MONITOR DOCUMENTATION.

NAS-SYS I

Index

SOFTWARE SECTION

NASCOM OPERATING SYSTEM - NAS-SYS

NAS-SYS is a 2K operating system for the Nascom 1 and Nascom 2 microcomputers. It makes it easy to enter, test and run machine code programs. It also provides a comprehensive set of routines which can be called by user programs. These routines are also used by the high level languages on the Nascom such as the 8K BASIC.

Facilities include:-

Modification and tabulation of memory.

Normal execution of programs, or single stepping through programs, or execution of a program with break-points set. During single stepping or at a breakpoint a full display of the machine registers is provided.

Comprehensive screen editing by using a blinking, non-destructive cursor. This makes it easy to correct and reenter incorrect commands.

The keyboard routine allows every possible code to be entered when using the expanded keyboard, and provides support for all features with the original Nascom 1 keyboard.

Cassette tapes can be used to store programs, using a fast and fully checked method of recording the data. Tapes can be written which automatically load and execute machine code programs without any commands being entered at the keyboard.

All parallel I/O ports can be controlled and checked by direct commands.

Full support of serial terminals and printers is included, so that the computer can be controlled by a Teletype. Paper tape can be used to store programs in ASCII format.

The computer can be used as a terminal, by connecting it with an acoustic coupler to a timesharing service, or even to exchange data with another Nascom.

All internal codes are in ASCII, making it easy to attach other peripherals such as parallel printers.

(This document was produced on a Nascom computer, using the NASPEN word processing package, and printed on a Teletype 43 terminal.)

LIST OF COMMANDS

A xxxx yyyy	Arithmetic - in hexadecimal
B xxxx	Breakpoint set or cleared
C xxxx yyyy zzzz	Copy - move data
E xxxx	Execute a program
G xxxx yyyy zzzz	Generate a self loading cassette tape
H	Half duplex terminal
I xxxx yyyy zzzz	Intelligent copy - move data safely
J	Jump to address FFFA - BASIC cold start
K xx	Keyboard option
L	Load a paper tape
M xxxx	Modify or examine data
N	Normal I/O to be resumed
O xx yy	Output data to port
Q xx	Query data from port
R	Read a cassette tape
S xxxx	Single step
T xxxx yyyy zzzz	Tabulate data or write a paper tape
U	User specified I/O routines activated
V	Verify cassette tape is readable
W xxxx yyyy	Write a cassette tape
X xx	External serial device activated
Z	Jump to address FFFD - BASIC warm start

(D, F, P, Y commands do not exist)

HOW TO ENTER A COMMAND

Type the command letter in the first position on the line, followed by any values required. Each value must be separated from the others by one or more spaces. Then press the `Enter´ (`New Line´) key. The line entered is ALWAYS the line where the cursor is blinking when the `Enter´ key is pressed. If the first position on the line is blank, no command is processed.

ERRORS WHEN ENTERING A COMMAND

If an invalid command is entered, the message "Error" is displayed on the next line. Very careful checking is performed by NAS-SYS, so that disastrous mistakes are less likely. The following types of error are detected:-

Invalid command character. (If the command character is blank, the line is simply ignored.)

More than ten values entered after the command character. (Only three values are ever needed, but a program can use up to ten values entered after the E command.)

Any value which is not a valid hexadecimal number. (Digits Ø to 9, and letters A to F.)

A value greater than hexadecimal FFFF.

HOW TO CORRECT ERRORS

Simply press `Backspace´ to correct errors which you notice while typing the command.

However the comprehensive screen editing facilities allow you to go back and reenter an incorrect line provided it is still on the screen. Simply move the cursor back up the screen, and edit the line using the cursor movement keys, including deleting and inserting characters if required. Then press `Enter´ to enter the line.

For details of the screen editing facilities, read the next section.

SCREEN EDITING

The screen editing facilities are available with both the expanded keyboard, and the original Nascom 1 keyboard.

	NAME	HEX CODE	KEYS PRESSED	FUNCTION
0	NULL	ØØ	CTRL/SHIFT/@	Ignored completely.
8	BS	Ø8	BACKSPACE	Move back and make position blank.
10	LF	ØA	LF or CTRL/J	Ignored on display.
12	FF	ØC	CS or SHIFT/BS	Clear screen and start at top left.
13	CR	ØD	ENTER or NEWLINE	Carriage return, line feed. Scroll up if at bottom.
17	CUL	11	Left arrow or CTRL/Q	Move cursor left.
18	CUR	12	Right arrow or CTRL/R	Move cursor right.
19	CUU	13	Up arrow or CTRL/S	Move cursor up.
20	CUD	14	Down arrow or CTRL/T	Move cursor down.
21	CSL	15	SHIFT/Left arrow or CTRL/U	Delete character at cursor, move rest of line to the left.
22	CSR	16	SHIFT/Right arrow or CTRL/V	Move rest of line to the right.
23	CH	17	CH or CTRL/W	Move cursor to start of line.
24	CCR	18	CTRL/X	If cursor at start of line, ignore. Otherwise do CR.
27	ESC	1B	ESC or SHIFT/ENTER	Delete current line, and place cursor at start of line.

THE KEYBOARD

The expanded keyboard has the ability to generate any of the 256 possible 8 bit codes, as follows:-

Hex 20 to 5F are the normal ASCII codes and are available as marked keys.

The letters are upper case, unless SHIFT is held down, when they become lower case. (Also see K command.) The @ key requires SHIFT to be held down, to produce an @.

Both the CTRL and the @ key operate as CTRL keys. This enables original Nascom 1 keyboards to use the screen editing facilities. When one of these keys is held down, and another key pressed, bit 6 is altered. This gives codes 00 to 1F and 60 to 7F.

