μP

MICROPOWER VOLUME 1, NUMBER 1

August, 1981 95 p

PROGRAM POWER PRICE LIST

ALL PROGRAMS SUPPLIED ON CASSETTE IN CUTS/KANSAS CITY FORMAT

THE SPACE SYNDROME	
Lunar Lander Supreme (16k/B/G)	£9.95
Startrek II (32k/G/B)	£9.95
Invasion Earth (MC/G)	£8.95
Invasion Earth (MC/G – sound chip)	£10.95
Alien Labyrinth (16k/B/G)	£6.95
Super Startrek (16k/B)	£6.95
Cliff Invasion (B/G)	£6.95
Space Fighter (B/G)	£5.95
SOUND OUTPUT	
AY-3-8910 Sound Chip	£6.45
60 Page Data Manual (No VAT)	£2.25
Sound Chip Interface Board	£13.50
Sound Chip Demo Program (MC)	£5.95
Audio Board and Speaker	£10.75
Music Box (16k/B)	£9.95
Road Race (MC/G)	£4.95
Cowboy Shoot Out (MC/G)	£3.95
Musical Break-out (MC/G)	£3.95
BOARD GAMES	
GAMES GRAPHIC ROM	£15.00
GAMES GRAPHICS ROM/ADAPTOR	£18.90
Sargon Chess Book	£9.50
Book with program	£19.50
Book/Prog/ROM/Adaptor	£35.00
Draughts (B/G) *	£7.95
Backgammon (16k/B/G) *	£7.95
*Please state Ordinary or ROM version	
3D Noughts & Crosses	£3.95
Tantaliser (B/G)	£3.95
Minotaur (16k/B)	£3.95
Othello (B)	£3.95
Submarine Chase (B/G)	£3.95
COTTIS BLANDFORD	
Add a Cottis Blandford cassette	
To your Nascom 1 for reliable	
And fast loading of CUTS tapes	
(NASCOM 2 format)	
Kit	£14.90

SPECIALITIES	
WORDEASE Word Processor (MC)	£25.00
Nascount Personal Finance (16k/MC)	£9.95
Club Membership (16k/MC)	£9.95
Constellation (16k/B)	£6.95
VORTEX Graphics subroutines (MC)	£8.95
Mini Toolbox (MC)	£5.95
Graph Plotter (B/G)	£4.95
Vocabulary Tutor (B)	£5.95
Renumber (MC)	£3.95
Xtal Basic 2.2 (MC)	£35.00
Nascii (B/G)	£4.95
Basic Programmers aid (B)	£4.95
Super life (MC)	£6.95
Indexed File Handler (B)	£5.95
Biorhythm (B/G)	£3.95
COMPUTER ASSISTED LEARNING	
WIRRAL PILOT v4.0 (MC)	£12.95
Butterfly (B/G)	£5.95
Reading Test Cards (B/G)	£4.95
Abacus (B/G)	£3.95
FAST INTERACTIVE GAMES	
Driver (B/G)	£4.95
Demonoes (B)	£3.95
Lumberj ack (MC)	£3.95
Slalom (B/G)	£3.95
Sheepdog Trials (B/G)	£3.95
Death Run (B/G)	£3.95
Dracula's Castle (B)	£3.95
Secret Agent (B/G)	£3.95
MISCELANEOUS	
BLACKJACK (B/G)	£5.95
Labyrinth (B/G)	£4.95
Spider (B/G)	£4.95
Mindbender (B)	£4.95
Fruit Machine (B/G)	£4.95
Beetle (B/G)	£3.95
Stockmarket (B)	£3.95
Hammurabi (B)	£3.95
Scramble (B)	£3.95
Code Breaker (B)	£3.95

B = Nascom Basic (state ROM or Tape)
MC = machine code
G = Nascom Graphics

The programs require 8K RAM unless otherwise stated

PROGRAM POWER, 5, WENSLEY ROAD, LEEDS, LS7 2LX

= - + - = - + = - + - = - + - =

CONTENTS

EDITORIAL

Why do we need a magazine for Nascom users? Firstly, there is a lot happening on the Nascom front at the moment. In the face of what appeared to be the imminent demise of the company many people who had been waiting for promised 'goodies' to appear started to produce their own add-ons. Lucas eventually came to the rescue, but the death threat had by then has its effect.

Secondly, the popular computer magazines do not give Nascom the space that Tandy, Apple, Pet, etc., etc., receive.

Thirdly, the system's users are tremendously enthusiastic. In many local computer clubs the Nascom owners are the most active group.

Finally - well, the answer might be that there is no place for the magazine. Perhaps Nascom owners are more interested in communicating with the machine than with other users. We shall soon find out, because the magazine will only survive if people - that means you - will write about their enthusiasms.

So write - say what you liked and didn't like, tell us what you want to read about, ask questions (and answer the questions of others), and above all write an article, short or long, because that is one way to ensure that the magazine contains topics that interest you.

A PROGRAMMABLE CHARACTER GENERATOR FOR NASCOM 1
by S. Hope

The main drawback of the standard ROM based graphics for Nascom is that while many programs use the standard "pixel" characters, others, for example Sargon, need special character sets to be most effective. One solution is to have two or more switch-selectable graphics ROMS, but a more effective answer is to store the data which defines the graphics characters in RAM. Any special characters required in a program can then be loaded into this RAM either from tape or by the program itself. In addition, a programmable character generator can be used to simulate bit-mapped high resolution graphics.

The unit described below is simple and cheap to build. It allows up to 128 characters to be loaded into RAM and used as normal graphics characters at any screen location. Each character consists of 16 rows of 8 dots; the data for the character is stored as 16 consecutive bytes in the P.C.G. RAM. A few modifications need to be made to the basic Nascom 1, but the unit does not need a buffer board or any other expansion to be present.

The 2K of RAM is mapped as "write only" memory at addresses £000 - £7FF, coincident with the monitor address space. This reduces the address decoding logic required, and ensures that the unit is compatible with the memory allocation of machines with different software layouts. This is very effective when the device is used purely as a character generator, but has a drawback when it is used in a program which needs to keep track of which dots on the screen are "on"; as the processor cannot read the P.C.G. RAM (reading this address space simply accesses data in the monitor) a copy of this 2K must be kept at some location in the main memory.

The unit is built on a prototyping board. Some soldering ability and considerable patience is required to hard wire all the address and data lines to the RAM. If you are short on either, or have any doubts about the modifications needed, you would probably do better to stick to one of the commercial P.C.G.'s available.

To simplify connection to the Nascom, the original 6576 character generator is resited on the new board, and a Jumper cable runs from this board to the 6576 socket on the Nascom. The state of bit 7 in the V.D.U. RAM determines whether the data for the display is obtained from the 6576 (bit 7 = 0) or from the RAM (bit 7 = 1). Because the outputs of the 6576 have no high impedance state this selection is done by means of a set of two multiplexors (74LS157, IC's 10 and 11). The 74157 is a four-pole two-way switch contolled by a voltage applied to pin 1.

The P.C.G. RAM is connected to the character and row select lines from the 6576, and the Nasbus address lines A0 - A9, through a further set of multiplexors (IC's 1 - 3). When the Nascom writes to an address in the range £000 - £7FF, these multiplexors are switched to the Nascom address lines, the chip select signal for the correct RAMs is taken low, and the write enable signal for all the 2114s is low. An 81LS95 octal buffer (IC 4) gates the data from the data bus into the 2114 I/O lines.

During the access -the data -to the V.D.U. is disabled by taking the enable lines to ICs 10 and 11 to +5 volts. A monostable, IC 16 can be used to lengthen the write pulse to reduce flashes on the screen.

The 2114 is a 1K x 4 bit static RAM, so a pair of 2114s needed for each 1K bytes. Identical address lines on each of the four RAM chips are tied together, and simlarly the eight data lines from the two pairs of chips are interconnected. When an address is accessed data is only interchanged with the pair of chips for which the CS signal is low, which is determined by the state of line 4 from the character generator socket for V.D.U. access, and by the state of A10 for processor access. The direction of the data transfer is determined by the WR signal.

The layout of the circuit is not particularly critical. The commonest problem is caused by coupling between the lines from pins 21 - 24 of the character generator socket. These lines carry signals derived from the crystal oscillator on the C.P.U. board via a divider chain. If a long ribbon cable is used to connect the unit to the main board interaction between these adjacent lines may cause screen Jitter. Fortunately the solution is simple - just separate the lines from the main cable.

MODIFICATIONS TO THE NASCOM

The Nascom character generator socket has two unconnected lines. These are used to connect bit 7 of the V.D.U. RAM to the unit and extra data output from the unit. This does not affect the operation of the Nascom if the unit is disconnected and the 6576 is replaced in its original socket. The following modifications should be made to the Nascom board (IC numbers prefixed by an N refer to the numbers used in the Nascom 1 manual).

Pin 11 of N IC 17 should be bent horizontal so that it is not in contact with its socket and connected to N IC 15 pin 1. Similarly, pin 10 of N IC 15 is bent horizontal and connected to pin 10 of the character generator socket on the Nascom. Pin 12 of N IC 20 is connected to N IC 17 pin 18 and pin 19 of N IC 17 is conected to pin 14 of the character generator socket. The connections to N IC 15 are necessary because in the standard Nascom 1 V.D.U. the last bit in each line of the characters is set to zero; this leaves a gap between each character, which is not wanted in the case of graphics characters. Please note that pin 11 of N IC 17 and pin 10 of N IC 15 are the only pins which should be bent (this is easier than tracing tracks and then cutting the wrong one!).

The unit is designed specifically for a Nascom 1. It should not be difficult to adapt it for a Nascom 2, because you already have a socket decoded for a graphics ROM. I have not tried adapting the circuit because none of my friends with Nascom 2's will let me perform the necessary surgery!

*** * * * * * * * * * * * * * ***

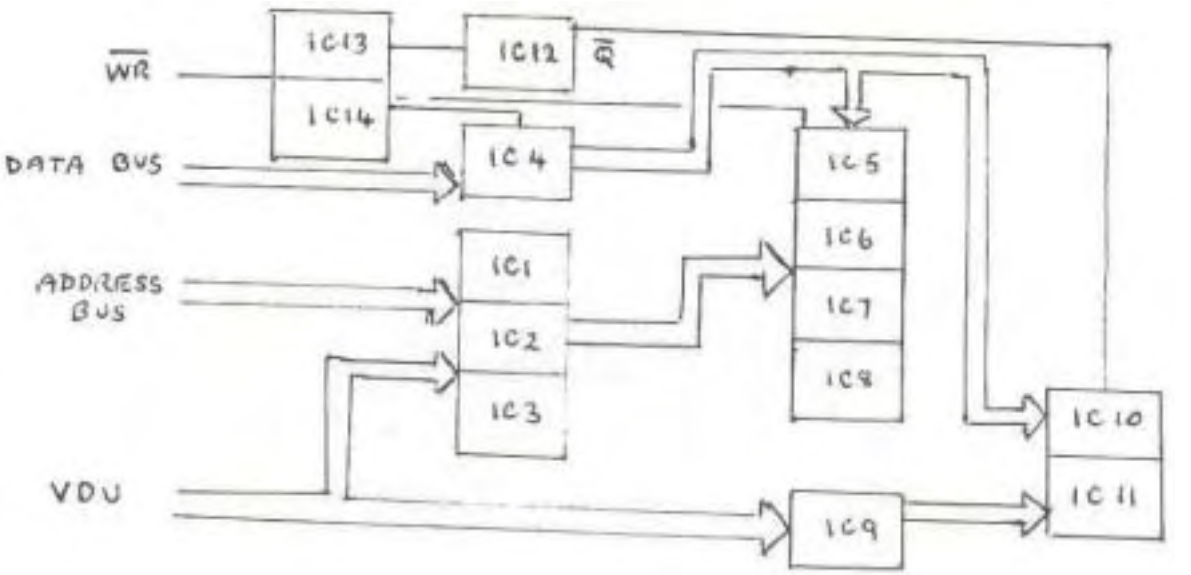
Chips used:-

74LS157	5 off	IC 1,2,3,10,11
81LS95 or 81LS97	1 off	IC4
2114	4 off	IC 5,6,7,8
6576	1 off	IC 9 (from Nascom board)
74LS121	1 off	IC 12
74LS00	2 off	IC 13,14

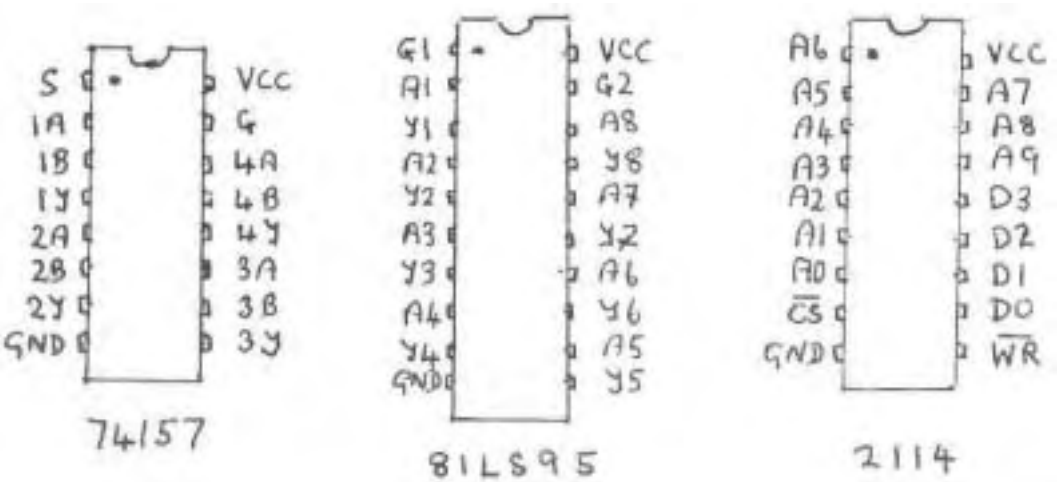
Connections to the Nascom 1:-

24 lines from the character generator socket.
Ten address lines (A0 –A9)
Eight data lines from the expansion socket
The $\overline{WR}$ line
Connections from pins 4 and 5 of N IC 36a.

