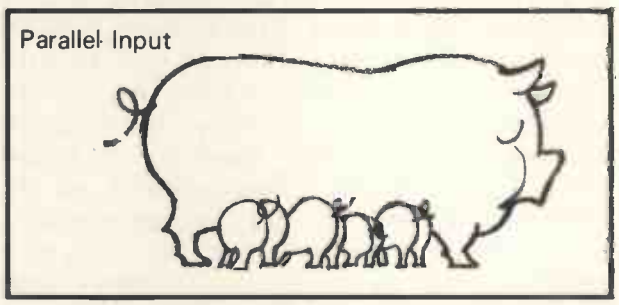
Machine	Software and Version Number	Yr.	Clock Frequency
IBM 370/115	COUL Lisbon R1.2		
NOVA 1220	Time Shared V 091-26-04	71	
NOVA 1210	Stand Alone System V9.02		
PDP8A Classic	OS/8 R3		
PDP8E	Educomp V3.4		
NOVA 1220	Multi-usr Ex B SOS R9		
T.I. 9900-10	Pre-release	77	
R.M. 380Z	TDL Zapple 8K V1.3	77	4MHz
NOVA 2-10	Ex B R4.0, RDOS R3.03		
NOVA 1210	4K Single User	73	
ALPHA LSI 2-20	CAI BASIC-2		
Z80 home made	TDL 8K V1.3		2.5MHz
SWTPC MP68	Altair 680b V1.0 R3.2	76	898kHz
SWTPC MP68	3K integer		898kHz
PDP8M	Edusystem 10		
R.M. 880Z	Palo Alto Tiny B V MB1.8	77	4MHz
PDP8L	8K		
Altair 8800b	Altair Extended B V4.0	77	2MHz
Altair 8800b	Altair Extended B V4.1 Disc	77	2MHz
PDP8L	Edusystem 10 4K		
PDP8L	Edusystem 10 4K modified		
Altair 8800b	Northstar V6 R2	77	2MHz
Altair 680b	Altair R1.1	76	500kHz
Altair 680b	Altair 680b V1.0 R3.2	76	500kHz
PDP8b	DEC 4K CINET		
IBM 5100	IBM 5100		
PDP8E	Edusystem 20 non-alphanumeric S.U.		
PDP8L	Edusystem 20 12K		
PDP8E	Edusystem 20 config for 1 user		
SWTPC MP68	8K V2.0	77	898kHz
Mycron 8080A	Tiny B	75	2MHz

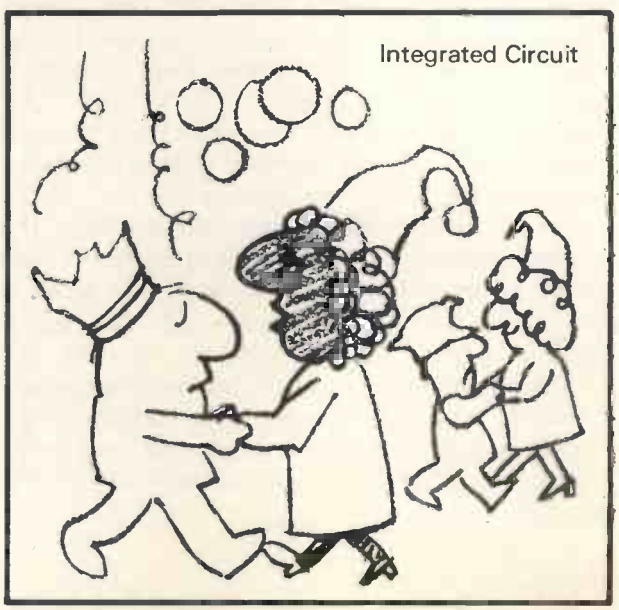
V=version number, Ex B=Extended Basic, SOS=Stand Alone System R=Revision or Release number, CAI=Computer Automation Inc, TDL=Technical Design Labs, S.U.=Single User.