The GRAPHICS key sets bit 7 on while it is held down, and this can be used to give graphic characters directly, with the Nascom 2 graphic option. When used in conjunction with the other keys, codes 80 to FF may be obtained. (Also see K command.)

THE RESET BUTTON

The Reset button is quite different to any other key. It sends a signal to the computer telling it to reinitialize itself. Pressing Reset does not result in the loss of whatever programs are stored in memory, but the operating system resets itself and takes control. The message "NAS-SYS 1" is displayed at the top left of the screen, and the computer waits for you to enter a command.

SCROLLING DISPLAY

When Reset is pressed or the screen is cleared, the cursor is positioned at the top left, and as commands are entered, the display scrolls down the screen. When the bottom line is reached, the whole display automatically starts to scroll upwards, showing the last 15 lines.

The top line of the display is never scrolled, making it a convenient place to display headings.

DETAILED DESCRIPTION OF EACH COMMAND

A xxxx yyyy — ARITHMETIC COMMAND

The Arithmetic command performs simple hexadecimal arithmetic. Three results are displayed, as follows:-

SSSS DDDD JJ

SSSS is the sum of the two values.

DDDD is the difference of the two values.

JJ is the displacement required in a Jump Relative instruction which starts at xxxx, to cause a jump to yyyy. If such a jump is not possible, then ?? is displayed instead.

B xxxx — BREAKPOINT COMMAND

The Breakpoint command is used to insert a trap code into the program at address xxxx specified. A breakpoint cannot be inserted into a program which is in Read Only Memory, such as NAS-SYS itself. Therefore entering the command B Ø turns off the breakpoint.

Initially the breakpoint is turned off, since Reset deactivates it. The breakpoint command may be entered at any time. Once it has been entered, NAS-SYS remembers the breakpoint address and also keeps a record of the value at that address. Then, when an Execute command is entered, code E7H is automatically inserted at the breakpoint address. If the code is encountered during execution, then the program registers are saved, and are also displayed. The original value is replaced at the breakpoint address. Any command can then be entered, so that for example the program can be modified. If an Execute or Step command is then entered without specifying an address to start execution, the program will automatically restart where the breakpoint was. If an Execute command is entered, then the original instruction at the breakpoint is executed, and the breakpoint will only stop execution again the next time that it is encountered.

Note that the original value is put back into the program no matter which way the program is ended. (See "How to end a program".)

Note that the breakpoint must only be set at the first byte of an instruction in the program.

C xxxx yyyy zzzz — COPY COMMAND

Copy a block of data, length zzzz from xxxx to yyyy. One byte is copied at a time, starting with the first byte, so if there is an overlap between the source and target areas, data may be destroyed. (Also see Intelligent copy command.)

The command can also be used to fill an area of storage with a single value. To do this, make the value yyyy one greater than xxxx. Then put the value into memory at xxxx, and set zzzz to the number of bytes into which the value is to be copied.

E xxxx — EXECUTE COMMAND

The Execute command executes a program, starting at address xxxx.

If address xxxx is not entered, then the stored program counter is used. This means that to continue execution after encountering a breakpoint, it is only necessary to enter E.

G xxxx yyyy zzzz — GENERATE COMMAND

The Generate command writes a cassette tape, which when read back in, loads a program and automatically executes it.

Data from address xxxx up to but not including address yyyy is written to the tape, and zzzz is the address at which execution is to start. Start the tape mechanism before entering the command. The LED is only on during output of the program.

When reading in the tape, there is no need to enter any commands at all. Simply start the tape, and stop it when the program has started execution.

The data on the tape is as follows:-
(CR) E0(CR) R(CR)
then the data in the same format as the Write command, then
Ezzzz (CR)

H — HALF DUPLEX TERMINAL

The H command executes a very simple program in NAS-SYS which waits for input characters, and outputs them. The program can only be ended by pressing Reset.

To make the Nascom act as a half duplex, ASCII, computer terminal, enter the command X 30 and then enter the H command.

I xxxx yyyy zzzz — INTELLIGENT COPY COMMAND

The Intelligent copy command is identical to the Copy command, in that it copies a block of data, length zzzz, from xxxx to yyyy.

However it is intelligent enough to start the copying process at whichever end is required to ensure that data in an overlapping region is never destroyed.

J — JUMP COMMAND

The Jump command starts executing a program at address FFFAH. This is normally the cold start for the 8K BASIC.

K xx — KEYBOARD COMMAND

The Keyboard command sets the method of operation of the keyboard.

K 0 is the normal option. This is the condition following Reset.

K 1 reverses the effect of the SHIFT key on the letters.

K 4 reverses the effect of the GRAPHICS key.

K 5 has the effect of both K 1 and K 4.

L — LOAD COMMAND

The Load command is used to load data stored on paper tape. The data must be in the format used by the Tabulate command for writing a paper tape.

The data loaded is carefully checked by NAS-SYS, and any lines containing errors are scrolled up on the screen, so that they can be reloaded or corrected. All correct lines are displayed for a moment, and then vanish from the screen.

M xxxx — MODIFY COMMAND

The Modify command enables locations in memory to be examined and modified, starting at address xxxx.

The address of the location to be modified is displayed, followed by the current value. This value may be changed, and when the line is entered the new value is set at that location. Several values may be entered on the line, in which case subsequent locations also have their values changed.

Enter `.' at the end of the line to end the Modify command.

Enter `:' at the end of the line to go back to the previous address instead of forward to the next.

Enter `/yyyy' at the end of the line to continue the Modify command at address yyyy.

Normally, hexadecimal values must be entered. However it is also possible to enter ASCII characters directly into memory by entering a comma followed by the character. For example `,A,B,C' would enter ABC into memory.