Block Diagram



Pin Connections



LETTERS

IS THERE ANYBODY OUT THERE?

Dear Sir,

I am very interested in you new magazine from a club/school point of view. However, I do not seem to be able to contact any user groups or computer clubs in my area, and it is not through want of trying. There must be some Nascom Novices keen to form a user group in the Berkshire area,

Yours faithfully,

Mike Rothery,
37, Eton Wick Road,
Windsor

HELP WANTED

Dear Sir,

I have a Nascom 2, I have read the manual, I have used PEEK and POKE in programs - BUT - what am I doing? So PEEK looks at memory and POKE modifies memory; why am I PEEKing and POKEing a particular number, what is the significance of that number, what number do I need to poke for a specific task, how do I arrive at the correct number, and, finally, why isn't there an index to such things?

Yours faithfully,

D. C. Killick,
Cardiff

* Alright, if that is what you want we will provide it.

WRITE NOW

Dear Sir,

I would like to see the magazines:-

a) Include information on the compatibility of the many add-ons that are now available from Nascom, Gemini, Microvalue, Interface, etc.

b) Publish listings of the software from Gemini, etc., as many products do not come with listings.

c) Publish an article on the history of Nascom

All the best with the new mag - I would rather have my Nascom than one of those American or Japanese plastic boxes any day,

Yours Sincerely,

C. F. A. Waller,
Dunstable

* To provide a full listing with a product would add quite a lot to the cost and I don't think firms would be very happy to see their software published!

MORE IDEAS

Dear Sir,

In general I would like to read articles on the following topics;

1) Hardware problems that people have experienced, and their solutions.
2) Any special hardware interfacing.
3) Interesting and unusual programs.
4) Useful routines and clever solutions that have been devised to overcome some of the machines shortcomings.

Yours faithfully,

A.J. Littlewood,
Langley

* I didn't know it had any!

SOFTWARE FOR PROGRAMMABLE GRAPHICS
by J.Haigh

A character generator is merely a memory device which stores data defining which points within the character area are 'on'. In the Nascom each character consists of 16 lines of 8 dots and the data is stored as 16 consecutive bytes. When the V.D.U. circuitry is displaying a particular character it looks at the data stored in the generator, selecting the byte appropriate to the current line. This byte is passed to a device which looks at each bit in turn; if the bit is a '1' the intensity of the beam scanning the T.V. screen is intensified and a dot appears.

In the Nascom 2 the data in the character generator is displayed with the least significant bit on the right of the character, while the system fitted to my Nascom 1, which is a combination of Steven Hope's P.C.G. and an early Bits and P.C.s graphics board, displays the bits in the opposite order. In the software discussed below I shall stick to the Nascom 2 display order, but indicate how to change the programs for non-standard machines. A further incompatibilty is caused by the fact that the Nascom 2 only displays 14 lines per character. However, a simple modification has been published in the Liverpool Software Gazette which produces the full 16 line display.

The simplest way to use a programmable character generator is to store sets of characters on tape and load each set as it is required, but this soon becomes tedious. Special characters can be incorporated into the program that uses them and written to the P.C.G. either by a copy routine, for a machine code program, or by a READ, POKE loop in Basic. However, character A0 and the pixel set (C0 - FF) merit individual attention. The Zeap assembler uses character A0 to mark the end of a line. In the standard Nascom 2 graphics set this character is identical to the space character and so is not noticed on the screen; if A0 is not clear, as it will not be if you have Just switched on or have been using special characters, the Zeap display is very untidy. A similar problem is found with Nas Debug, which uses character C0 as a separator. Of course, you can always clear these characters directly by a modify command, but a better solution is to have a short routine in EPROM which clears A0 and writes the standard pixel set from C0 to FF. A listing of such a routine is given below.

It is followed by an even simpler program which writes the TRS80 pixel set into the Nascom P.C.G. This is quite useful if you are trying to adapt a program which uses TRS80 graphics to run on your Nascom. If your machine displays the generator bits in inverse order you will have to change line 270 to LD A 15 and line 320 to ADD A £F0; similarly, the values in lines 540 and 580 should be interchanged. On a Nascom 1 the upper 4 pixels consist of 5 lines of 4 dots, while the bottom 2 contain 6 lines. As an unmodified Nascom 2 misses out the two bottom lines, in this case the lowest pixels are 4 x 4.

You will soon want to invent your own characters, either to be used singly or in blocks - you can get very impressive high-resolution pictures with 128 graphics characters in a 16x8 block. The difficulty is working out what data to put in the P.C.G. RAM. One method is to draw the characters on graph paper and then convert the diagrams to bytes. The third program below can be used to draw characters directly from the keyboard. The program is executed by entering E1000 AAAA, where AAAA is the address of your P.C.G. RAM or, if your P.C.G. address is coincident with an area of EPROM as in Steve Hope's design, an area of

free RAM where a copy of the P.C.G. data can be maintained. The characters are shown on the right of the screen on a large scale, each bit being represented by two pixels. The pixels are turned on and off by the eight leftmost alphanumeric keys in the four rows of the keyboard (1-8, Q-I, A-K and Z-,). An arrow indicates which set of four lines is currently selected; keys 0, P, ;, and / move the arrow so that all 16 lines can be modified. The character being defined appears at the cursor, which can be moved around the left half of the screen by means of the cursor keys. This character can be left at any position by pressing key -, and removed by = (shift/-); you can thus build up blocks of characters to draw a complex diagram.

At the bottom of the screen the program lists the hex value of the character being defined and the 16 bytes of data in that character in hex (for use in machine code programs) and in decimal (for use in Basic). You can step forwards or backwards through the character set with the N/L and BS keys. When you first select a character it will probably contain unwanted data; this can be erased by CONTROL/C. Because the program uses the pixel set to produce the large-scale diagram of the character only 64 characters can be defined at one time; the program will thus work with a P.C.G. system which has only 1K of RAM and keeps the pixel set permanently in EPROM. The modifications for inverse display are:- change (1055) to £0F, (105F) to £F0. (124A) to £01, (1250) to 02, and (12A9), (12B3), (12BD), (12D9) to 0E. If your P.C.G. RAM is coincident with a block of ROM, its address should be placed at £12EC, £12ED.

The fourth program is included as a demonstration of what can be produced by programming a block of characters. It displays a picture of the space shuttle which moves smoothly across the screen. Most of the program consists of a data table which is copied to the P.C.G. RAM to produce the intitial image. This is then moved by rotating each byte of the graphics characters one bit at a time. If you wish to convert this program for an “inverse order” display you will have to change the byte at £135F to £16, and invert the data table. I have included a routine at £13C0 which performs this inversion.

Finally, there are two short programs to demonstrate plotting points from Basic. The first program plots the orbit of a satellite around two primaries, one visible and the other invisible. The second shows how to produce simple graphs in Basic. Both routines set the points directly from Basic and could obviously be speeded up by using machine code accessed by a USR call. With a 128 character P.C.G. you have just enough characters to plot a sine curve and the axes at full-screen :Size. If you try to plot multiple curves you will find that you run out of characters - for such displays you need bit-mapped graphics. However, it is surprising what effective displays you can obtain with a simple P.C.G.; a life program is most impressive on a 384 x 240 array. Now that several commercial graphics units are available I hope that software will be produced which makes full use of the high resolution possible.

****

```
1000            0010            ORG     £1000       ;  Nascom 2 pixel set
1000  210000    0020            LD      HL 0        ;  Put your P.C.G. address here
1003  110002    0030            LD      DE £200     ;  Offset to character £A0
1006  19        0040            ADD     HL DE
1007  E5        0050            PUSH    HL          ;  Save HL
1008  0610      0060            LD      B 16        ;  Sixteen bytes to be cleared
100A  3600      0070  CLRAO     LD      (HL) 0      ;  Set all bytes to zero
100C  23        0080            INC     HL
100D  10FB      0090            DJNZ    CLRA0
100F  E1        0100            POP     HL          ;  Recover HL
1010  19        0110            ADD     HL DE       ;  Now go to character £C0
1011  0EC0      0120            LD      C  £C0          ; Character is kept in C
1013  C5        0130  SBC       PUSH    BC          ;  Save BC
1014  D70B      0140            RCAL    PIX5        ;  Set 5 top bytes as necessa
1016  D709      0150            RCAL    PIX5        ;  Set next 5 bytes
1018  0606      0160            LD      B 6         ;  Set bottom 6 bytes
101A  D707      0170            RCAL    PIXEL
101C  C1        0180            POP     BC          ;  Recover BC
101D  0C        0190            INC     C           ;  Next character
101E  20F3      0200            JR      NZ SBC      ;  Continue until zero
1020  C9        0210            RET
1021  0605      0220  PIX5      LD      B 5         ;  Routine to set 5 bytes
1023  AF        0230  PIXEL     XOR     A           ;  Clear A
1024  CB09      0240            RRC     C           ;  Test bits 0,1 or 2
1026  C5        0250            PUSH    BC
1027  3002      0260            JR      NC RRC3     ;  Jump if bit = 0
1029  3EF0      0270            LD      A £F0       ;  Draw pixel if bit = 1
102B  CB09      0280  RRC3      RRC     C           ;  Now test bits 3,4 or 5
102D  CB09      0290            RRC     C
102F  CB09      0300            RRC     C
1031  3002      0310            JR      NC LDHLA    ;  Jump if bit = 0
1033  C60F      0320            ADD     A 15        ;  Draw pixel if bit = 1
1035  77        0330  LDHLA     LD      (HL) A      ;  Set number of lines
1036  23        0340            INC     HL          ;  Stored in B
1037  10FC      0350            DJNZ    LDHLA
1039  C1        0360            POP     BC          ;  Recover BC
103A  C9        0370            RET
1000            0380            ORG     £1000       ;  TRS80 pixel set
1000  0EC0      0390            LD      C £C0
1002  210000    0400            LD      HL 0        ;  Put your P.C.G. address here
1005  C5        0410  TPUSH     PUSH    BC
1006  D70B      0420            RCAL    TRSP5       ;  Two sets of 5 lines
1008  D709      0430            RCAL    TRSP5
100A  0606      0440            LD      B 6
100C  D707      0450            RCAL    TRSPIX      ;  One set of six lines
100E  C1        0460            POP     BC
100F  0C        0470            INC     C
1010  20F3      0480            JR      NZ TPUSH
1012  C9        0490            RET
1013  0605      0500  TRSP5     LD      B 5
1015  AF        0510  TRSPIX    XOR     A
1016  CB09      0520            RRC     C           ;  Test bits 0, 2, or 4
1018  3002      0530            JR      NC TRRC
101A  3EF0      0540            LD      A £F0       ;  Draw pixel if bit = 1
101C  CB09      0550  TRRC      RRC     C           ;  Test bits 1, 3, or 5
101E  C5        0560            PUSH    BC
101F  3002      0570            JR      NC TRR2
1021  C60F      0580            ADD     A 15        ;  Draw pixel if bit = 1
1023  77        0590  TRR2      LD      (HL) A      ;  Set number of lines
1024  23        0600            INC     HL          ;  Stored in B
1025  10FC      0610            DJNZ    TRR2
1027  C1        0620            POP     BC
1028  C9        0630            RET
```