BM1	BM2	BM3	BM4	BM5	BM6	BM7	BM8	BM 7&8
1.3	1.4	2.3	1.9	2.1	3.1	4.6	1.9	6.5
2.3	4.5	8.4	10.5	11.0	16.9	23.8	2.7	26.5
1.0	3.0	7.7	9.1	9.6	16.5	24.0	2.6	26.8
2.7	2.5	5.5	6.1	6.7	19.2	24.2	3.0	27.2
1.4	4.4	8.5	11.8	9.3	15.8	27.8	3.7	31.5
1.8	4.0	10.0	11.6	12.4	20.8	29.8	3.8	38.6
3.2	2.4	4.5	5.5	6.1	25.0	29.0	—	—
1.4	6.5	13.2	13.9	15.0	22.3	31.6	6.2	37.8
1	3	8	9	9	16	25	16	41
0.6	1.8	5.8	7.3	7.6	12.5	18.2	32.2	50.4
2.2	7.0	14.5	15.0	16.0	29.5	48.0	3.7	51.7
1.8	8.5	18.4	19.5	21.1	31.5	44.7	8.9	53.6
1.4	9.0	16.8	18.1	20.0	31.0	45.2	8.5	53.7
1.6	8.7	15.6	16.4	21.6	31.4	46.5	—	—
3.0	4.6	14.6	15.0	15.4	34.0	47.1	7.0	54.1
0.9	8.6	13.6	17.2	21.2	—	—	—	—
4.0	6.8	17.0	20.2	21.0	38.8	57.0	10.8	67.8
1.9	7.5	20.6	20.9	22.1	37.0	58.5	9.9	68.4
1.9	7.5	20.6	20.9	22.1	37.0	58.5	9.9	68.4
4.8	7.0	16.8	20.2	20.5	44.8	61.5	9.5	71.0
5.0	7.2	18.0	21.5	22.0	49.5	67.0	10.2	77.2
2.4	9.0	26.8	31.3	33.4	50.1	72.5	22.0	94.0
2.6	16.4	30.9	33.7	36.6	56.0	81.9	15.0	96.9
2.5	16.2	30.6	33.2	36.2	55.9	81.7	15.3	97.0
4.8	29.0	45.8	55.2	60.0	73.0	133.6	12.0	145.6
4.5	21.1	57.4	54.5	59.0	88.2	174.9	26.7	201.6
8.5	28.0	52.3	52.4	65.4	137.5	203.7	12.4	216.1
12.5	39.2	74.6	73.8	91.0	187.0	288.5	16.5	305.0
12.5	47.3	86.0	86.1	96.4	194.4	333.6	18.3	351.9
15.6	25.4	96.9	105.9	109.8	174.5	205.1	172.0	377.1
12.1	20.9	57.6	58.4	101.0	204.0	306.0	79.6	386.0

I would like to thank the many people who have helped to collect data for this investigation. Clearly without their help it would have been impossible.

John A. Coll
Oundle School
Electronics Department
Oundle
Peterborough, PE8 4AQ
England.





CUTTING THE WORLD DOWN TO SIZE

Werner Sieber

'To see a World in a grain of sand, and Heaven in a wild flower, Hold Infinity in the palm of your hand, and Eternity in an hour.' — W. BLAKE.

Introduction

Fatal crash in living room. No victims.

In science, models existed for a long time before computing machines of any description made their entry onto the scene. A model, in the sense which shall concern us in the present article, has, at its origin, little to do with computing. The advent of electronic digital circuits has however led to a revolution in the practical art of modelling. Although the basic mathematics used had been invented much earlier, their application was too tedious for human endurance until man was relieved by the machine in this field.

In the most general sense, any self-consistent reflection, in the human mind, of a part or aspect of Reality might be called a model, although nowadays the term is mostly applied to 'externally' stored information structures. When the Ancient Greek philosophers, by pure mental induction, concluded that all matter must be made up of small indivisible particles which were mostly alike but, by combining in different ways, gave rise to the infinite variety of the visible world, they were constructing a model of this world. Present-day scientists content themselves with tiny parts of the world, like hydrogen atoms and deoxyribonucleic acid molecules (J.D. Watson, The Double Helix, London, Weidenfeld & Nicolson, 1968.) In the latter case, a weath of data on this biological molecule had been available, yet the final and decisive melting together of this empirical knowledge in the shape of the double helix took place in the human mind. The model was *generated by a creative process.* Once it was 'born', many more observations could be linked, *explained* (eg the replication of DNA) and *predicted* — an essential purpose of modelling in general. Models are neither restricted to things invisibly small (Ptolemy and Copernicus, on the basis of the same astronomical observations, created two very distinct moving pictures of the Universe) nor indeed to things material, as the proliferation of economic, sociological and psychological models proves. Hegel's view of the dialectic progress of history and the Club of Rome's 'Limits to Growth' are just two examples. The human activities of learning and interaction have also been the subject of model studies. Models of such extended systems as the world weather or the US Economy are still dreams of the future.

Since it is extremely difficult to define exhaustively such general and versatile entities as models — and agreement on the definition is far from complete — we shall forgo any attempt at formal definition. We shall try to build up an understanding of the concept by discussing the questions arising at the various stages of making, using, testing and judging scientific or technological models. We shall also try to delineate the philosophical characteristics of models as opposed to real objects on one hand, theories on the other. We shall see that by *simulation* one understands the modelling of physical processes by computer programs, and the substitution of experiments by such programs. Our questions will however not be restricted to models of this type but include, especially for illustrative purposes, material objects used or constructed as models. Numerous examples will be mentioned in order to give an impression of the truly interdisciplinary nature of the model concept.