Invalid values entered are detected, and an error message displayed. In this case the command continues automatically at the address at the start of the last line.

N — NORMAL COMMAND

The Normal command resets the pointers to the input and output tables to their normal values. This has the effect of turning off an X or U command. All input will then be from the Nascom keyboard or the serial input, and output will be only to the screen.

O xx yy — OUTPUT COMMAND

The Output command sends data to a port. The value yy is sent to the port xx. For example `O 7 F' would send 0FH to port 07H. To learn how to use the parallel ports, read the PIO technical manual.

Q xx — QUERY COMMAND

The Query command obtains the value from a port and displays it in hexadecimal. The port queried is the port xx. To learn how to use the parallel ports, read the PIO technical manual.

R — READ COMMAND

The Read command reads a cassette tape which was written by the Write command. See the Write command for the format of the data.

As each block of data is read, the header information is displayed:-

SSSS BBLL .

SSSS is the start address of the data, BB is the block number, and LL is the length (0=256).

After the last block, block 00, has been read correctly, the Read command automatically ends. During the command, the LED is switched on.

The start of each block is recognised by reading the four start of block characters. All data is ignored until the start of a block. If the checksum for the header data is incorrect, then a question mark is displayed, and the program waits for the start of the next block. The data following is not loaded.

If the checksum for the actual data loaded is incorrect, then instead of a full stop, a question mark is displayed, and the program waits for the start of the next block. In this case, invalid data will have been loaded, but only at the correct locations. This technique eliminates the need for a buffer.

Do not press keys on the keyboard since this will cause errors. (Which will be detected.)

A visual check of the display is required to ensure that all blocks have been loaded correctly. If any errors are encountered, simply rewind the tape for about two blocks and carry on.

To stop the Read command in the middle, press ESC (SHIFT/ENTER) four times. This only works between blocks, so if necessary first press any keys until end of block is reached.

S xxxx — SINGLE STEP COMMAND

The Single Step command executes a single instruction of a program, at address xxxx.

If address xxxx is not specified, then the stored program counter is used. This means that to execute a single step after encountering a breakpoint, it is only necessary to enter S.

If the previous command was a Step command, then even S need not be entered.

After the instruction has been executed, the program registers are saved, and are also displayed.

Note that it is not possible to single step through certain parts of NAS-SYS itself, including the execution of the input and output routines. It is best to set a breakpoint immediately after any call to a NAS-SYS routine, and enter E to perform the routine. Single stepping can then continue in the user program. This is more productive in any case, since NAS-SYS should not contain any errors!

T xxxx yyyy zzzz — TABULATE COMMAND

The Tabulate command displays a block of memory, starting at address xxxx and continuing up to but not including address yyyy. Each line shows the address followed by the values of the eight bytes starting at that location. zzzz lines are displayed at a time. Press any key to display the next group of lines, or press ESC or SHIFT/ENTER to end the command.

By setting zzzz to Ø, all of the data will be output without a pause. By entering the command X Ø and then entering the Tabulate command with zzzz set to Ø, a continuous block of data is transmitted to the serial output port, where it can be punched onto paper tape. A checksum is output at the end of each line. The Load command can be used to load this data back into the computer.

U — USER I/O COMMAND

The User I/O command activates user specified input and output routines, in addition to the normal keyboard input and screen output. These routines can for examole control a parallel printer. The Normal command can be used to turn these routines off again.

The user output routine must be pointed to by an address at ØC78H, and the user input routine must have its address stored at ØC7BH. On Reset these addresses point to a Return instruction within NAS-SYS, so if these addresses are not altered, entering the U command will have no effect.

The U command is automatically deactivated during execution of Load, Read, Write and Generate commands, and reactivated afterwards.

V — VERIFY COMMAND

The Verify command is identical to the Read command, except that the data read from the cassette tape is not loaded into memory. The purpose of the command is to check that a tape can be read without error.

W xxxx yyyy — WRITE COMMAND

The Write command writes data to a cassette tape. Data from address xxxx up to but not including address yyyy is written to the tape.

Data is output in blocks, each of 256 bytes, except the last block, which may have less. The format of each block is as follows:-

ØØ	Null (Ø).
FF FF FF FF	Four start of block characters (FFH).
SS SS	Start address, low order first.
LL	Length of data (Ø=256).
BB	Block number. This is one less for each block. The last block is block ØØ.
CC	Checksum for the header data.
DD DD	Data.
EE	Checksum for the data.
ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ ØØ	Ten nulls (Ø).

As each block is written, the header data, consisting of the start address, block number and length are displayed as follows:-

SSSS BBLL

When the command is entered, the LED is switched on, there is a brief delay, 256 nulls are output, and then each block is output. At the end, the LED is switched off and the command is ended.

The extra nulls at the end of each block ensure that even if several characters are lost in a block, the next block can still be read correctly. The extra null before the start of each block ensures that an initial spurious start of block character is ignored. The 256 nulls at the start ensure that error correction is always possible merely by rewinding the tape and playing it again.

X xx — EXTERNAL COMMAND

The External command activates input and output routines contained within NAS-SYS, which provide comprehensive capabilities for communication with external devices such as ASCII terminals (e.g. Teletype), and mainframe computers, through the serial input/output ports. The Normal command can be used to turn these routines off again.

The value entered after the X specifies the X option, as follows:-

Ø	Support a terminal in full duplex mode. Every character typed is sent back to the terminal. Line feed is automatically supplied after carriage return. All output is even parity.
1Ø	Same as Ø, but line feed is not supplied.
2Ø	Same as Ø, but half duplex. Characters entered are not sent back. However, line feed is always supplied after carriage return.
3Ø	Same as 2Ø, but line feed is not supplied. This option makes the Nascom into a half duplex terminal.
1, 11, 21, 31	Same as Ø, 1Ø, 2Ø, 3Ø, respectively, but the output parity is odd instead of even. (Input parity is always ignored.)