T1000 1300 0 8 1

1000	C3	83	10	31	32	33	34	35	36	37	38	51	57	45	52	54
1010	59	55	49	41	53	44	46	47	48	4A	4B	5A	58	43	56	42
1020	4E	4D	2C	74	65	64	20	11	10	27	CD	40	10	11	E8	03
1030	CD	40	10	11	64	00	CD	40	10	1E	0A	CD	40	10	1E	01
1040	AF	3C	ED	52	30	FB	3D	19	F6	30	F7	C9	06	05	AF	CB
1050	09	C5	30	02	3E	F0	CB	09	CB	09	CB	09	30	02	F6	0F
1060	77	FD	77	00	FD	23	23	10	F7	C1	C9	3A	23	10	26	00
1070	CB	BF	07	17	CB	14	17	CB	14	17	CB	14	6F	ED	5B	25
1080	10	19	C9	EF	0C	00	3A	0B	0C	FE	02	28	32	EF	0D	59
1090	6F	75	20	68	61	76	65	20	6E	6F	74	20	65	6E	74	65
10A0	72	65	64	20	74	68	65	20	50	2E	43	2E	47	2E	20	52
10B0	41	4D	20	61	64	64	72	65	73	73	2E	0D	00	DF	5B	2A
10C0	0E	0C	22	25	10	3E	20	32	24	10	3E	80	32	23	10	DD
10D0	21	00	00	FD	2A	25	10	11	00	04	FD	19	EB	0E	40	C5
10E0	CD	4C	10	04	CD	4E	10	CD	4C	10	C1	0C	F2	DF	10	EF
10F0	0C	00	21	6A	08	36	E0	06	08	2C	36	E4	10	FB	2C	36
1100	C4	06	05	11	40	00	19	36	C7	10	FB	19	36	C3	06	08
1110	2D	36	D2	10	FB	2D	36	D8	06	05	ED	52	36	F8	10	FA
1120	3E	0D	32	B4	08	3A	23	10	2A	29	0C	77	47	E5	21	6A
1130	0A	22	29	0C	EF	43	68	61	72	2E	20	00	78	DF	68	CD
1140	6B	10	EB	21	CA	0A	22	29	0C	21	10	00	19	EB	E5	DF
1150	54	21	4A	0B	22	29	0C	E1	06	04	CD	66	11	DF	6A	06
1160	04	CD	66	11	18	27	EF	20	00	5E	23	56	23	E5	21	00
1170	80	A7	ED	52	30	0A	EF	2D	00	67	6F	A7	ED	52	18	04
1180	EB	EF	20	00	CD	27	10	DF	69	E1	10	DD	C9	E1	22	29
1190	0C	DF	7B	11	FF	FF	FE	11	28	15	11	01	00	FE	12	28
11A0	0E	11	C0	FF	FE	13	28	07	11	40	00	FE	14	20	34	2A
11B0	29	0C	3A	24	10	77	19	7C	FE	07	20	04	26	09	18	19
11C0	FE	0A	20	04	26	08	18	11	7D	E6	3F	FE	2A	20	06	7D
11D0	EE	20	6F	18	04	FE	09	28	F6	7E	32	24	10	3A	23	10
11E0	77	18	AB	FE	03	20	0D	CD	6B	10	06	10	36	00	23	10
11F0	FB	C3	94	12	FE	30	20	23	DD	21	00	00	1E	00	16	00
1200	21	B4	08	01	40	00	36	20	09	09	36	20	09	36	20	09
1210	36	20	21	B4	08	19	36	0D	C3	91	11	FE	50	20	08	DD
1220	21	04	00	1E	80	18	D7	FE	3B	20	08	1E	C0	DD	21	08
1230	00	18	CB	FE	2F	20	09	DD	21	0C	00	11	00	01	18	C0
1240	21	03	10	0E	04	1E	00	06	08	16	80	BE	28	37	23	CB
1250	0A	10	F8	1C	0D	20	F0	FE	2D	20	09	3A	23	10	32	24
1260	10	C3	25	11	FE	3D	20	04	3E	20	18	F2	FE	08	20	0B
1270	3A	23	10	3D	CB	FF	32	23	10	18	19	FE	0D	20	E2	3A
1280	23	10	3C	18	EF	D5	CD	6B	10	D1	7A	16	00	19	DD	E5
1290	D1	19	AE	77	CD	6B	10	E5	FD	E1	21	AB	08	11	38	00
12A0	0E	05	06	08	3E	C0	FD	CB	00	06	30	04	CB	C7	CB	DF
12B0	FD	CB	01	06	30	04	CB	CF	CB	E7	FD	CB	02	06	30	04
12C0	CB	D7	CB	EF	77	23	10	DC	FD	23	FD	23	FD	23	19	0D
12D0	20	D0	06	08	3E	C0	FD	CB	00	06	30	04	CB	C7	CB	DF
12E0	CB	CF	CB	E7	77	23	10	EC	2A	25	10	11	00	00	01	00
12F0	04	ED	B0	C3	25	11	6E	20	74	68	61	74	20	63	68	61

T1000 13D8 0 8 1

1000	C3	06	13	4C	44	01	03	00	03	03	03	03	03	03	03	78
1010	7F	FF	FF	FF	7F	F8	FC	0E	EF	EF	EF	EF	EF	EF	EF	0F
1020	C0	FF	FF	FF	FF	00	00	00	00	80	C0	E0	F0	F8	FC	FE
1030	FF	3F	CF	F7	F7	00	00	00	00	00	00	00	00	00	00	00
1040	00	80	C0	E0	F8	00	00	00	00	00	00	00	00	00	00	00
1050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
1060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
1070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
1080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
1090	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
10A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
10B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
10C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
10D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
10E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
10F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
1100	00	00	00	00	00	FF	FF	FF	7F	FF	FF	FF	7F	78	02	1E
1110	7E	1E	00	1E	7E	FF	FF	FF	FF	FF	FF	FC	C3	3F	FF	FF
1120	FF	FF	FF	FF	FF	F8	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
1130	FF	FF	FF	FF	FF	00	FF	FF	FF	FF	FF	FF	FF	FF	FF	FF
1140	FF	FF	FF	FF	FF	00	FF	FF	FF	FF	FF	DD	CD	D5	D5	D9
1150	FF	FF	FF	FF	FF	00	FC	FF	FF	FF	FF	F3	ED	E1	ED	ED
1160	FF	FF	FF	FF	FF	00	00	FF	FF	FF	FF	F1	F7	F3	FD	F1
1170	FF	FF	FF	FF	FF	00	00	C0	FF	FF	FF	F9	F6	F0	F6	F6
1180	FF	FF	FF	FF	FF	00	00	00	FF	FF	FF	FF	FF	FF	FF	FF
1190	FF	FF	FF	FF	FF	00	00	00	FF	FF	FF	FF	FF	FF	FF	FF
11A0	FF	FF	FF	FF	FF	00	00	00	00	FF	FF	FF	FF	FF	FF	FF
11B0	FF	FF	FF	FF	FF	00	00	00	00	FF	FF	FC	FC	FE	FF	FF
11C0	FF	FF	FF	FF	FF	00	00	00	00	F0	FC	07	31	79	31	FF
11D0	FF	FF	FF	FF	FF	00	00	00	00	00	00	00	00	00	80	F0
11E0	FE	FF	FF	FF	FF	00	00	00	00	00	00	00	00	00	00	00
11F0	00	70	7C	7F	7F	00	00	00	00	00	00	00	00	00	00	00
1200	00	00	00	00	00	1E	00	00	00	00	00	00	00	00	00	00
1210	00	00	00	00	00	FF	FF	00	00	00	00	00	00	00	00	00
1220	00	00	00	00	00	FF	FF	00	00	00	00	00	00	00	00	00
1230	00	00	00	00	00	FF	FF	00	00	00	00	00	00	00	01	03
1240	07	0F	1F	3F	00	FF	FF	00	03	07	0F	1F	3F	7F	FF	FF
1250	FF	FF	FF	FF	00	FF	FF	55	FF	FF	FF	FF	FF	FF	FF	FF
1260	FF	FF	FF	F0	00	FF	D5	FF	FF	FF	FF	FF	FF	FF	FF	FF
1270	FF	FE	E0	00	00	55	FF	FF	FF	FF	FF	FF	FF	FF	FF	FE
1280	E0	00	00	00	00	AA	FF	FF	FF	FF	FF	FF	FF	FF	E0	00
1290	00	00	00	00	00	AA	FF	FF	FF	FF	FF	FF	FC	00	00	00
12A0	00	00	00	00	00	54	FF	FF	FF	FF	FF	E0	00	00	00	00
12B0	00	00	00	00	00	28	FF	FF	FF	F8	00	00	00	00	00	00
12C0	00	00	00	00	00	7F	9F	DF	80	00	00	00	00	00	00	00
12D0	00	00	00	00	00	FF	FF	FF	00	00	00	00	00	00	00	00
12E0	00	00	00	00	00	7F	7E	70	00	00	00	00	00	00	00	00
12F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
1300	00	00	00	00	00	00	EF	0C	00	3A	0B	0C	FE	02	28	0B
1310	EF	41	72	67	2E	20	00	DF	6B	DF	5B	21	0A	09	22	03
1320	10	36	8F	11	40	00	19	36	9F	19	36	AF	19	36	BF	21
1330	05	10	ED	5B	0E	0C	01	00	03	ED	B0	EB	36	00	23	10
1340	FB	2A	0E	0C	11	00	00	01	00	04	ED	B0	DD	2E	08	DD
1350	26	04	2A	0E	0C	11	10	00	0E	10	E5	06	10	A7	CB	1E
1360	F5	FF	19	F1	10	F8	E1	23	0D	20	EF	11	F0	00	19	DD
1370	25	20	E2	2A	0E	0C	11	00	00	01	00	04	ED	B0	DD	2D
1380	20	CD	21	38	09	11	40	00	E5	CD	A2	13	E1	19	CB	4C
1390	28	F6	ED	5B	0E	0C	21	10	00	19	01	00	03	ED	B0	C3
13A0	3B	13	D5	E5	D1	1C	01	2E	00	ED	B8	7E	12	D1	E6	0F
13B0	20	02	36	21	35	C9	20	61	6E	64	20	72	65	6D	6F	76
13C0	21	05	10	11	05	13	06	08	CB	06	1F	10	FB	77	23	A7
13D0	ED	52	19	38	F1	DF	5B	20	63	61	6E	20	74	68	75	73

```
]LIST

1 REM: “ORBITS” BY S. HOPE.
5 LT=1:G=500:CLS:SC=3:CH=129
10 DIMS(LT),V(LT),A(LT),SN(LT),VN(LT),D(7),CH(2047)
20 FORI=0TOLT:INPUT”CO-ORDS FOR START (S,V) “;S(I),V(I):NEXT
30 INPUT”GIVE 2ND MASS COORDS”;M(0),M(1)
40 CLS:FORI=0TO7:READD(I):NEXT:DATA128,64,32,16,8,4,2,1
45 FORI=1TO40:X=48*RND(1):Y=15*RND(1):PRINT@X,Y,”.”;:NEXT
50 FORI=0TO31:DOKEI,0:NEXT:FORI=13TO17:READA:POKEI,A:CH(I)=A:NEXT
60 DATA28,62,62,62,28
70 PRINT@22,9,CHR$(128);@ 22,10,CHR$(129);
100 GOSUB1000:X=S(0)/SC:Y=S(1)/SC
102 J=2592+X/8-INT(Y/16)*64:IFPEEK(J)>100THENK=(PEEK(J)-128)*16:GOTO110
104 CH=CH+1:IFCH>255THENPRINT@0,0,”NO MORE CHARS”:STOP
105 K=16*(CH-128):FORI=KTO14+KSTEP2:DOKEI,0:NEXT
107 IFPEEK(J)=46THENDOKEK+7,6168
108 POKEJ,CH:PRINT@42,0,255-CH;
110 A=KBD:IFA<>THENTL=A-48
115 L=K+15-(YAND15):CH(L)=CH(L)ORD(XAND7):POKEL,CH(L):GOTO100
1000 R=0:R1=0:FORI=0TOLT:VN(I)=V(I)+A(I):SN(I)=S(I)+V(I)+A(I)/2
1002 X(SN(I)+S(I))/2:R=X*X+R:Y=X-M(I):R1=R1+Y*Y
1003 NEXT:IFR<1ORR1<1THENPRINT@0,0,”BANG”;:STOP
1010 A=-G/R/SQR(R):A1=-G/R1/SQR(R1)
1015 FORI=0TOLT:A(I)=A*(S(I)+SN(I))/2+A1*((S(I)+SN(I))/2-M(I))
1020 V(I)=(VN(I)+V(I)+A(I))/2:S(I)=(SN(I)+S(I)+V(I)+A(I)/2)/2:NEXT:RETURN
Ok

LIST

5 REM: A PROGRAM TO PLOT SIMPLE GRAPHS
10 CLEAR:DIMCH(2047),DT(7)
20 DATA 128,64,32,16,8,4,2,1
30 FOR N=0 TO 7:READ DT(N):NEXT
40 GOSUB 900
45 REM: THE SINE CURVE
50 FORX=0 TO 383:Y=120+100*SIN((X-191)/60)
60 GOSUB 1000:NEXT
70 FORT=1TO2000:NEXT
80 GOSUB900:FORX=0TO383:XS=(X-191.5)/30
85 REM: THE GRAPH OF SIN(X)/X
90 Y=120+100*SIN(XS)/XS
100 GOSUB 1000:NEXT
800 END
900 CLS:CR=127:FORX=0TO383:Y=120:GOSUB1000:NEXT
910 FORY=0TO239:X=191:GOSUB1000:NEXT:RETURN
1000 X1=INT(X/8):X2=X-8*X1
1005 Y1=INT(Y/16):Y2=Y-16*Y1
1010 M=2954+X1-64*Y1:C=(PEEK(M)-128)*16
1020 IF C>0 THEN 1060
1030 CR=CR+1:IFCR>255THEN RETURN
1040 C=16*(CR-128):POKEM,CR
1050 FORK=CTOC+15:POKEK,0:CH(K)=0:NEXT
1060 N=C+15-Y2:CH(N)=CH(N)ORDT=(X2)POKEN,CH(N)
1070 RETURN
OK
```