The first question we ought to consider is, naturally, that of the use and purpose of models.

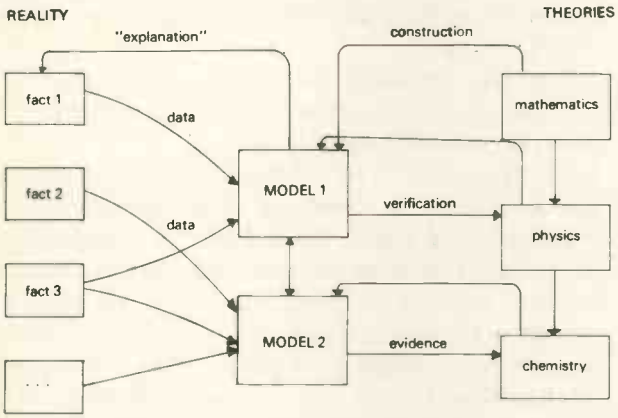
What do we need models for?

Scientific research and technological development are based on a continual interaction between experience and theory. Models play an essential part as mediators between the two stages of this dialectic process. In order to develop a theory, we have to extract information from the surrounding world according to a certain plan. Only certain phenomena will be relevant to the field of a particular theory. It is useful to collect and condense this *essential information* in the shape of a model, abstracting it thereby from all the 'irrelevant', 'disturbing', or 'obscuring' facts. (The artifacts that are likely to be introduced at this stage will be discussed later).

If we want to study, for example, the impact of a car collision on the body of the driver, the model used, that of the human body, may be rather simple. The dummy has to be of the correct size and mass, and roughly reflect the shape of the 'real object'. The *effects* of the crash can then be recorded in two ways: either we construct the model from materials similar in strength and elasticity to those of the original, and observe the resulting changes after and/or during the experiment (high speed camera) — or we install various *sensors* in the dummy, which are connected to measuring and recording devices. We then obtain a record of forces, stresses, accelerations etc, as a *continuous function of time.* From this and the known properties of human flesh and bone we may then compute the expected damage.

A third, more radically different approach is the following: We dispense with the battered old car, the rails and the pulleys, the dummy and the sensors altogether. We represent a number of sites in the human body by so many sets of *parameters,* numbers which are substituted in appropriate *differential equations.* We then chose a schedule for the 'fatal crash', eg position of the car at discrete

times 0 sec., .01 sec., .02 sec. etc. It is not important how long our particular computing machine actually takes to work out the results for every step. We can choose the time intervals as short as we wish. We shall probably still save time overall, considering the amount of preparation rendered unneccesary by this *simulation method*. On the other hand, it is obvious that a much closer *quantitative* description of the original object is needed in order to set up the equations required. This information has to come either from *theory* or from assumptions based on *analogy* to known systems, since the actual 'empirical' data is what we want to get out of our 'experiment' (the quotation marks are to caution the reader against the make-believe character of any simulation as compared with a true experiment). Probably we shall have to invoke theories belonging to various disciplines in order to construct a satisfactory model. The circle of the dialectic process (Fig. 1) is thereby closed.



Several reasons for the use of models are distinguishable in the above example, besides their scientific function as links between experience and theory: The ethical taboo against experiments on the real object, the saving in space and expense, the wider choice of conditions and, last but not least, the possibility, with simulation programs, of expanding or contracting (or even reversing) that haunting factor, *time*.

A different but no less important purpose of models is that of demonstration and illustration. As we see in the case of this very essay, it is virtually impossible to communicate ideas of an abstract or general nature without resorting to illustrations or 'examples'. The human mind has a surprising (and sometimes disquieting) ability for generalisation, while our skill in visualising purely abstract information is often less well developed. Pupils in a science class cannot be expected to go through all the battles waged by the outstanding minds that produced, over the past few centuries, the present understanding of the material world. Models are therefore constructed, *starting from axioms and theories known to the teacher*. These models emphasise or exaggerate certain aspects of reality. They leave out distracting factors like friction, impurities, natural fluctuations and irregularities of shape. Expansion or contraction in the dimensions of time and space is possible. Certain physical quantities are symbolically represented by other quantities. At the same time the models should appear sufficiently similar to everyday objects in order to appeal to our basic senses of perception, since sensory experience is the foundation, as well as the motivating stimulus, of every scientific concept or framework.

An offshoot of this didactic application of models is found in the proliferating variety of *simulation games* played on video screens, sometimes supported by sound effects. Car races can be run in this way with no danger to the 'bystanders'.