To store data on paper tape, use the X command (usually option Ø), then the Tabulate command. Then use the Load command to load this data back into the computer.

When a character is received from the terminal, unless it is Escape (1BH), Null (Ø), or Del (7FH), it is assumed that the program will try to output the character. Therefore an indicator is set so that the next character output by the program is not sent to the serial output.

To input a Backspace on a terminal which does not have this key, use CTRL/H.

The X command is automatically deactivated during execution of Load, Read, Write and Generate commands, and reactivated afterwards.

The X command also activates the U command output routine, if one has been supplied. (See the U command.) This makes it very easy to have the Nascom keyboard and display, and a Teletype keyboard and printer, and a parallel printer, all working at the same time.

Z — Z COMMAND

The Z command starts executing a program at address FFFDH. This is normally the warm start for the 8K Basic.

HOW TO ENTER AND TEST A PROGRAM

1. Write out the program on paper, in Z80 Assembler language. You will need to refer to the Z80 programming manual, and the sections of this manual describing the NAS-SYS facilities you can use for input and output, and how to end the program. There are also many useful routines in NAS-SYS which you may wish to make use of.

2. Then you can assemble the program by hand. This means that you must convert each instruction into machine code. The Z80 programming manual gives all the codes. The NAS-SYS Arithmetic command makes it easy to calculate the values to put into relative jump instructions.

3. If you have ZEAP, the Nascom Editor and Assembler package, then you can type in your program in Assembler, and ZEAP will generate all the machine code for you.

4. If you don't have ZEAP, then use the Modify command to enter the program into memory.

5. Since there might be an error in the program, and this error might change any part of memory and quite possibly wipe out the program or corrupt it, it is wise to save the program on tape before starting to test it. Use the Write command to write the program to cassette tape. Unless you are confidant that your tape recorder and tape are working properly, use the Verify command to check that the tape can be read back.

6. To test the program, use the Execute command to run it. You can set a breakpoint first by using the Breakpoint command. You can also use the Single Step command to execute small parts of the program which are causing trouble. You can also use the Modify or Tabulate commands to examine the program and areas of memory used by the program, after a breakpoint or during single stepping.

7. If you crash the program, you can always use the Reset button to regain control of the computer, and then use the Read command to reload the program for another attempt.

8. Although Assembler and machine code programming may seem slow and difficult at first, it is also fascinating, and it provides the only way to get the most out of the machine, in terms of both speed of execution of programs, and also in the depth of understanding which you will gain about the Z80 microprocessor and computing in general. Good luck!

DISPLAY OF PROGRAM REGISTERS

When the Single Step command is used, or when a breakpoint set by the Breakpoint command is encountered, or if code E7H (RST 20H) is executed in a program, the program registers are displayed as follows:-

```
-SP- -PC- -AF- -HL- -DE- -BC- I  -IX- -IY- Flags
```

The flags are a decoded representation of the F register. The following characters may be displayed, indicating which flag bits have been set:-

```
S Z H P N C
```

The register display is often an essential aid when debugging a program. The Program Counter (PC) shows the address of the next instruction to be executed, and the Stack Pointer (SP) shows the position on the stack. The other registers show the effect of the program instructions on them. When the registers have been displayed, it is often necessary to investigate the actions of the program in more detail, by using the Modify command to determine the contents of various memory locations. For example you can find what is on the stack, or what the memory locations pointed to by the HL register contain.

SIMPLE INSTRUCTIONS FOR INPUT AND OUTPUT

NAS-SYS uses a most powerful method of controlling input and output. This is described in the section `Input and Output'.

However, to simply output the contents of the A register to the screen, enter the code F7H (RST 30H) in your program. You simply type in F7 as part of your program.

To simply wait for an input character from the keyboard or from the serial input, enter the code CFH (RST 8) in your program. You simply type in CF as part of your program. The input character will be returned in the A register.

In both cases, no other registers will be affected.

Even if you use only this simple method provided by NAS-SYS for input and output, you can use the X or U commands to control terminals and printers. See the descriptions of these commands.

HOW TO END A PROGRAM

One of the following methods should be used to end each of your programs - and no other!

1. Press the Reset button at any time, to restart the system. A HALT instruction (76H) may be placed in the program to make the Halt LED light, to indicate that the program has finished, and then you can press the Reset button to continue. This is a very primitive solution.

2. Execute code RST 0 (C7H) in the program. This is equivalent to pressing the Reset button. Like the first method this is simple but not very clever. Both these methods have the disadvantage that the screen is cleared so that you can't see what was output before the program ended.

3. The normal and recommended method of ending a program is to execute code DF 5B in the program. This provides a controlled return to NAS-SYS, so that you can enter commands. (The program registers are not saved, and the user stack pointer is set back to 1000H.)

4. The method of ending a program at a place which it shouldn't ever get to is to execute code RST 20H (E7H) in the program. This stores the program registers and displays them, before returning control to NAS-SYS. This can be useful to indicate an abnormal end of program, as well as providing useful information for debugging. Execution of the program may then be continued by entering an Execute or Single Step command, but this command must specify the address at which execution is to start.

5. Generate a Non-Maskable Interrupt (NMI). If the computer has a hardware feature to allow the user to generate a single NMI by pushing a button, then this will have the same effect as method 4 above, except that execution can be continued by simply entering E.