PRINTERFACE
THE EPSON MX-80 F/T
by P. Whittaker

The Epson MX-80 is a nine-wire dot matrix printer which as supplied has a standard Centronics 8 bit parallel interface. This can be connected directly to a Nascom PIO, so all that you need to run the printer is the software to interface it to the output ports. The first requirement is an initialisation routine to set the eight bits of one port to output lines, for transfer of data to the printer, and to configure the other port for control of the transfer. The simplest way to synchronise the machines only needs the "busy' line from the printer, which is low whenever the printer is able to receive data, and a strobe signal from the computer, which is taken low to tell the printer that data is to be read in.

In the standard Nascom port A of the PIO is addressed as port 4, and the control port which specifies its configuration is port 6; similarly, data for port B is output to port 5, and control words to port 7. A suitable PIO initialisation routine is thus:-

```
        LD A, £CF       Control word to specify mode 3
        OUT (6) , A     Send this to port 6
        XOR A           Next byte sent to port 6 specifies which
        OUT (6), A      lines are outputs (0), which inputs (1)
        LD A, £CF       Now set port B to mode 3
        OUT (7), A
        LD A, 1         Use bit 0 as input (for 'busy' line)
        OUT (7), A
        LD A, 2         Set strobe line high
        OUT (5), A
```

If you are using Nas-sys, the initialisation routine should end by storing the address of the printer output program at £0C78, so that when the user I/O is activated with command U output will be routed to the printer. With the earlier monitors the corresponding address is £C4B, but the output program will have to call the CRT routine. After a reset the output address will have to be reinserted, but the PIO will not need to be initialised again.

The output routine has to test the 'busy' line before sending a character to the printer; as soon as this line is low, the character is output at port A and the strobe line is pulled to zero volts.

```
        PUSH AF         Save the character to be printed
TSTO    IN A, (5)       Read port B
        RRCA            Rotate bit 0 to carry flag
        JR C TSTO       Loop while bit 0 is high
        POP AF          Recover character
        PUSH AF         Resave
        OUT (4), A      Output to printer
        XOR A           Strobe signal to printer
        OUT (5), A
        LD A, 2         Reset strobe high
        OUT (5), A
        POP AF          Recover character
        RET             End of routine
```

You will now be able to get hard copy of machine code tabulations, Basic listings, and source and object code from assemblers and disassemblers. If you use the printer with Nas-Dis you will come across a strange problem - the machine goes into double-print mode. The reason for this is that after accepting the user options Nas-Dis uses an ESCAPE (£1B) to get to the beginning of a line, and then prompt "Go?". Unfortunately the printer interprets ESCAPE/G as an instruction to double print. The only cures are to modify Nas-Dis (for example, change to a lower case g), or to use a special printer output routine for Nas-Dis which ignores ESCAPE/G.

If you try to use the pixel set in the Epson you will find that it is not compatible with the Nascom pixels; in fact, it is the TRS 80 pixel set, using characters £80 - £BF or £A0 - £DF (switch selectable). The following section of code carries out the necessary conversion. The effect of each step in the process is listed on the right. An X indicates that the value is not a copy of a specific original bit.

	Operation	Current location of Original bits	Carry Flag	Zero Flag
	RRCA	07654321	0	X
	AND A	07654321	X	X
	RRA	X0765432	1	X
	RRA	1X076543	2	X
	RRA	21X07654	3	X
	JR NC, ANDA			
	SET 5,A	21307654	3	X
ANDA	AND A	21307654	X	X
	RRA	X2130765	4	X
	BIT 6,A	X2130765	4	2
	JR Z, RES6			
	SET 7, A	22130765	4	2
RES6	RES 6, A	2X130765	4	2
	JR NC, RRCA			
	SET 6, A	24130765	4	2
RRCA	RRCA	52413076	5	X
	RRCA	65241307	6	X
	RRCA	76524130	7	X

This completes the bit. manipulation; SUB £20 now shifts the characters to £A0 - £DF, SUB £40 converts them to £80 - £BF.

The graphics ROM in the Epson also contains 64 Japanese characters, which can be accessed when pin 7 of DIP switch 1, at the back of the printer PCB, is set to ON. These characters are coincident with the pixels. If you send control code ESCAPE/5 (i.e., £1B £35 in hex., 27 53 in decimal) subsequent characters in the graphics range will produce the Japanese set; ESCAPE/4 selects the pixel set. Now, you may not want to print in Japanese, but a bit of research into the software inside the machine shows that you could easily reprogram these 64 characters to your own design.

At a first glance at the beautifully made P.C.B. there appear to be some chips missing. There are three 24 pin sockets near the back edge of the board labelled 2716, but only one of these contains a chip, which is a 2332! However, this 4K ROM can be removed and replaced by two 2716s without any modifications to the board.

The first 2K is part of the operating system of the printer (the rest of this is located in a ROM contained in the 8049 processor chip which controls the printer); the second 2K contains the data for the characters. The character is defined by 9 bytes in which. Just to be awkward, a 1 means leave a space and a 0 means print a dot; each bit controls one of the matrix needles. So, what about the ninth needle - after all the adverts make great play of the fact that this is a 9x9 printer. Well in fact the ninth needle is only used for letters with descenders (g,p,q,y); the bytes which define these letters are applied to the bottom eight needles. The letter J uses the eighth needle, and all the other ASCII characters use only the top seven.

The pixels are not stored in the graphics section of the ROM, but are printed, as blocks of 3 x 4 dots, by a special routine in the operating system. To print pixels the machine does two passes per line, and also prints in one direction only, so it is four times as slow as the bidirectional printing of ASCII characters. If you program your own characters into the machine, you can access them as codes £80 - £BF and design your output routine so that when it receives a character in this range it prefixes ESCAPE/5, while a pixel character is prefixed by ESCAPE/4; both characters should then be converted to the range that the printer is selected to accept.

The printer has quite a range of control codes with which you can select the print size, print density, line spacing, form length, and horizontal and vertical tabs. Using pixels in the 'condensed print' mode you have a resolution of 264 points across the 8 inch printing width of the Epson. However, there is a high-resolution graphics version of this printer - the MX80 F/T 2. The software for this version is contained in three 2716s, the third one replacing the 8049's internal ROM! To do this a single wire link is cut on the P.C.B., which pulls pin 7 of the 8049, the External Access pin, to +5v. This forces the 8049 to ignore the internal ROM and read program data from an external device. The F/T 2 has normal density graphics, with a resolution of 480 dots/line, and double density, 960 dots/line. When printing in high-resolution mode each bit of the data received controls one of the top eight needles, with the most significant bit uppermost.

The F/T 2 has a couple of features which I would like to see on the F/T 1; it deletes single characters on receipt of a Backspace (you can only clear the whole buffer with the F/T 1) and it has automatic 'skip over perforation' (particularly useful with Zeap, which has no paging capability). Unfortunately, you lose many of the print options which are needed for 'correspondence quality' text, and you also have no pixel characters. It would be nice to have a machine with the best features of both versions without having to swap ROMs.

The MX80 is a very well designed machine - a great improvement in appearance and performance on the TX 80. In fact the most serious faults I can find are the DIP selection switches, which are only accessible by removing the whole upper case, and the buzzer which indicates that the printer is out of paper - this goes on for a full 30 seconds, and it can be infuriating (this fault has been rectified on the F/T 2).

***** *** *** *** *****

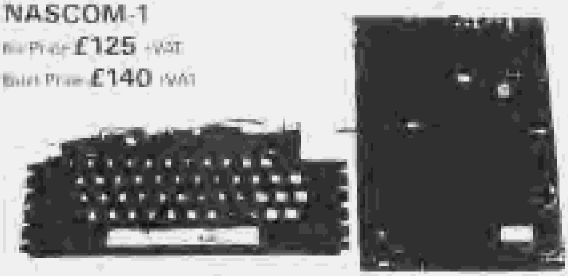
nascom

.... Computing for everyone

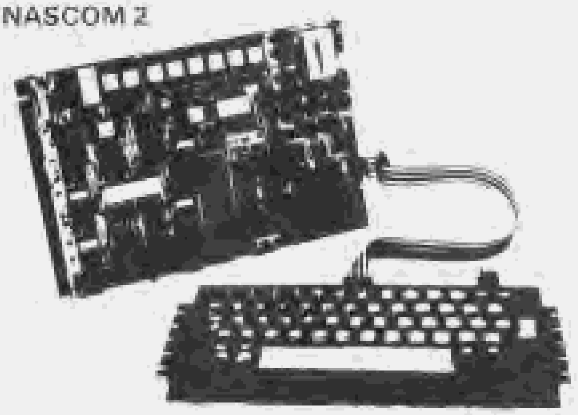
NASCOM-1

Kit Price £125 +VAT

Built Price £140 +VAT



NASCOM 2



RAMBOARD

16K £100 + VAT 32K £115 + VAT 48K £130 + VAT

I/O Board

£45.00 +VAT

TO BE INTRODUCED:

Colour graphics board

WATCH IT!!

Please send SAE for list of distributors

Optional Packs for I/O Board

PIO Pack

UART Pack

CTC

NASCOM IMP PLAIN PAPER PRINTER

£325 +VAT

NASCOM FIRMWARE

ASSEMBLER £50.00 plus VAT

DISASSEMBLER £37.50 plus VAT

NASPEN

£30.00 plus VAT

NAS-SYS 3: THE NEW OPERATING SYSTEM FOR NASCOM 2

£25.00 plus VAT

See us at the Personal Computer World Show Stand B28

New A moulded case for the Nascom computer

New Twin floppy disks

Lucas Logic

Nascom Microcomputers
Division of Lucas Logic Limited
Welton Road Wedgnock Industrial Estate Warwick CV34 5PZ
Tel. 0926 497733 Telex. 312333

HANDS-ON!
bv Viktor

This series of articles aims to take the proud owner of a newly-assembled, up and running Nascom from the intial stages of diplomatically negotiated half-hours away from gardening and decorating to the point at which you spend every evening and most weekends to the accompaniment of "you think more of that machine than of me!". In the case of the single person the machine meets with less direct opposition and the time to total obsession is much shorter.

When you start using the computer everything is a mystery. Facts which you need to know are hidden deep within a manual which has been written by someone who is thouroughly familiar with all aspects of computing and who has forgotten the depth of ignorance of the tyro. So here is an introduction to the Nascom 2 written by someone who is still learning. I hope that I can bring the manuals to life and ease the path for other beginners.

The keyboard and screen

When you switch on the computer, the screen is empty apart from a message at the top left with a short flashing line underneath it. The 'Message' is the name of the operating system - the set of instruction within the machine which tells it how to communicate with the outside world. There have been several different operating systems for Nascoms; the first used with the Nascom 2 was Nas-Sys 1, and the latest is Nas-Sys 3. If you go along to a computer club you will find that these boring 'commercials' have been replaced by much more enterprising expressions, from "Now what!" to phrases which are quite unprintable. The operating system is often referred to as the monitor, which is unfortunate, as the same term is used for the display device (also known as a CRT or VDU – it sounds much more scientific than TV).