Pupils and pleasure-seekers are not the only ones to take advantage of the illustrative capacity of models. The scientist or mathematician, after setting up a highly abstract theory, often wants to test its self-consistency and freedom from contradiction by applying the theory to a model, ie a 'concrete' case specially constructed for the purpose. In other instances, the situation as described by words or formulae is so complicated that it resists visualisation. A simple model often permits unexpected insights. If a chemist is confronted, for example, with a moderately complex organic molecule (Fig. 2),

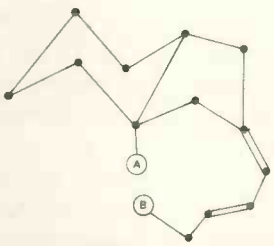


Figure 2.

Structural formula of an organic molecule. Dots represent carbon atoms, hydrogen atoms are not shown, A and B are two reactive groups, angles between bonds are fixed, rotation is possible about single but not double bonds.

containing rigid and flexible joints, construction of a model is often the only practical way to find out if two groups A and B are likely to approach each other closely enough for reaction to occur. For this particular task we represent the atoms by their centres of gravity and the bonds by rigid sticks. In other contexts, different properties of the real molecule (eg volume of the atoms) have to be represented. Since a model always consists in a *selection* of aspects of reality, the choice of a type of model best suited for its purpose is not a trivial task.

Types of models and modelling systems

Let us assume that we are again in a situation that requires a model of the human body. But this time the purpose of the experiment is entirely different: We want to do research into the dependence of blood pressure on posture and gravity in preparation for a prolonged space flight. Obviously we will end up with something quite different from the dummy in the car-crash study. The outer shape, for example, will be irrelevant for this experiment. The two models of the same original might resemble each other less than they 'resemble' the real object. Before we consider in some detail the problem of classifying

models, let us explain some structural features which seem to be common to most models.

In order to construct a satisfactory model, we have to *assign unequivocally* a physical quantity *A* occurring in the real object *O* to a corresponding quantity *A'* in the model *M*. We also *assign* a *relation* between quantities *A* and *B* (in *O*) to a corresponding relation between *A'* and *B'* (in *M*). The actual form of the assignment is only dictated by considerations of suitability, as long as it is *single-valued*.

The quantities characterising a model can themselves be described according to their function and fate (overlap between the following classes is frequent. Examples referring to our 'car-crash' are given in parentheses): *Parameters* remain constant during an experiment (size and mass of body); *input variables* represent, in the model, the external influences acting upon the object (acceleration); *variables of state* describe the momentary condition of the model (position of body in car); *output variables* are the ones we eventually want to know (deformation of body). A simulation program computes the state of the model at time t_1 from the values of the variables of state at the earlier time t_0 *and* those of the input variables at times t_0 and t_1 (since the input variables are known over the whole time interval). At the design stage, a simulation program is often represented by a network of *blocks*. Each block has input and output connections. The manner in which the variables depend on each other is given by the *transfer function* of each block. It can also be described in the form of a *frequency response* curve. Blocks without input connections are called *source blocks*. They may 'spontaneously' generate periodical or random functions of time.

By now it should be clear that, with the aid of a programming language and a digital computer, we can create an infinite variety of models. The computer (hardware plus software) constitutes a *modelling system*, the scope of which is only limited by its storage capacity. Other modelling systems are the kits of balls and sticks used to 'build' molecules, or a set of springs and weights used to demonstrate coupled oscillators in the classroom. Physicists and mathematicians, performing 'thought experiments', often dispense with material devices altogether.

It is easy to see that the computer is the most versatile and comprehensive modelling system. We all witness the increasing replacement of material models by simulation programs combined with suitable input-output devices (punched cards, typewriter, light pen, video screen, plotter . . .), yet we should at all times beware of the growing distance from sensory reality which is inherent in this development. Certainly it is technically possible to present output in the form of a three-dimensional colour picture — at a price. But nobody, as far as I know, has yet managed to convey simultaneously the touch and the smell of the object depicted . . . For some time to come, the input-output device will represent a bottleneck in the communication pathway between the human mind and the computer.

Regardless of the modelling system used, models can be classified according to the degree of resemblance (the *extent of analogy*) between real object and model. If the projected quantities are restricted to spatial coordinates and their relations are not time-dependent, we are dealing with a *structural model*. Straightforward examples are the now-familiar molecular models, but architectural or geological models belong to this class as well. Those models need not be rigid, the projection need not be linear ('distortion' is often deliberately introduced. In the case of world maps it is unavoidable.) Moreover, the term 'space' can be generalised to include spaces other than our familiar three-dimensional (euclidean) space. In sociology, economy or linguistics, for instance, the notion of 'structure' is used without reference to geometrical relationships. This does not prevent us from projecting such abstract structures into the geometrical space (eg in the case of the familiar *organigrammes*).