NAS-SYS RESTARTS AND ROUTINES

NAS-SYS provides many useful routines which can be called by user programs. These fall into three categories:-

1. Restart instructions. These functions are called by using the one byte Z80 RST instructions.

2. Routines which are called by special two byte instructions which in fact consist of a RST 18H (DFH) instruction followed by a number which indicates the routine to be called. NAS-SYS routines should always be called by this method and never by a CALL instruction, because if NAS-SYS was changed, the routine numbers would be the same, but the absolute addresses would be different.

3. There is one routine which is called by a normal CALL instruction. This routine allows another program to perform NAS-SYS initialisation without losing control.

The routines in each of these three categories will now be described in detail, except for the low level input and output routines, which are described in the section `Input and Output'.

NAS-SYS RESTART INSTRUCTIONS

CODE	ASSEMBLER	NAME	FUNCTION
C7	RST 0	START	Reset computer. Initialise NAS-SYS.
CF	RST 8	RIN	Obtain an input character in the A register.
D7	RST 10H	RCAL	Relative Call. Follow this code with the displacement to the routine to be called. This is similar to the Z80 Jump Relative instruction, and it allows relocatable code to be written.
DF	RST 18H	SCAL	Subroutine Call. Follow this code with the number of the routine to be called. This is the method used to call the NAS-SYS routines. See the next section.
E7	RST 20H	BRKPT	Store and display the program registers, then return control to NAS-SYS. This is used by the Breakpoint command.
EF	RST 28H	PRS	Output the string of characters following this code, until a 0 is encountered. Then continue execution with the next instruction. This provides a very simple way of displaying a message. The A register is set to 0.
F7	RST 30H	ROUT	Output the character in the A register.
FF	RST 38H	RDEL	Wait for a period of time dependent on the value in the A register. A is set to 0.

NAS-SYS ROUTINES

These routines are called simply by putting the codes listed beside them into your program. For example routine ERRM outputs an error message. So to output an error message, simply put the codes DF 6B into your program.

It is also possible to call the routines which are the NAS-SYS commands. To do this use the code DF followed by the ASCII code for the letter of the command. Registers HL, DE and BC should be set to the (up to three) values which would normally be typed in after the command. For example codes DF 57 would call the NAS-SYS Write command routine. You would need to set HL to the start of the data to be written, and DE to the next address after the end of the data, before calling the routine. You may need to examine the listing of NAS-SYS to learn exactly how the commands work.

You might want to define your own set of commands and routines, or add functions that are not in NAS-SYS. You can do this because the address of the start of the table of routine addresses is stored at address 0C71-0C72. You can change this value to point to your own table of routine addresses. Examine the listing of NAS-SYS to learn how it works.

Before the list of the routines you can call, here is the description of a special routine which is called at address 000D, by the codes CD 0D 00. This routine, called STMON, reinitialises NAS-SYS and clears the screen. The point is that if the computer is designed so that on Reset it starts execution at an address other than 0, then the program which starts can set the Stack Pointer and then call STMON. It can then continue just as if it had been executed from NAS-SYS, and can use the NAS-SYS routines.

The listing of the NAS-SYS routines follows. You should find them easy to incorporate into your programs, and they should allow you to develop working programs more quickly.

CODES	NAME	FUNCTION
DF 5B	MRET	This is not a normal routine, but is instead used to end a program and return control to NAS-SYS. See `How to end a program´.
DF 5C	SCALJ	This is not a normal routine, but is instead used to call any other routine, when the routine number is not known when the program is written. Store the routine number at address ARGC (ØCØA) and then execute code DF 5C. The routine will be called for you.
DF 5D	TDEL	Wait for about 1 second (at 4 MHz). Registers A and B are set to Ø.
DF 5E	FFLP	Flip one or more bits of output port Ø, then immediately flip them back again. Register A must have a 1 in each bit position to be flipped. A is modified.
DF 5F	MFLP	Alter the state of (turn on or off) the tape drive LED. Register A is modified.
DF 6Ø	ARGS	Load the contents of ARG1 into HL, ARG2 into DE and ARG3 into BC. ARG1, 2 and 3 are the first three values entered after a NAS-SYS command. You could use this routine to pick up extra values typed in as part of the E command used to run the program.
DF 62	IN	Scan for an input character. Instead of waiting for an input, like code CF (RST 8), it just checks to see if there has been an input. If there has been, then the Carry flag is set and the character is in A. The A register is modified.
DF 63	INLIN	Obtain an input line. This is the routine used by NAS-SYS to get commands. It provides a blinking cursor and waits for `Enter´ or `New Line´ to be pressed. The DE register is set to the address of the start of the line where the cursor was when when the line was entered. The A register is modified.

CODES	NAME	FUNCTION
DF 64	NUM	Examine an input line and convert a hexadecimal value from ASCII to binary. Set DE to point to the start of the line. Leading blanks are ignored. The value is ended by a blank or null (Ø). DE is returned pointing to the next position. If the value is invalid (not Ø-9, A-F, or >FFFFH), then the Carry flag is set, and DE points to the invalid character. The resulting value is placed in NUMV (ØC21-ØC22) and the number of characters in the ASCII value is placed in NUMN (ØC2Ø). The HL and A registers are modified.
DF 66	TBCD3	Output the value in the HL register in ASCII, followed by a space. Also add H and L into the C register. The A register is modified.
DF 67	TBCD2	Output the value in the A register in ASCII. Also add A into the C register. The A register is modified.
DF 68	B2HEX	Output the value in the A register in ASCII. The A register is modified.
DF 69	SPACE	Output a space. The A register is set to a space.
DF 6A	CRLF	Output a Carriage return/line feed. A is set to a CR.
DF 6B	ERRM	Output the message `Error' followed by a CR. A is set to a CR.
DF 6C	TX1	Output HL in ASCII, then a space, then DE, then another space. Also, H, L, D and E are added into the C register. Register A is modified.
DF 6D	SOUT	Send a string of characters directly to the serial output port. HL must be set to point to the start of the string, and B must be set to the length. The C register is set to Ø and then all the characters are added into it. Registers HL, B and A are also modified.