The flashing line is the cursor - it is there to tell you where anything you type on the keyboard will appear on the screen. The cursor can be moved about the screen with the cursor control keys - the four keys on either side of the space bar. If you move the cursor and then type a letter, this letter will replace the cursor, which will move to the next available space. If you move the cursor over letters that you have already typed, you will notice that it does not erase these letters – it is a 'non-destructive' cursor. You can erase text one character at a time with the "backspace" key if you hold down the shift key and then press "enter" (also referred to as the "new line' or "return" key) you can remove the whole line that the cursor is currently on, while shift/backspace clears the whole screen. You can open up a space in a line of text by positioning the cursor with the control keys until it is under the first letter to be moved and then typing shift/cursor right. The whole line to the right of the initial position moves to the right, and letters can then be inserted as required. Similarly, shift/cursor left can be used to delete characters, the line to the right of the deletion point moving to the left. In fact, the Nas-sys full-screen editing facilities are excellent, and put many other copmputers to shame.

The enter key has a very important function; it tells the computer to scan the line containing the cursor and carry out any legitimate command on that line. However, computers tend to be very fussy about the commands they accept; if you put the command in the wrong place, or make a mistake in spelling or punctuation, you will upset it and it will come back at you with its favourite message - Error.

Screen Layout

The Nascom uses a memory mapped display. This means that each position at which a character can appear on the screen corresponds to a location in a special region of the computers memory, often referred to as the VDU RAM (the term 'RAM', and acronym for Acc ess Memory dating from the early days of computing, is now used for any type of memory that can be modified directly by the processor during the running of a program). To produce a character on the screen you merely have to put the appropriate code into the correct location. Special circuitry scans the display memory continuously and converts the data it finds there into a signal which writes the characters on a TV screen. You can change the contents of the display memory directly from the keyboard and watch the characters appear. Clear the screen by typing shift/backspace, and then enter M9E3, i.e., type M9E3 and then press 'enter'. Make sure that the letter M is on the far left of the screen, otherwise the computer will ignore it. If you have entered the command to the computers satisfaction the cursor will jump to the next line down the screen where 09E3 20 will be printed, with the cursor flashing at the number 2. This shows that the current content of memory location 09E3, which represents the middle of the screen, is 20 – the code for a space. If you now type 07, when you press 'enter' this character replaces the space and a small bell-shaped figure with two legs projecting downwards appears in the middle of the screen. You will have noticed that with this method of entry the position at which the change occurs is independent of the cursor location.

The screen can hold 16 lines of 48 characters. The top line is special; you will find that you cannot move the cursor onto the top line by means of the control keys. Of course, you can put data directly into the memory locations corresponding to the top line; if you modify the contents of location BE3 to 07 the little bell will appear in the middle of the top line. If you try to move the cursor off the bottom of the screen, you will find that the screen 'scrolls', that is, the cursor stays on the bottom line but all the data on the screen is moved up one line. However, the contents of the top line are unchanged, and the data in the second line is lost. The top line is used for headings, program names, etc. The Nascom screen is not mapped to the display memory in a straighforward way; the start of the memory corresponds to the second line of the display, it runs down to the bottom line, and it is then followed by the top line. It is thus usual to refer to the scrolling part of the display as lines 1 - 15, and the top line as line 16, which can cause some confusion until you become used to it.

Getting Basic

The microprocessor which controls the operation of your Nascom is an electrical device, and it only responds to the patterns of electricity known as

machine code. One way to communicate with your Nascom is thus for you to learn machine code, but a simpler method is to make the computer do the 'thinking' and use the Basic interpreter with which the Nascom is provided. A Basic interpreter is a list of instructions, written in machine code, which tell the Nascom how to obey a series of standard command words. Because most of the commands are simple English words it is much easier to learn to control a computer with Basic than to learn machine code. In the Nascom 2 the Basic interpreter is stored in a device known as a read-only memory (ROM) because, while data can be read from this memory when required, the processor cannot change the contents of the device by writing new data to it.

After switching on the computer you start the Basic interpreter by entering J. This is known as a 'cold start'; it allows the interpreter to intialise an area of memory, known as the Basic workspace, in which it stores information it needs during the operation of the interpreter. After a cold start the machine displays the prompt "Memory Size?" and then waits for you to tell it how many memory locations are available to it. If you press enter without typing in the number of locations the machine will work out the number for itself by searching through possible memory locations from the end of its workspace upwards until it finds a position that does not change when it writes to it. You only need to enter a value if you wish to stop Basic using a particular region of memory.

Once you are in Basic you can enter programs, that is, lists of instructions, from the keyboard or from tape. If you enter a line number, that is, any whole number up to 65535, followed by a command or series of commands the whole line will not be obeyed immediately, but it will be stored in the computers memory. Any command entered without a line number is executed as soon as you press "newline". Should you come out of Basic and then return to it with another cold start you will lose any program you may have entered. To get back to the interpreter without loss of program you must use the 'warm start' command, Z.

At this point the 'reset' button must be mentioned. When you press reset you force the processor to stop doing whatever it is doing at the moment and obey the instructions which begin at location 0, which is the start of the operating system. Data in memory is not corrupted by a reset, and so you can use the reset button to exit from Basic, and can then return to Basic with your program intact by entering Z.

Finally, as a demonstration of the power of Nas-Sys screen editing in Basic, try the following. Cold start the interpreter and type the lines:

10 REM: NASCOM RULES!

and then press 'enter'. If you move the cursor with the control keys to the line number, modify this to 20, and press 'enter' you will now have two lines stored, as you can check by entering the command LIST. Line duplication may seem to be of limited use, but in fact it can be an extremely effective method for changing the order of lines when 'debugging' (removing errors) or modifying complicated programs.

In the next article I shall delve further into Basic, and also cover a few points on the use of the cassette interface.

**** ** ** ** ** ** ** ****

io systems ltd.

NASCOM GRAPHICS

VERY HIGH RESOLUTION FOR NASCOM 2

380 x 220 individually addressable points

FEATURES :

- fully bit mapped from dynamic RAM
- software controlled
- software supplied for point-plot, line-draw,
  -block-shading and display control
- mixed text and graphics
- real time plotting
- display size variable to suit memory available

Price ... £55 + 15% VAT (post free)

EPROM PROGRAMMER

FEATURES :

- programs: 3-rail : 2708, 2716
  and single rail : 2758, 2508
  2716, 2516
  2732, 2532

- EPROM type selected by plug-in modules - 3 modules
- supplied with simple wiring diagrams for all EPROM types
- driven from NASCOM 1 or 2 PIO
- powered from NASCOM and transformer (supplied)
- software supplied for READ / PROGRAM / VERIFY

****** CAN BE USED WITH OTHER MACHINES WITH 2 PARALLEL PORTS

Price ... £ 63 + 15% VAT (post free)

Both products built & fully tested supplied with comprehensive documentation and full instructions for simple installation

Send SAE for free data sheets

AVAILABLE NOW direct from:-

io systems ltd.
6 LALEHAM AVENUE, LONDON NW7 3HL.
Tel. 01-959 0106

SNOWDINGER
by Dougal

If you have a Nascom 1, you will have noticed that certain processor activities cause a lot of on-screen disturbance patterns of apparently random white speckling all over the TV screen. You may, of course, have modified your machine by fitting the "snow plough" suggested in the first two issues of INMC news, or you may have a Nascom 2 (or a Video Genie, an Atom, etc., - they all seem to have similar problems). In that case you will still notice on-screen disturbances in the form of short black lines chopping up the displayed information. On a mainly black screen this can just about be tolerated, but with any sort of graphics capability all that the "snow-plough" approach succeeds in doing is to substitute black "snow" for white.

This is a great pity because a clean display is so much more satisfactory, and there are, in fact, several ways in which this can be achieved. The method detailed in this article is economical, and it has been success-fully implemented by a number of Nascom owners in my local club (even O.N.J. managed it).

The disturbances are, of course, caused by the need for two functionally independent units - the display logic and the processor - to access the same memory area: the V.D.U. RAM. The display logic cannot be kept waiting for the obvious reason that that the televison line scan just keeps right on going, so in the Nascom designs (and many others) the processor grabs the address and data busses to make its access, and whatever the display logic is trying to do at that time is simply not done. Either, with an unmodified Nascom 1, it is fooled completely, and proceeds as if the data that the processor has just accessed is the data that it was itself trying to access - in which case it looks up in the character generator ROM an "appropriate" bit pattern for the line currently being displayed, and merrily puts that up on the screen -or, with a Nascom 2 or a "snowplough", it blanks out any information from the display for a period covering the disturbance, which is fine if nothing is supposed to be displayed at that position on screen, but not otherwise.

So, what approaches can be taken to provide a final solution - a good, clean display with no disturbances whatsoever? Three solutions come to mind, all based on the principle that if the display logic must not be disturbed, then the processor will have to be.

1) A software solution. For about 4 milliseconds in every 20, nothing will show on the screen because this is the frame-blanking time. All the screen updating can be done during this period, as follows. Two "monitor" routines are needed, one entered by means of a frame-sync interrupt, the other entered by a user call. The interrupt routine is given sale and exclusive authority to manipulate the contents of the video RAM, which it does by working down and emptying a list of demands. These are prepared for it by the other routine, which makes an entry in the demand list for every call from a user program.

2) A hardware solution. Use "wait-state" logic to permit processor accesses into V.D.U. ram only during the line-blanking time, which is 16 microseconds in every 64. Of course, during frame-blanking accesses need not be delayed.

3) Another hardware solution. The display logic accesses the V.D.U. RAM once per microsecond - it is Just possible to arrange for two accesses in every microsecond, and hand over alternate ones to the processor when it needs them. Wait state logic can then be used to synchronize the processor with these time slots.

Now all these are perfectly feasible solutions, but each has its own peculiar advantages and disadvantages. Approach (1) for example would need a new version of Nas-sys, and would meet with displeasure from all those who like the flexibility of memory-mapped display (no more Basic pokes to screen for example).

Approach (2) involves the processor in waits of up to 48 microseconds, and also has some wrinkles associated with the fact that the first character displayed on a line is actually being accessed 2 microseconds earlier. This approach formed the basis of a brave but unfortunately over-simple attempt in issue 3 of INMC 80 - problems such as this cannot be tackled just with Boolean algebra. The best tools are sheets of squared paper on which waveform diagrams can be drawn and redrawn, an the detailed information provided by the chip manufacturers on how to make Z-80s Jump through hoops.

A solution to approach (3) is presented here. It too has its disadvantages: early Nascom 1's may have V.D.U. RAMs which are too slow to permit two accesses per microsecond. As it is, the modification is so simple that it is worth trying anyway - if the result produces spurious display misbehaviour then your RAMs are too slow, in which case have a go at approach (2) (if you do, be sure that you completely understand the display mechanism, prepare careful timing diagrams, and study the Z-80 manual on T-states and access cycles),

Another disadvantage is that the Z-80 will no longer be able to execute code from the V.D.U. RAM - which is a daft thing to want to do anyway. (This quirk occurs because, for reasons best known to the chip designers, M1 cycles - code accesses - have a different timing requirement to all others. This already creates all sorts of problems on Nascoms, such as the need for wait-states to access EPROMS on 4MHz machines.)

The advantages of approach (3) are, however, persuasive. Unlike (2), the maximum wait of the processor is one microsecond - undetectable in practice. It is extremely simple - there is little more logic involved than was required for the "snowplough". The 2MHz and 4MHz versions are similar enough to permit a speed selection switch, for those of you whose reflexes are not up to landing a moon-rocket at 4 MHz, and don't want to modify software.

The waveform diagrams contain all of the theory of operation. One very important point to realise is that the processor clock is the inverse of that used by an unmodified Nascom - there is an extra inverter in the signal path. This is essential to ensure that the "wait state" generator is sampled by the Z-80 at the only

suitable time - in order to complete the access by the end of the 500 nanosecond processor time slot.

Two other signals are generated. One ($\overline{\text{RVSEL}}$) is intended to gate the address and data steering logic and permit a "read-enable" to reach the RAM. The other ($\overline{\text{WVSEL}}$) is a signal timed to be contained within this, used to permit a "write enable" to reach the RAM, with satisfactory data set up and hold timing. A small capacitor can be fitted to lengthen this signal slightly - this is only needed for slow RAMs and is included because some of the early Nascom 1's might then be able to use this approach.