If time-dependent relations are introduced among the variables of a model, and input as well as output variables may include quantities other than spatial coordinates, we obtain a *behavioural* model (a dynamic model). Such models are designed to reflect as closely as possible the observable reactions of an object in its surroundings (or the events occurring in a system, which is an equivalent description). In a planetarium — a typical member of this class — the physical processes causing the movement of the 'stars' are totally different from those in the real object, the Universe, yet all the phenomena directly observable are accurately represented. 'Robots' belong to this family as well — models acting like humans, yet based on different physical processes and driven by different forces. Digital simulation programs, finally, can be thought of as generalised robots — ready to take on the character of any moving or changing thing, without a single alteration among the silicon chips carrying the 'traits' of that character.

The physical laws underlying the behaviour of a real object are often analysed in the course of constructing a simulation program. This serves to gain knowledge about the functional interdependence of the variables. Knowledge of the physical laws is not indispensable, though. *The same goal could be achieved,* albeit less efficiently, by *collecting a sufficient amount of empirical data* from the real object under varying circumstances. This second method is often followed in the case of industrial production units. 'Learning' programs exist which assimilate themselves automatically to the real object.

If, finally, there is a direct identity between certain physical processes in the real object on one hand, in the model on the other, we call the latter a *functional model*. Examples that spring to our mind are model steam engines, chemical pilot plant or sections of aeroplane wings suspended in wind tunnels. Biological *tissue cultures*, kept in carefully controlled media, and subjected to attack by bacteria and virus,

also fall into this category. Even though those bits of tissues were originally cut out of the 'real object', they are models nonetheless. Their use is based on the expectation (but not certainty) that they function in a way identical or analogous to tissue remaining in its original place in the body. The motivations for their use are similar to those mentioned in the car-crash example.

It is seldom possible to obtain a functional model simply by reducing the size of a real object. Physical quantities depend on spatial dimensions in a variety of ways. A chemical reaction, carried out in a 100ml-beaker, may well run smoothly, while the same reaction, scaled-up by a factor of 1000 in a pilot plant, may boil over or even explode. The reason here is that heat production increases with the 3rd power of the linear dimensions, while heat *loss* by radiation and conduction is proportional to the surface area (square of the linear dimensions). For the same reason, a mouse has to eat much more, in terms of its weight, than a man. — Systems containing moving fluids present even more complex problems of this type, because non-linear dependencies are encountered. If the reader wishes to catch a glimpse of the enormous amount of work which has been done in this field, he need only open a modern handbook of aerodynamics.

The examples of this section clearly show that different properties of the real object have to be considered according to the type of model we wish to construct. Which is the most suitable type in turn depends on three main categories of facts: the intended purpose of the model; the nature of the real object; and the extent of our knowledge about it. In the next section we shall discuss, in a slightly more systematic manner, the steps that lead to the construction of a model. We shall concentrate mainly on dynamic models, especially digital simulation programs.

How to make a model

As a starting point, let us assume that we are in possession of a digital computer, an appropriate programming language (eg DYNAMO, cf. Jay W. Forrester, 'Industrial Dynamics', MIT Press, Cambridge, Mass.), and that our computer is equipped with a corresponding compiler (translating program). As we shall see, there still remains a considerable amount of work to do on the way from the project to the functioning model. Let us take an extreme example: We are interested in the behaviour of an aeroplane wing during flight. Now there is so far no programming language which accepts such 'input quantities' as 'wing', 'storm', 'bend' or 'break'. We have to use our own brain to describe the problem in terms of physical quantities and functions. We shall start by setting up a suitable *coordinate system*. The choice or coordinates can very much affect the ease of subsequent operations. There is hardly an object which has not — at least in its idealised form — some type of *symmetry*. The coordinate system should be adapted to this symmetry: Cartesian coordinates for plane or cubical symmetry, polar coordinates for cylindrical or spherical situations (e.g. electric or magnetic fields, flow through tubes). Coordinate axes are best laid parallel to axes of symmetry. Many physical phenomena are represented by *vectors* (quantities which have a direction, like velocity or force). They require several numbers for complete characterisation. Morevoer, in an aeroplane wing as well as in the surrounding air, every quantity changes continuously as we go from one point in space to another. As the digital computer accepts only discrete numbers, we have to divide our space (in addition to time) into suitably small areas, within which the physical quantities are assumed constant. Each of these areas interacts with surrounding ones via one or more variables. In the following we shall not, however, consider systems with such *distributed variables*. Often a distributed variable can, by some device, be replaced by a localised variable (eg centre of gravity, moment of inertia) or else the problem is solved for a small area and later generalised for a more extended system.