CODES	NAME	FUNCTION
DF 79	RLIN	Examine an input line and convert up to ten hexadecimal values separated by spaces from ASCII to binary. Set DE to point to the start of the line. The line must end with a null (Ø) character. RLIN uses the NUM routine to convert the values. If an invalid value is found or if there are more than ten values, the Carry flag is set. ARGN (ØCØB) is set to the number of values found. ARG1 (ØCØC-ØCØD) to ARG1Ø (ØC1E-ØC1F) are set to the values. NAS-SYS uses this routine to get the values from commands entered. The HL, DE, BC and A registers are modified.
DF 7A	B1HEX	Output the low order (right) half of the A register in ASCII. The A register is modified.
DF 7B	BLINK	Obtain an input character in the A register. While waiting for the input, blink the cursor on the screen. The display is not modified. Registers HL and DE are modified.
DF 7C	CPOS	The HL register must be set to point to a position on the screen. Then the routine is called. It sets HL to point to the address of the first character on that line on the screen. Registers E and A are modified.

INPUT AND OUTPUT

This section describes the powerful method which NAS-SYS uses to control input and output, so that you can make best use of it. However, it is not necessary to read or understand this section to make extensive use of NAS-SYS and its input and output facilities.

The descriptions of the N, U and X commands show you how to contol terminals and printers using NAS-SYS. The NAS-SYS Restart instructions RIN, ROUT and PRS enable you to input characters and output single characters and messages. The NAS-SYS routines IN, BLINK and INLIN provide various input options, and TBCD3, TBCD2, B2HEX, B1HEX, SPACE, CRLF, ERRM and TX1 provide several easy ways to output data. (NUM and RLIN go a stage further and help you to get data from input lines.)

So at this point you should know how to specify input and output in your program, and how to control where this data is to go to by the N, U and X commands.

However, you can control where the data is to come from for input and go to for output, much more flexibly, if you understand the following description of how NAS-SYS works.

Both input and output work the same way, as follows:-

Every time an input or output is requested, a special routine (called ATE) in NAS-SYS calls each of a number of input/output handling routines in turn. Like all other NAS-SYS routines, these have routine numbers. There is a table of routine numbers for the input routines, and another for the output routines. Each of these tables is ended by a null (Ø). The address of the input table is stored at $IN (ØC75-ØC76), and the address of the output table is stored at $OUT (ØC73-ØC74).

As usual in NAS-SYS, the routine numbers are converted to actual addresses by referring to the table of subroutine addresses. The address of the start of this table is stored at $STAB (ØC71-ØC72).

The ATE routine which calls each of the input/output routines in turn, automatically preserves the HL, DE and BC registers. Output routines must preserve the AF register. Input routines must return with the Carry flag set and the input character in the A register if there is an input, otherwise the Carry flag must be reset.

The way that the NAS-SYS N, U and X commands work is to change the address of the input and output tables, at $IN and $OUT. NAS-SYS contains alternative input/output tables and other input and output routines for this purpose.

Now that you know how the input and output works, you will see that it is possible to set up your own tables of routine numbers, and then change the addresses at $IN and $OUT to point to your tables. You may also wish to add your own routines, and this is done most easily by using the jump instructions provided for the U command. NAS-SYS provides some routines to make it easier for you to change the addresses of the tables. These are as follows:-

CODES	NAME	FUNCTION
DF 71	NOM	Set HL to the address of the new output table, then call this routine. It changes the address for you, and returns with the previous address in HL.
DF 72	NIM	Set HL to the address of the new input table, then call this routine. It changes the address for you, and returns with the previous address in HL.
DF 77	NNOM	Set the output table back to normal. It returns with the previous address in HL.
DF 78	NNIM	Set the input table back to normal. It returns with the previous address in HL.

You now need to know the routine numbers to put in your tables. Remember to put a 0 at the end of each table. The routine numbers are as follows:-

INPUT ROUTINES

CODE	NAME	FUNCTION
61	KBD	Scan Nascom keyboard.
70	SRLIN	Scan serial input port.
74	XKBD	Scan external ASCII keyboard. (See X command.)
76	UIN	User specified input routine. (See U command.)

OUTPUT ROUTINES

CODE	NAME	FUNCTION
65	CRT	Display on Nascom screen. See 'Screen Editing' for details of facilities.
6F	SRLX	Output to serial output port.
6E	XOUT	Output to external ASCII device. (See X command.)
75	UOUT	Output to user specified output routine. (See U command.)

NAS-SYS WORKSPACE

NAS-SYS requires an area of memory to use as a workspace. This occupies locations ØC00 to ØC7F, so the first free area for user programs is at ØC80. The first part of the workspace, from ØC00 to ØC6A is initialised to Ø on reset. The second part, from ØC6B to ØC7D is initialised from a table in NAS-SYS. ØC7E to ØC7F is also initialised by NAS-SYS. User programs may choose to make use of or alter certain values in the workspace, so the following table is included to supplement the listing of NAS-SYS. The table shows the address and length of each value, in hexadecimal, and its name and function.