On the implementation given, these signals are generated twice, for two different address fields. One is the V.D.U. RAM, but some readers will be interested in the possible application of the other. The club mentioned earlier has become dissatisfied with the limited character set imposed by ROM-based graphics, so a programmable character generator is being included as part of a P.C.B. we are developing (The resulting board will be 8"x8", Nasbus compatible, and also contain 64K of RAM and a few other bells and whistles, for those interested). Now the immediate use of programmable graphics is to download special characters chessmen is a good example - from tape along with the software that uses them. After that, however, is the rather more fascinating possibility of using this programmability dynamically to produce smooth movement: watch your Klingons grow in size in your starship viewscreen as they sweep in to the attack.

Our P.C.B. uses separately addressed RAM, which is expected to be regularly accessed by both the video display logic and the processor, so the same conditions exist as for the V.D.U. RAM. Not surprisingly, the same solution to on-screen disturbances also applies; hence the second pair of signals. Since they require no additional I.C.s for their generation, I have included them in the diagram.

The instructions constitute a modification for Nascom 1, for which I won't apologise: there are a lot of you, and also you have lived with the problem longer! Nascom 2 owners should have little difficulty in working out their own version, but if there is sufficient demand then another article is always possible. To assist with installation a sketch is included of the veroboard layout as fitted to my own machine. It will be noticed that the connections to IC31 are made by adding "extension legs" from the pins of the 7406 down through the veroboard. This makes for space saving, wiring minimisation, and a certain amount of mechanical support all in one, so is to be recommended. These extension legs were taken from a broken DIL headers I suggest that you experiment and search your Junk box.

INSTALLATION

In the instructions which follows "Lift IC xx pin y" means: carefully remove IC xx from its socket, with a pair of thin, flat-nosed pliers bend pin y horizontal so that it no longer makes contact with the socket, and replace. "Connect pin of IC xx/y to ..." means: solder one end wire directly to the pin bent horizontal in a "Lift" instruction.

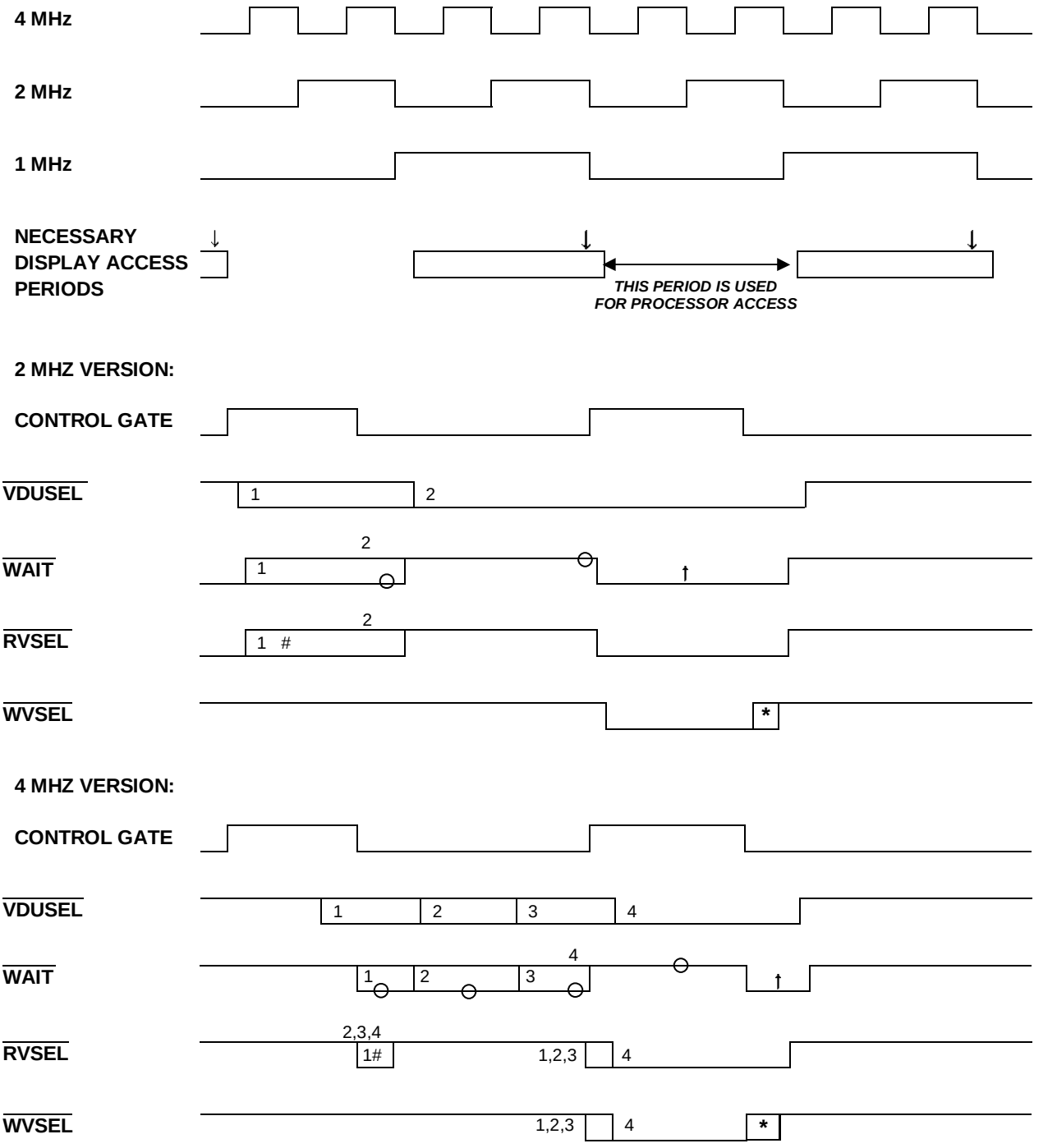
"Connect pad of ICxx/y to ..." means: solder one end of wire to any suitable point on the P.C.B. track that pin y used to connect to, before it was lifted. Plated through holes are ideal for such connections.
"Connect IC xx/y to . .." means: solder one end of wire to any suitable point on the P.C.B. track that pin y connects to.

Preliminary: Remove "snowplough" if fitted and restore board to its original state.
1): Remove IC 31. This 7406 may be used, if desired, on the veroboard, or retained in case the 2102's are too slow
2): Lift IC 11 pin 5. Connect pin of IC 11/5 to IC 11/4
3): Lift IC 45 pin 13
4): Lift IC 36 pin 6
5): Plug veroboard into IC 31, ensuring that the 7406 has the same orientation as originally. The major part of the board fits over the group of R's and C's adjacent to IC 31, and between the PIO and the character generator. Subsequent instructions assume that the switch is already wired in, and that wires using the colour code on the logic diagram have been soldered to the veroboard.
6): Connect pin of IC 45/13 to WVSEL (black)
7): Connect pin of IC 36/6 to VDUSEL (blue)
8): Connect pad of IC 36/6 to RVSEL (white)
9): Connect 1MHz pin adjacent to IC 19 to 1MHz (brown)
10): Connect 2MHz pin adjacent to IC 19 to 2MHz (red)
11): Connect 4MHz pin adjacent to IC 19 to 4MHz (yellow)
12): Connect IC37/24 to WAIT (green)

NOTE - Those of you who are using Nascom buffer boards are currently driving their processors with a different oscillator to the on-board one driving the V.D.U, logic, this being Nascom's solution to a bus skew problem. The modification presented here cannot tolerate independent oscillators since it uses a synchronisation technique, and continues to use the Nascom 1 on-board oscillator to drive the Z-80. The buffer oscillator must be diconnected in its entirety from both the Nascom 1 interface, pin 38, and the Nasbus, pin 5. The latter, if connected to pin 11 of the 7406 on this modification, produces the same solution to skew as the buffer implements - i.e., two gate delays. If you don't like flying leads, and are prepared to do some P.C.B. track cutting, isolate the Nascom 1 interface pin 38 and use this as a method of linking the signal from the Nascom 1 to buffer and hence to Nasbus.

When you have carried out the above modifications, try the system at 2MHz. Any screen misbehaviour will probably show with "tabulate", but experiment with several programs making dynamic use of the screen. If misbehaviour is seen, fit a 150 pf capacitor between pin 8 of the 7406 on the veroboard and ground. If the system still doesn't work properly then your V.D.U. RAMs are too slow - either get some faster 2102's or try an alternative suggested in the text. If all is well, and providing that you are quite certain that your system can run at 4MHz i.e., it did so before installing the modification - then carry out the same tests at the higher speed. Hopefully, you now have a clean screen without any disturbances.

WAVEFORM DIAGRAMS



NOTES **1,2,3,4** *- DIFFERENT STARTS OF Z-80 ACCESS CYCLE WITH RESPECT TO 1 MHZ*

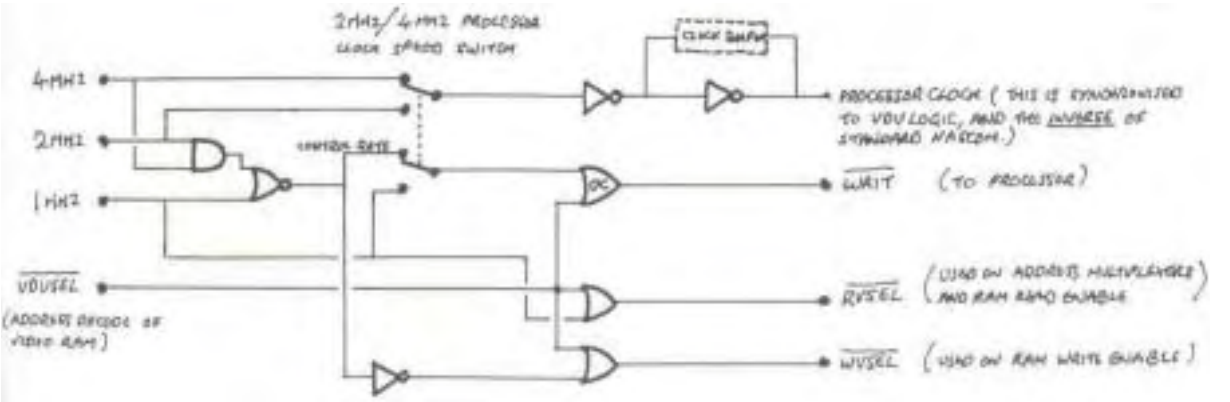
-THIS POINT IN TIME THAT THE 'WAIT' SIGNAL IS SAMPLED BY THE Z-80

***** *-EFFECT OF ADDING CAPACITOR TO EXTEND WRITE ACCESS – NOT NORMALLY NEEDED*

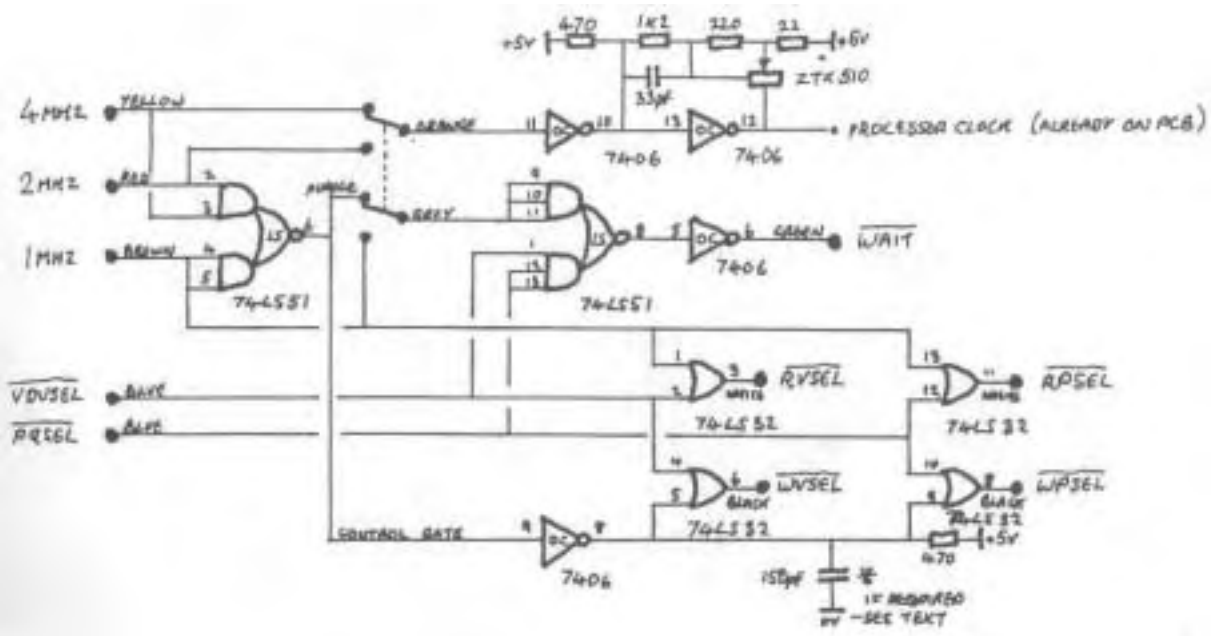
↑ *-THIS ENSURES THAT 'WAITS' GENERATED FROM DIFFERENT SOURCES DO NOT DISTURB THIS SYNCHRONISATION LOGIC.*