A dynamic system (or sub-system) is characterised by the fact that its output variables depend not only on the *momentary values* of the input variables, but also on their *rate of change*. The rate of change is given by the first derivative with respect to time. Second or higher derivatives also occur, but they can be resolved into first derivatives. The rates of change of the output variables may in turn depend on their momentary values or on that of the input variables. Mathematically, this state of affairs is described by differential equations. Programming languages often relieve us from the task of explicitly setting up these equations, yet we still have to know the mutual dependencies of the variables and their derivatives.

If for example, a kettle with a hole in the bottom has been filled with water, the rate of water loss, expressed as the speed at which the level falls, depends linearly on the height of the level of water in the kettle (eq. 1):

$$-\frac{dw}{dt} = k \times w \qquad (1)$$

w = height of level
t = time

The *parameter* k is associated with the size of the hole. The larger the hole, the faster the level, will fall. Equation 1 is only valid if no water is being *added* to the kettle. If we open the tap and leave it at a certain position, we may call the water influx I, in pints per minute, for example. This is obviously an input variable. We may now write (eq 2):

$$\frac{dw}{dt} = -k \times w + I \qquad (2)$$

The escape from the hole tends to *lower* the level (negative term), the stream from the tap tends to *raise* it (positive term). Expressed in the programming language DYNAMO, eq 2 could look as follows:

```
R  CHLEV.KL = - CONST * LEV.K. + INFL
L  LEV.L    =   LEV.K + DT*CHLEV.KL
```

R signifies 'rate equation', L signifies 'level equation',
CHLEV arbitrarily stands for 'change of level',
.KL refers to the step from state 'K' to state 'L' of the system whereby L = K+1
LEV.K is the value of the level in the state 'K',
DT is the time interval.

The great achievement of the computer lies in the fact that we need not solve equation 2 analytically. It need not even be possible to solve it analytically. As we have already mentioned, the machine does it by dividing time into small increments *dt*, and computing the state of the system at the time $t + dt$ from the data of the previous state at time *t*. One question must now spring to everybody's mind: How are we going to *start* this chain of events? In fact, the *initial values* of the variables have to be supplied, and they can have a considerable influence on the visible behaviour of the model. The initial values are just a special case of *boundary conditions*, which are needed to specify the solution of a differential equation. Let us illustrate this fact with the aid of our leaking kettle. First let us consider the case where the inflow exactly matches the loss of water from the kettle, ie

$$k \times w = I \qquad (3)$$

From eq (2) and (3) we instantly find the logical consequence:

$$\frac{dw}{dt} = 0 \qquad (4)$$

which means that the level does not change at all. Furthermore we see that this condition is only fulfilled at a specific height of the water level in the kettles:

$$w_{ss} = \frac{I}{k} \qquad (5)$$

The subscript 'ss' refers to the *steady state*. If we start our experiment with a level w_o, lower than w_{ss}, we shall observe a gradual rise of the level up to w_{ss}, while in the opposite case the level will gradually fall to w_{ss}. These results (graphically depicted in Fig 3)

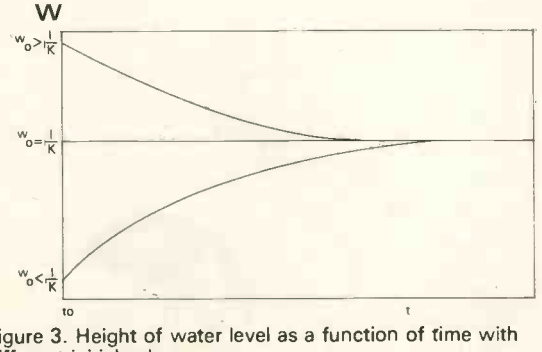


Figure 3. Height of water level as a function of time with different initial values.

would be obtained from our computer program. The plot would show steps, though, but these may be smoothed out at the output stage.

If we have dwelt on that kettle at some length, it is because it may itself serve as a model for a variety of other real systems, like a chemical reaction, or a combination of a resistor and a capacitor (Fig 4), where an equation (6) analogous to (2) can be derived:

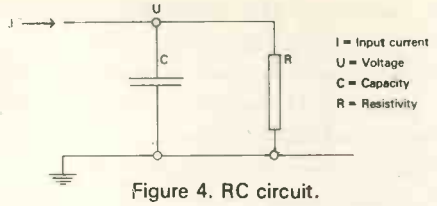


Figure 4. RC circuit.

$$\frac{dU}{dt} = -\frac{U}{RxC} + \frac{I}{C} \qquad (6)$$

It is left to the reader to work out the steady-state voltage U_{ss}. Interesting insights may be gained if, starting from the steady state, the 'tap-water' or the 'current' is suddenly turned on for a short period of time, or a sinusoidal current is fed into the circuit of Fig 3. The treatment of these problems is beyond the scope of this article. Elements similar to Fig 3 frequently occur as parts of more complex systems, be it in electronics or, by analogy, in economics or biology.