ADDR	LEN	NAME	FUNCTION
ØC00	1	PORTØ	Copy of current state of output port Ø.
ØC01	9	KMAP	Map of state of keyboard.
ØC0A	1	ARGC	Command letter or routine number last processed for command execution or input/output.
ØC0B	1	ARGN	Number of values in input line.
ØC0C	2	ARG1	First value entered.
ØC0E	2	ARG2	Second value entered.
ØC10	2	ARG3	Third value entered.
ØC12	ØC	ARG49	Fourth to ninth values entered.
ØC1E	2	ARG10	Tenth value entered.
ØC20	1	NUMN	Number of characters in value examined by routine NUM.
ØC21	2	NUMV	Value returned by routine NUM.
ØC23	2	BRKADR	Breakpoint address.
ØC25	1	BRKVAL	Stored value from the breakpoint address.
ØC26	1	CONFLG	Normally Ø, but set to -1 for the E command.
ØC27	1	$KOPT	Keyboard option. See K command.
ØC28	1	$XOPT	X option. See X command.
ØC29	2	CURSOR	Position of the cursor.
ØC2B	1	ARGX	Last command letter entered.
ØC2C	35	MONSTK	NAS-SYS stack.
ØC61	2	RBC	Register save area. Register BC.
ØC63	2	RDE	Register save area. Register DE.
ØC65	2	RHL	Register save area. Register HL.
ØC67	2	RAF	Register save area. Register AF.
ØC69	2	RPC	Register save area. Program counter.
-------Initialised by table from here on------			
ØC6B	2	RSP	Register save area. Stack pointer.
ØC6D	2	$KTABL	Length of keyboard table.
ØC6F	2	$KTAB	Start of keyboard table.
ØC71	2	$STAB	Start of table of routine addresses, for routine ØØ. Since the first routine is in fact number 41H, the actual table starts 82H beyond this address.
ØC73	2	$OUT	Start of table of output routines.
ØC75	2	$IN	Start of table of input routines.
ØC77	3	$UOUT	Jump to user specified output routine.
ØC7A	3	$UIN	Jump to user specified input routine.
ØC7D	3	$NMI	Jump to NMI routine. NAS-SYS sets this to display the registers.

ØC2C KCNT
ØC2E KLONG
ØC30 KSHORT

ADDRESSING OF VIDEO RAM

The video RAM is addressed as shown in the diagram below. The top line is addressed after the other 15, and it is not scrolled by NAS-SYS. There is a 10 byte margin on the left of each line, then a 48 byte line, then a 6 byte margin on the right. When NAS-SYS clears the screen, all visible locations are made blank, and all the margins are set to nulls (0), except for the first 10 bytes, which are to the left of line 1, and the last 6 bytes, which are to the right of line 16, the top line.