\# *--THIS 'PRELIMINARY PULSE ON $\overline{\text{RVSEL}}$ HAS NO EFFECT SINCE CYCLE IS NOT COMPLETE, IT IS DURING AN UNUSED PROCESSOR ACCESS 'SLOT', AND THERE IS NO CORRESPONDING WRITE GATE $\overline{\text{WVSEL}}$*

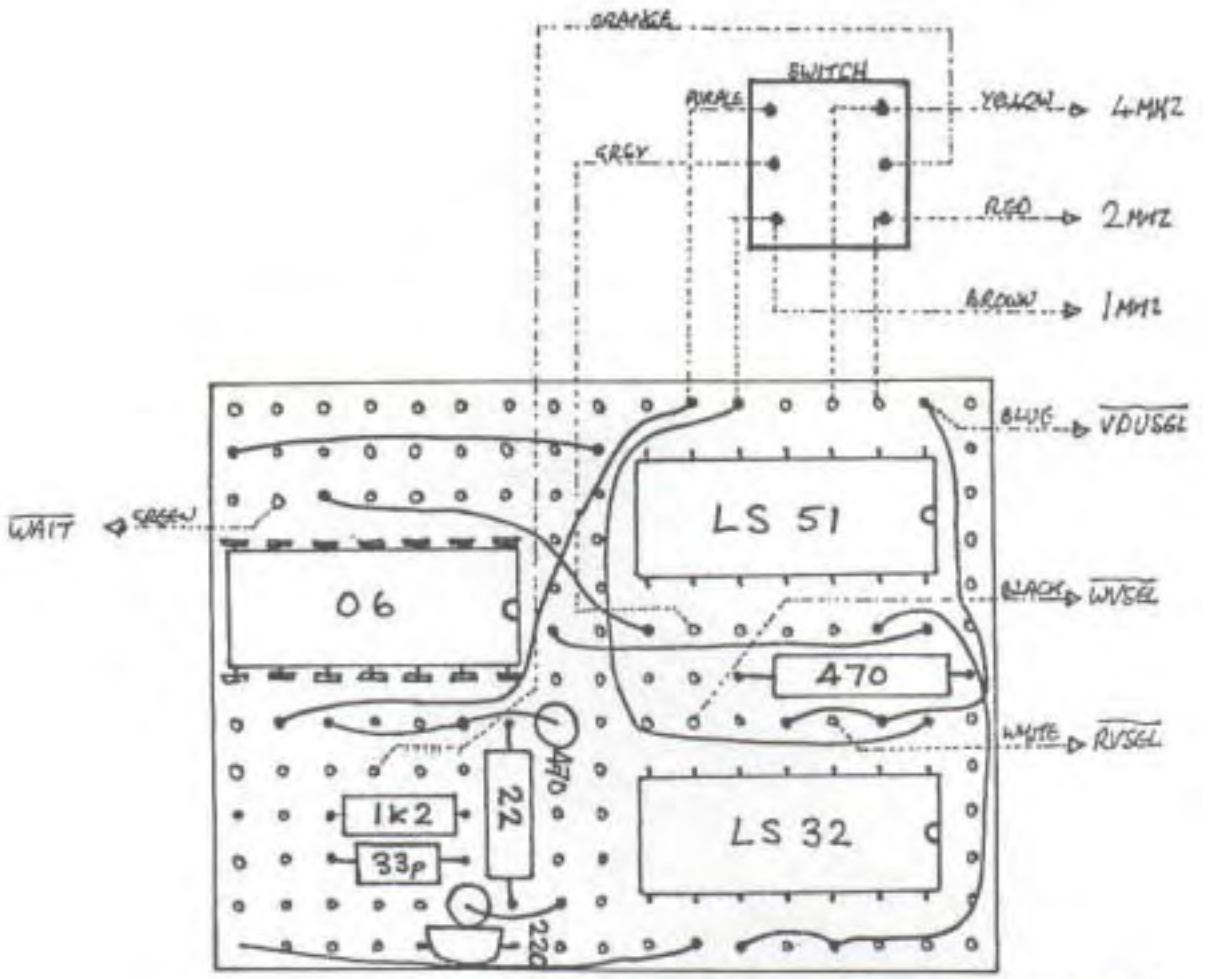
BASIC FORM OF MOD



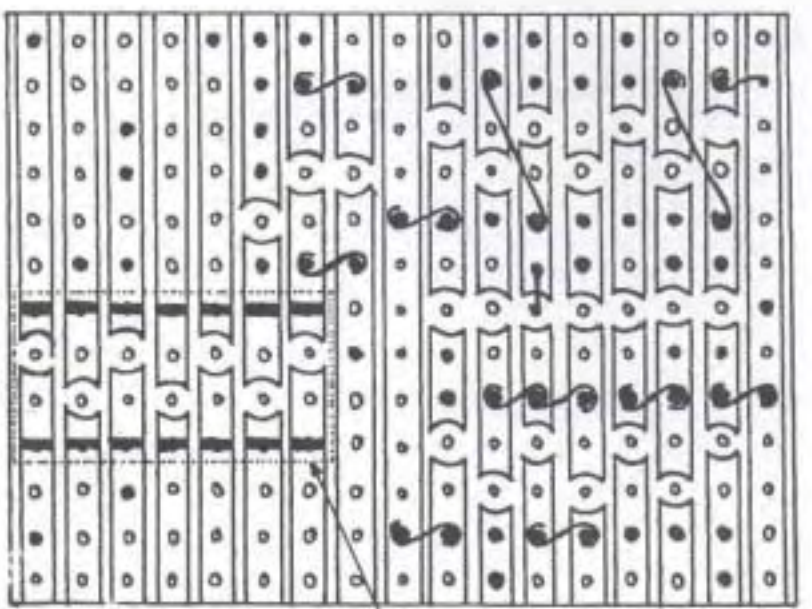
NASCOM – 1 IMPLEMENTATION



NOTE: Gates Marked 'OC' are Open-Collector Outputs. $\overline{\text{WAIT}}$ Must be Driven by Such a Gate, to allow Common-Collectoring with Other Wait-State Generators



EXTERNAL CONNECTIONS SHOWN DOTTED AND COLOUR-CODED
EXTERNAL CONNECTIONS FOR PG ADDRESS HOLD OMITTED



PINS FROM D.I.L. HEADER EXTEND LEGS OF 7406
DOWN BELOW VEROBOARD. PLUGS INTO IC31 SOCKET.

RUBIK CUBE DISPLAY
by J. K. Richardson

At nearly all the club meetings I have been to recently someone has been playing with a Rubik cube. The program below displays a front and rear view of the cube. The program accepts the standard notation for the cube (using Left, Right, Front, Back, Up, Down for the faces); thus L1 or L Newline rotates the left face 90 degrees clockwise, L2 rotates it 180 degrees, and L3 or L' rotates it 270 degrees. You can backstep through a sequence of moves with BS, and Jump directly to an unscrambled cube with shift/newline. The program is the first step towards a program which solves the cube - I should be pleased to hear from anyone who has already written such a program!

```
TC80 FE0 0 10 1
   0C80   C3 47 0E 55 09 11 09 CD 08 91 08 55 08 99 08 DD 08 19 09 D5 08 94 09 14
   0C98   0A 94 0A 50 0A 0C 0A 8C 09 0C 09 50 09 D0 09 96 09 5A 09 1E 09 9E 09 1E
   0CB0   0A 5A 0A 96 0A 16 0A DA 09 AF 09 F3 09 37 0A 73 0A AF 0A 6B 0A 27 0A EB
   0CC8   09 2F 0A 6E 09 AA 09 E6 09 66 09 E6 08 AA 08 6E 08 EE 08 2A 09 70 09 F0
   0CE0   08 70 08 B4 08 F8 08 78 09 F8 09 B4 09 34 09 02 06 05 03 03 04 06 01 01
   0CF8   05 04 02 06 02 03 05 04 03 01 06 05 01 02 04 55 4C 46 44 52 42 52 55 42
   0D10   49 4B 27 53 20 43 55 42 45 8E 0F 79 D6 03 30 02 C6 06 4F C9 7A CD 35 0D
   0D28   CD 1B 0D E5 06 00 21 07 0D 09 56 E1 C9 E5 21 0C 0D 01 06 00 ED B9 E1 C9
   0D40   21 0A 0B 06 30 22 29 0C 36 20 2C 10 FB C9 DD 7E 00 87 87 87 DD 86 00 87
   0D58   D5 5F 16 00 FD 21 71 0C FD 19 D1 DD 23 C9 79 87 87 87 81 87 16 00 5F DD
   0D70   21 83 0C DD 19 06 08 DD 6E 0C DD 66 0D 56 DD 6E 0E DD 66 0F 5E DD 6E 00
   0D88   DD 66 01 7E 72 53 5F DD 23 DD 23 10 F0 79 87 87 5F 16 00 DD 21 EF 0C DD
   0DA0   19 DD E5 CD 4E 0D FD 6E 00 FD 66 01 56 CD 4E 0D FD 6E 08 FD 66 09 5E 72
   0DB8   CD 4E 0D FD 6E 0C FD 66 0D 56 73 CD 4E 0D FD 6E 04 FD 66 05 5E 72 DD E1
   0DD0   DD E5 CD 4E 0D FD 6E 00 FD 66 01 73 FD 6E 0E FD 66 0F 56 CD 4E 0D FD 6E
   0DE8   06 FD 66 07 5E 72 CD 4E 0D FD 6E 0A FD 66 0B 56 73 CD 4E 0D FD 6E 02 FD
   0E00   66 03 5E 72 DD E1 DD E5 CD 4E 0D FD 6E 0E FD 66 0F 73 FD 6E 0C FD 66 0D
   0E18   56 CD 4E 0D FD 6E 04 FD 66 05 5E 72 CD 4E 0D FD 6E 08 FD 66 09 56 73 CD
   0E30   4E 0D FD 6E 00 FD 66 01 5E 72 DD E1 CD 4E 0D FD 6E 0C FD 66 00 73 C9 EF
   0E48   0C 00 21 8E 0F 22 19 0D 21 0D 0D 11 DC 0B 01 0C 00 ED B0 21 83 0C 0E 06
   0E60   11 07 0D 1A 13 D5 06 09 5E 23 56 23 12 10 F9 D1 0D 20 F0 CD 40 0D EF 4D
   0E78   6F 76 65 3A 20 00 CF 00 00 FE 1B 28 C2 FE 08 20 21 2A 19 0D 2B 11 8E 0F
   0E90   ED 52 19 38 DE 7E CD 35 0D 1E 31 28 06 5F 2B 7E CD 35 0D 57 3E 64 93 5F
   0EA8   18 0D CD 35 0D 20 C4 57 F7 00 00 CF 00 00 5F D5 FE 0D 28 2F FE 31 28 2B
   0EC0   FE 32 28 24 FE 33 28 1D FE 27 28 19 FE 41 20 08 CD 66 0D CD 1B 0D 18 13
   0ED8   FE 53 28 03 D1 18 94 CD 66 0D CD 1B 0D CD 66 0D CD 66 0D CD 66 0D 2A 19
   0EF0   0D D1 D5 11 8E 0F A7 ED 52 19 D1 28 4B 2B 7E CD 35 0D 06 01 28 03 47 2B
   0F08   7E 23 BA 28 06 05 28 38 23 18 35 2B 7B FE 53 28 04 FE 41 20 1B 78 3C E6
   0F20   03 20 0E CD 24 0D 7B 1E 01 FE 41 28 C5 1E 03 18 C1 F6 30 23 77 23 18 EB
   0F38   80 E6 03 2B 28 2A 23 3D 28 26 C6 31 23 77 18 20 72 7B FE 53 20 08 1E 33
   0F50   23 CD 24 0D 18 F2 FE 41 20 04 1E 31 18 F2 E6 03 FE 01 28 04 F6 30 23 77
   0F68   23 11 BE 0F A7 ED 52 19 30 03 22 19 0D 21 4A 0B CD 43 0D 21 8E 0F ED 5B
   0F80   19 0D A7 ED 52 19 CA 73 0E 7E F7 23 18 F4 4C 33 76 69 6E 67 20 74 6F 20
   0F98   74 68 65 20 A0 6C 65 66 74 2E 20 49 6E 20 66 61 63 74 2C 20 74 68 65 20
   0FB0   4E 61 73 2D 73 79 73 20 66 75 6C 6C 2D 73 63 72 65 65 6E 20 65 64 69 74
   0FC8   69 6E 67 20 A0 66 61 63 69 6C 69 74 69 65 73 20 61 72 65 20 65 78 63 65
```

NAS-SYS MONITORS
by J. Haigh

INTRODUCTION

The purpose of this series of articles is to dissect the latest Nascom monitors, Nas-sys 1 and Nas-Sys 3, partly to provide a backcloth for a discussion of machine code programming; and partly because you can only use monitor routines effectively when you understand them and I hope to gain understanding by writing the articles.