Up to this point we have tacitly assumed that all causal connections in our real objects were sufficiently well known or knowable to be represented by a *deterministic model*. We described physical quantities, which can only be measured with finite accuracy, by exact mathematical symbols. We disregarded *random fluctuations* of electrical current and water flow. Surprisingly enough, this procedure is justified in most cases, since in numerous practical systems the random fluctuations lose their effect if a time-span of sufficient length is considered. The fluctuating quantity can then be replaced by its *time-average*. If the fluctuations are, on the other extreme, very slow, the *quantity* may be regarded as constant for the duration of the experiment. Yet there are cases where the random behaviour of certain variables is very essential to the model, and where it leads to consequences which would not occur in a purely deterministic system. As an example, we need only mention the infamous phenomenon of 'bunching' on metropolitan bus networks: Whenever a bus is delayed on its route, more passengers accumulate at the stops ahead, which leads to further delays. The bus following finds the stops deserted and catches up fast with the first one . . . If traffic jams and bus queues were subject to some known law, the schedules of the buses could be adjusted accordingly. But unfortunately they have to be regarded as random processes, and bunching is a consequence of their randomness plus the presence of *feedback loops* in the system. The randomness of the traffic jams is of greater consequence than the random build-up of the queues. In a simplified model, the latter could be replaced by a monotonous function of time. It is left to the reader to work out a detailed picture of such a model, choosing suitable parameters, eg starting intervals of the buses, time between stations, delay per passenger, etc. The values of the random variables must be represented by *random numbers*. True random numbers cannot

be 'computed' by strictly mathematical means. Often this problem is solved by using so-called pseudo-random numbers. The methods to generate such numbers have developed into a science in its own right. They require a *'seed'*, a starting number from which subsequent pseudo-random numbers are produced by intricate schemes. If the left half of a multi-digit number is truncated, for example, the value of the resulting number does not depend in a predictable way on that of its predecessor, if the latter has been produced from a 'seed' by operations like squaring and multiplication by irrational numbers. An alternative method is to use natural 'noise' (eg electronic noise) as a source for random numbers.

Simulation of random processes, the *Monte-Carlo Method*, was first developed during the construction of the atom bomb. Inter-action of neutrons with matter and the ensuing fission events are only knowable in terms of probabilities, and it is not possible to represent the system by a deterministic model. As it is well known, the atom bomb is based on a *chain reaction*, ie a system with strong positive feed-back which is a kind of snowballing effect. The probabilities of the radio-active decay-events and collisions which start the reaction have to be known rather accurately in order to predict the likely outcome . . .

The general problem of judging the performance of a model is again a very complex branch of science, which we shall briefly discuss in the following section.

How good is our model?

As we have seen, any model has certain features distinguishing it from real objects:

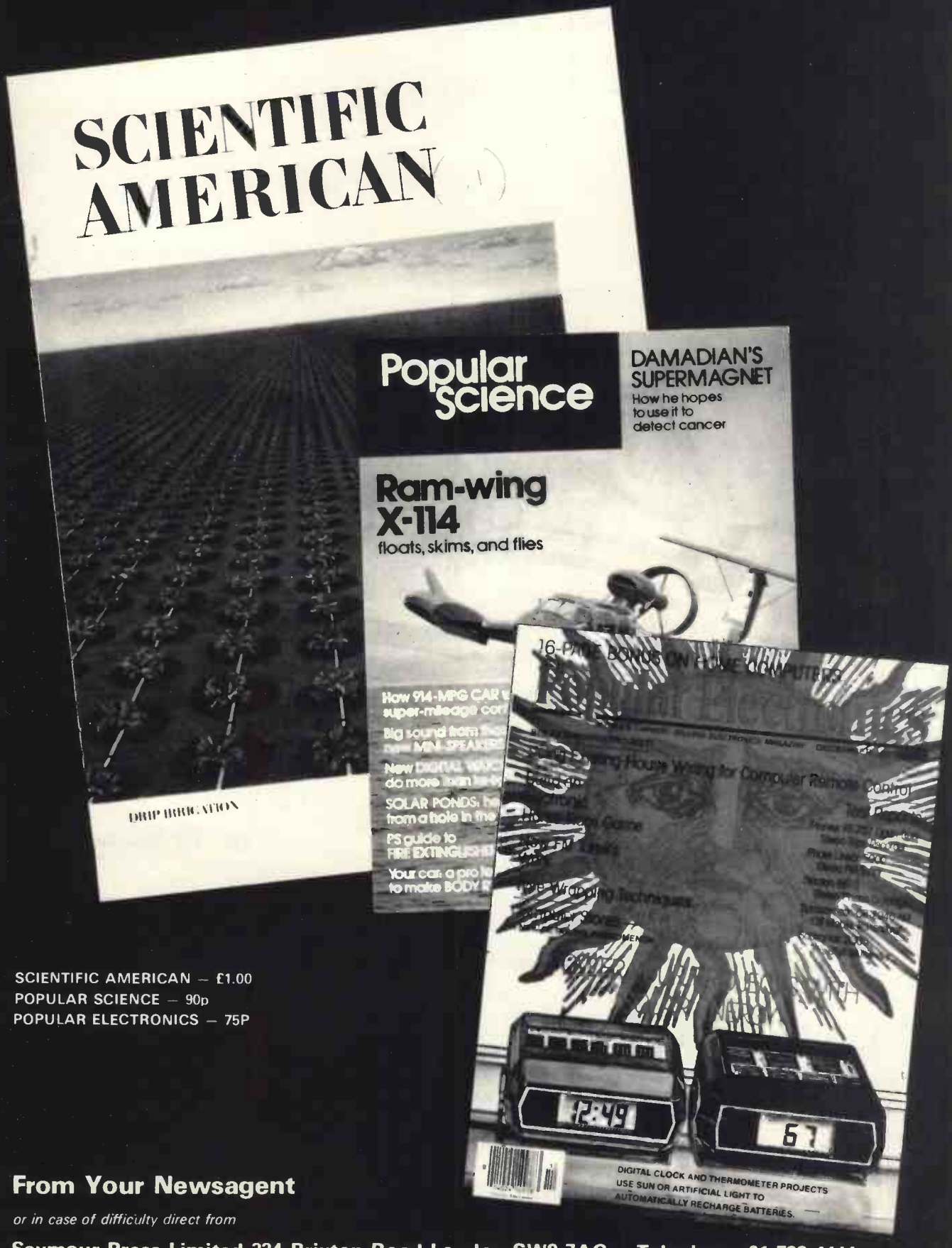
- Selective simplification of elements
- Idealised description of interactions
- Interpolation or extrapolation of situations which were never actually observed
- Representation of continuous variables by finite numbers of steps

Because of its very purpose, which is to find new facts, a model has to be based on incomplete data of limited accuracy. All these characteristic traits of a model are likely to introduce *errors, biases,* or *artifacts* of some sort. In order to find out about these, we have to *test* the model, ie to compare its performance, if possible, with that of the real object under known and accessible circumstances. Invariably we shall find some deviations between the output quantities of the model and those of the real object. Often we can adjust the parameters of the model so that the deviations are minimised. But in a moderately complex system it is by no means obvious which parameters are to be adjusted first, which ones are best left alone. *Sensitivity analysis* is concerned with the question of how the behaviour of the model is affected by small changes in one or more of the parameters. The average differences of the output variables between model and real object might be reduced, for instance, but the *shape* of the curve representing their temporal variations might become more dissimilar. Decisions will have to be taken as to which aspect is more important. If we construct a model to simulate a vibration or oscillation, the output variable has to be periodic in time and/or space. Deviations may occur in the frequency (the frequency spectrum or the waveform), the phase and the amplitude.

Another important issue is that of the *generality* of the model. Is it able to cope with all the situations which it was designed for? Are its approximations realistic in borderline cases? Is there a risk of instability? (In an *unstable* system, the variables start to grow without limit, either monotonously or in an oscillatory manner.) All those criteria depend so much on the concrete application that a general discussion would not be of much use. Last but not least, the perfection of a model, in any practical application, has to be weighed against its *cost* in terms of man-hours, equipment, delays etc. An entrepreneur will then and only then consider a model 'good' (however crude it might look to the scientist), if its use offers him some advantage over the corresponding real object or an alternative (mechanical) model. With the explosive development of cheaper and smaller computing devices, as we witness it today, there is great promise for further inroads of digital simulation into the most varied fields of pure and applied science, technology, management, economics, transport and teaching. Many problems still remain to be solved on all levels, from the reliability of input-output devices to the simplicity of software for widespread use. Yet in a world which has a growing need to rely on fast and efficient thinking in order to survive, those problems will certainly be overcome some day.

SCIENCE & TECHNOLOGY

at your fingertips...



SCIENTIFIC AMERICAN – £1.00
POPULAR SCIENCE – 90p
POPULAR ELECTRONICS – 75P

From Your Newsagent

or in case of difficulty direct from

Seymour Press Limited 334 Brixton Road London SW9 7AG Telephone 01-733 4444

Computer Workshop
Congratulates

PERSONAL COMPUTER WORLD

On Its
1st Issue.

Computer Workshop
174 Ifield Road
London SW10 9AG
01-373 8571