Anyone who has used other microcomputer systems will realise that the Nascom monitors in general are very powerful for their size. This does not mean that they should be regarded as sacrosanct - ideally the standard monitor supplied with your machine should be regarded as the starting point for a truly personal system. Tampering with monitors has a peculiar fascination; you usually have a strictly limited space in which to fit the routine you are rewriting, and it always seems to be too short. In the articles I shall try to stick to the authorised versions of Nas-Sys , but I shall probably suggest the occasional 'improvement'. I hope this will encourage people to write in with their own modifications, or at least to write and say why mine are wrong.

I shall assume that the reader is familiar with the commoner 'buzz words' for computing and knows what hexadecimal code is (if you have never heard of hexadecimal, you should read the series Teach Yourself Z80 in INMC 80). If there is anything you don't understand, you can always write and ask.

THE RESTARTS

The Z80 has a set of eight single-byte instructions which call subroutines on page zero.

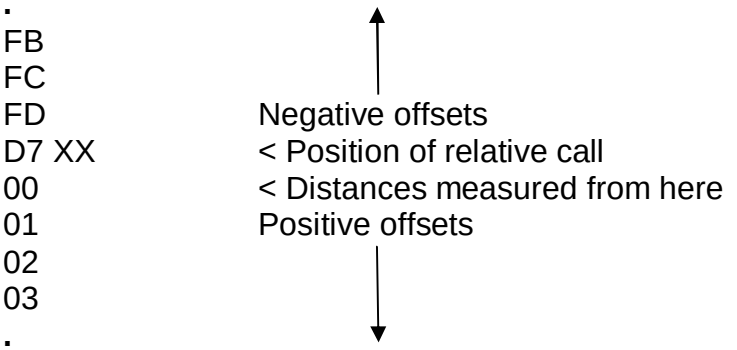
HEX. Code	Assembly Name	Address Called	Nas-Sys Function
£C7	RST £00	£0000	Resets the system
£CF	RST £08	£0008	Waits for an input character
£D7	RST £10	£0010	Relative call routine
£DF	RST £18	£0018	Access to Nas-sys routines
£E7	RST £20	£0020	Breakpoint code
£EF	RST £28	£0028	Print a character string
£F7	RST £30	£0030	Output the character in A
£FF	RST £38	£0038	Delay determined by value in A

RST 0 can be used on any monitor to reset the system at the end of a program, but any information on display will be lost, so its value is limited.

The second restart is the Nas-sys equivalent of the CHIN routine (CD 3E 00) in T4 and the earlier monitors; it uses a subroutine in the monitor to scan continuously through a table of routines until one of them receives an input. The address of the table is stored in the workspace at £0C75, and on initialisation this points to a keyboard and a serial input routine. The user can extend or change the table with the U and X commands, or he can provide his own input table. If one of the routines in the table sets the carry flag, subsequent routines will be skipped.

The Nas-Sys 3 table contains a repeat keyboard routine, which is not available under Nas-Sys 1.

The next two restarts (relative call, £D7, and subroutine call, £DF) both use the byte following the restart code to determine the address of the routine they are accessing. In a relative call the second byte is used as an offset to the routine being accessed, and as in the case of a relative jump instruction the distance is measured from the code following the second byte. If the offset lies in the range £80 - £FF the subroutine will precede the instruction calling it.



By using the relative call routine you can write programs that are completely relocatable - that is, they will work at any address without modification. This is particularly useful for programs that are to be stored in EPROM, as you can plug the device in any spare socket. Of course, you don't get this for nothing; the routine has quite a lot of software manipulation to carry out, and consequently it is much slower than a normal subroutine call. So, if you want a program to run as quickly as possible you will have to avoid relative calls in the critical parts of the code.

The subroutine call, £DF, uses the second byte to access a table which contains the addresses of monitor subroutines. In Nas-sys 3 this table starts at £0782; the first 26 addresses are the routines for the single letter commands. Only F and L are unused, and the table contains the address of the error-message routine for these letters. If you have an EPROM blower you can add extra commands by using these letters, or by replacing any of the standard commands that you don't use. As an example, here is a routine which compares two blocks of memory and list four lines of each block whenever it finds a difference.

```
1B          DEC DE        ; On entry DE and HL point
2B          DEC HL        ; to the start of the blocks
13  CPLOOP  INC DE        ; Increment the block pointers
23          INC HL
1A          LD A (DE)     ; Get byte from block 1 in
BE          CP (HL)       ; Is it the same as block 2?
28 FA       JR Z CPLOOP   ; If so, keep going
D7 0C       RCAL TAB      ; Tabulate first block
EB          EX DE HL      ; Swop the pointers to the blocks
D7 09       RCAL TAB      ; Tabulate the second block
EB          EX DE HL      ; Swop pointers back
CF          RST 8         ; Wait for user to press any key
```

```
FE 1B           CP £IB          ; Is it an 'escape' ?
20 EF           JR NZ CPLOOP    ; If not, look for next difference
DF 6A           SCAL CRLF       ; End of routine; scroll screen
C9              RET             ; and return to monitor
E5      TAB     PUSH HL         ; Save current poition of
D5              PUSH DE         ; pointers to blocks 1 and 2
01 F0 FF        LD BC -16       ; Tabulate from 16 bytes
09              ADD HL BC       ; before difference
54              LD D H          ; Put this address into DE
5D              LD E L
01 20 00        LD BC 32        ; Tabulate to 16 bytes
09              ADD HL BC       ; beyond the difference
EB              EX DE HL        ; Now get 1st address in HL,  2nd in DE
DF 54           SCAL "T         ; Call the tabulate command
D1              POP DE          ; Recover pointers
E1              POP HL
C9              RET             ; End of subroutine
```

You will see that the routine is relocatable; to incorporate it into your monitor as command F you must put the start address at £078C in Nas-Sys 3, at £0792 in Nas-Sys 1. To compare blocks of memory starting at £1000 and £2000 you simply enter F1000 2000. When one of the command letters is entered the first three hexadecimal values following the letter are held in HL, DE and BC, but if you want to run the program under the E command, entering EXXXX 1000 2000, you will have to prefix a routine to pick up the block addresses; for examples:-

```
2A 0E 0C        LD HL (£0C0E)   ; Get block 1 address in HL
ED 5B 10 0C     LD DE (£0C10)   ; and block 2 in DE
```

XXXX will be the start address of the modified program. Also the return instruction at the end of the program (not the one at the end of the subroutine) should be changed to DF 5B, SCAL MRET. Because this means replacing a single byte code by a two byte one you will have to move the subroutine down one byte and change the values in the relative calls. However, you can avoid this by using C7, RST 0, to get back to the monitor.

There are 34 remaining subroutine calls in Nas-Sys 1 (DF 5B - DF 7C). These cannot be entered as commands from the keyboard, but can all be used in programs. Nas-Sys 3 has 3 extra subroutine calls, including the repeat keyboard facility.

Restart £20, £E7, is used by the breakpoint routine to halt the execution of a program and display the current state of the processor. When a breakpoint is entered at address XXXX by entering BXXXX, the address is stored in the workspace at £0C23. The next execute command saves the code it finds at XXXX and then, providing that the address entered was not zero, it inserts a breakpoint restart. When the program arrives at this restart the original code is replaced, and the contents of the registers are saved in the workspace (from £0C61 to £0C6C) and displayed on the screen. The program then returns to the monitor to wait for the next command. If you now enter E without specifying an address execution continues at the replaced code; once again restart £E7 will be inserted (because £0C23 still contains address XXXX) and if

the program returns to this point the register display will be repeated.

But if the restart code is inserted when the Execute command is entered, why doesn't the program stop immediately when you try to continue after a breakpoint? In fact the £E7 is not inserted until one instruction has been carried out, as the following simple experiment shows. Enter the following code at £0C80:-

```
0C80  00        NOP       ; No operation
0C81  23        INC HL    ; HL register pair incremented
0C82  C3 80 0C  JP £0C80  ; Jump back to beginning
```

Press reset, then enter BC81 followed by EC80. The register display will show the contents of HL to be 0000; this is because the execute command picks up the values in the register save area of the workspace, and these bytes are zeroed by a reset. Now press reset, enter BC80 and then EC80. HL will be 0001; the restart was not inserted until the first instruction (NOP) had been executed, so HL was incremented before it came back to the breakpoint. If you insert £E7 directly in the program at £0C80 using the modify command, the program will break immediately on execution. The main use of this restart is in 'debugging' – tracing faults in programs that don't work properly. You can also use it as a single byte end to a program when you want to know the contents of the registers.

The next restart will be familiar to all Nascom users, as it has remained virtually unchanged from the earliest monitor. It is the 'print string' routine, £EF, which outputs the characters following the code until a zero is reached. The routine can handle control codes and graphics characters as well as the standard ASCII codes. The bytes in the string are loaded successively into the A register, and are then output by restart £F7. At the end of the routine the A register always contains 00, the byte that marks the end of the string.

The output restart, £F7, operates in a similar way to the subroutine called by the input restart; that is, it accesses a table of routines, the address of which is stored at £0C73. On initialisation this is set to use only the standard CRT routine, but the user can extend the table with the U and X commands, or can provide his own routine table. As in the case of the input restart, if one of the routines in the table sets the carry flag subsequent routines will not be used.

Finally, restart £FF produces a delay depending on the value in the A register when the restart is called. When A is one, the delay at 4Mhz, including the call and return, is 5.2 microseonds, while the maximum delay, produced when A is zero, is 2.7 milliseconds. Of course, the times are doubled with a 2Mhz clock rate.

That completes the Z80 restarts; in the next article I shall deal with some of the individual subroutine calls.

*** + * + * + * + * + * + * + ***

NEWS FROM THE CLUBS

I hope that we get enough feedback to make this a regular item - not just a boring list of names and addresses, but what goes on at the meetings, details of any pet projects the clubs are working on, etc. So will members of clubs please twist the secretaries' arms and get them to send as much information as possible.

The Mid-Sussex Microcomputer Club have sent in samples of their very impressive Newsletter. The club caters for all micros, but if the contents of the Newsletter are anything to go by it has a very strong bias towards Nascoms. The next two meetings of the club will be on Friday, August 21st, and Wednesday, September 16th, at Ardingly College. Further details from the secretary, B. Langton, Tain, Broadwater Lane, Copsale, Nr. Horsham, W. Sussex, RH13 6QW.

The Merseyside Nascom Users Group, which claims to be the largest Nascom club in the world, meets on the first Wednesday of each month in the Mona pub in Liverpool (near Pierhead). They are planning to start a regular newsletter in the near future.

Bristol Amateur Radio Society Computer Group, which meets on the last Tuesday of the month at the University Settlement, Dulcie Road, Barton Hill, Bristol, apparently consists almost entirely of Nascom enthusiasts. Further details - from Terry Rowe, Bristol 559398.

There is also the Nascom Group of Bristol Computing Club, run by Noel Wright, 3, Brighton Road, Redland, Bristol, Tel. 36552.

The Cornwall Amateur Radio Society Computer Group has a good proportion of Nascom users. Details from the president, A.H.Hammett, 1, Rose Hill, Ladock, Truro, Tel. Grampound Road 882758

Nottingham Micro-Computer Club meets monthly at Trent Polytechnic. They have sent us a copy of their Newsletter, which contains an interesting list of useful routines in the Nascom ROM Basic. Membership costs £5.00 per annum , Students and OAPs half price, family membership £7.50. Contact the secretary, K. S. Swainson, 9, Brayton Crescent, Bulwell, Nottingham, Tel. 0602 - 751742, or send a sae for a membership form to R. Brumpton, 182, Lowdham Lane, Woodborough, Nottingham, NG12 6DN

The South Yorkshire Personal Computing Group meets in the Department of Applied Science, Sheffield University (St. Georges Square) on the 2nd Wednesday of the month, at 7.30 pm.

The Kirklees Computer Club has a strong Nascom contingent. They hold informal meeting every Monday in the White Swan pub, Kirkgate, Huddersfield. A newsletter is to be produced, and the production team is headed by a Nascom owner.