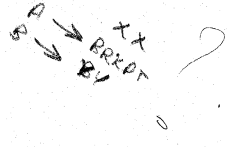
LINE NO.	LEFT MARGIN	LEFT SIDE		RIGHT SIDE	RIGHT END
16	0BC0	0BCA	0BCB	0BF9	0BFF
1	0800	080A		0839	083F
2	0840	084A		0879	087F
3	0880	088A		08B9	08BF
4	08C0	08CA		08F9	08FF
5	0900	090A		0939	093F
6	0940	094A		0979	097F
7	0980	098A		09B9	09BF
8	09C0	09CA		09F9	09FF
9	0A00	0A0A		0A39	0A3F
10	0A40	0A4A		0A79	0A7F
11	0A80	0A8A		0AB9	0ABF
12	0AC0	0ACA		0AF9	0AFF
13	0B00	0B0A		0B39	0B3F
14	0B40	0B4A		0B79	0B7F
15	0B80	0B8A	0B8B	0BB9	0BBF
		LEFT SIDE		RIGHT SIDE	

```
                        0005 ; NAS-SYS 1
                        0010 ;
0008                    0015 BS     EQU  #08
000A                    0020 LF     EQU  #0A
000C                    0025 CS     EQU  #0C
000D                    0030 CR     EQU  #0D
0011                    0035 CUL    EQU  #11
0012                    0040 CUR    EQU  #12
0013                    0045 CUU    EQU  #13
0014                    0050 CUD    EQU  #14
0015                    0055 CSL    EQU  #15
0016                    0060 CSR    EQU  #16
0017                    0065 CH     EQU  #17
0018                    0070 CCR    EQU  #18
001B                    0075 ESC    EQU  #1B
005F                    0080 CU     EQU  #5F
007F                    0085 DEL    EQU  #7F
                        0090 ;
0000                    0095 ROM    EQU  #0
FFFA                    0100 BPRC   EQU  #FFFA
FFFD                    0105 BPRW   EQU  #FFFD
                        0110 ;
                        0115 ; VIDEO RAM
0800                    0120 VRAM   EQU  #0800
080A                    0125 VL1    EQU  VRAM+10
084A                    0130 VL2    EQU  VL1+64
0B8A                    0135 VL15   EQU  VRAM+#038A
0C00                    0140 VEND   EQU  VRAM+#400
                        0145 ;
                        0150 ; WORKSPACE RAM
0C00                    0155 RAM    EQU  #0C00
1000                    0160 RAMTOP EQU  #1000
                        0165 ;
                        0170 ; WORKSPACE
                        0175 ;  INITIALISED TO 0
0C00                    0180 INITZ  ORG  RAM
                        0185 ; COPY OF PORT 0
0001                    0190 PORT0  DEFS 1
                        0195 ; KEYBOARD STATUS MAP
0009                    0200 KMAP   DEFS 9
                        0205 ; COMMAND CHAR
0001                    0210 ARGC   DEFS 1
                        0215 ; NO OF ARGS
0001                    0220 ARGN   DEFS 1
                        0225 ; UP TO 10 ARGS
0002                    0230 ARG1   DEFS 2
0002                    0235 ARG2   DEFS 2
0002                    0240 ARG3   DEFS 2
000C                    0245 ARG49  DEFS 12
0002                    0250 ARG10  DEFS 2
                        0255 ; NO OF CHARS IN HEX VALUE
0001                    0260 NUMN   DEFS 1
                        0265 ; HEX VALUE ENTERED
0002                    0270 NUMV   DEFS 2
                        0275 ; BPT ADDRESS
0002                    0280 BRKADR DEFS 2
                        0285 ; BPT VALUE
0001                    0290 BRKVAL DEFS 1
                        0295 ; CONFLG NOT 0 IF E COMMAND
0001                    0300 CONFLG DEFS 1
                        0305 ; K OPTION
0001                    0310 $KOPT  DEFS 1
                        0315 ; X OPTION
0001                    0320 $XOPT  DEFS 1
                        0325 ; CURSOR POSITION
0002                    0330 CURSOR DEFS 2
                        0335 ; LAST COMMAND
0001                    0340 ARGX   DEFS 1
                        0345 ; MONITOR STACK
0035                    0350 MONSTK DEFS #35
0C61                    0355 STACK  EQU  $
                        0360 ; REGISTER SAVE AREA
0002                    0365 RBC    DEFS 2
0002                    0370 RDE    DEFS 2
0002                    0375 RHL    DEFS 2
0002                    0380 RAF    DEFS 2
0002                    0385 RPC    DEFS 2
                        0390 ;
                        0395 ; WORKSPACE
                        0400 ;  INITIALISED BY TABLE
0C6B                    0405 INITR  EQU  $
                        0410 ; USER SP
0002                    0415 RSP    DEFS 2
                        0420 ; END OF REG SAVE AREA
0C6D                    0425 RSAE   EQU  $
                        0430 ; LENGTH OF KTAB
0002                    0435 $KTABL DEFS 2
                        0440 ; ADDRESS OF KTAB
0002                    0445 $KTAB  DEFS 2
                        0450 ; ADDRESS OF STAB
0002                    0455 $STAB  DEFS 2
                        0460 ; OUTPUT TABLE
0002                    0465 $OUT   DEFS 2
                        0470 ; INPUT TABLE
0002                    0475 $IN    DEFS 2
```

```
                0480 ; USER JUMPS
0003            0485 $UOUT  DEFS 3
0003            0490 $UIN   DEFS 3
                0495 ; NMI JUMP
0003            0500 $NMI   DEFS 3
                0505 ;
                0510 ; END OF WORKSPACE
0C80            0515 INITE  EQU  $
                0520 ;
                0525 ;
                0530 ; START OF MONITOR
0000            0535 START  ORG  ROM
0000 310010     0540        LD   SP,RAMTOP
0003 D7         0545        RST  RCAL
0004 08         0550        DEFB STMON-$-1
0005 C3B203     0555        JP   MRET
                0560 ;
                0565 ;
                0570 ; GET INPUT
0008 DF         0575 RIN    RST  SCAL
0009 62         0580        DEFB ZIN
000A D8         0585        RET  C
000B 18FB       0590        JR   RIN
                0595 ;
                0600 ;
                0605 ; INITIALISE MONITOR
000D C39A03     0610 STMON  JP   STRTB
                0615 ;
                0620 ;
                0625 ; RELATIVE CALL
0010 E5         0630 RCAL   PUSH HL
0011 E1         0635        POP  HL
                0640 ; INC RET ADDRESS
0012 E1         0645        POP  HL
0013 23         0650        INC  HL
0014 E5         0655        PUSH HL
0015 C37005     0660        JP   RCALB
                0665 ;
                0670 ;
                0675 ; SUBROUTINE CALL
0018 E5         0680 SCAL   PUSH HL
0019 E1         0685        POP  HL
                0690 ; INC RET ADDRESS
001A E1         0695        POP  HL
001B 23         0700        INC  HL
001C E5         0705        PUSH HL
001D C38005     0710        JP   SCALB
                0715 ;
                0720 ;
                0725 ; BPT
                0730 ; DECREMENT PC ON STACK

0020 E3         0735 BRKPT  EX   (SP),HL
0021 2B         0740        DEC  HL
0022 E3         0745        EX   (SP),HL
0023 C31A04     0750        JP   TRAP
0026 00         0755        DEFB 0,0; FILL
                0760 ;
                0765 ;
                0770 ; OUTPUT A STRING
0028 E3         0775 PRS    EX   (SP),HL
0029 7E         0780 PRS1   LD   A,(HL)
002A 23         0785        INC  HL
                0790 ; OUTPUT UNLESS 0
002B B7         0795        OR   A
002C 2006       0800        JR   NZ,PRS2
002E E3         0805        EX   (SP),HL
002F C9         0810 PRET   RET
                0815 ;
                0820 ;
                0825 ; OUTPUT A CHAR
0030 E5         0830 ROUT   PUSH HL
0031 C35A07     0835        JP   AOUT
                0840 ;
                0845 ;
                0850 ; MORE OF PRS
0034 F7         0855 PRS2   RST  ROUT
0035 18F2       0860        JR   PRS1
0037 00         0865        DEFB 0; FILL
                0870 ;
                0875 ;
                0880 ; DELAY
0038 3D         0885 RDEL   DEC  A
0039 C8         0890        RET  Z
003A F5         0895        PUSH AF
003B F1         0900        POP  AF
003C 18FA       0905        JR   RDEL
                0910 ;
                0915 ;
                0920 ; DELAY
003E AF         0925 TDEL   XOR  A
003F 47         0930        LD   B,A
0040 FF         0935 TDEL2  RST  RDEL
0041 FF         0940        RST  RDEL
0042 10FC       0945        DJNZ TDEL2
0044 C9         0950        RET
                0955 ;
                0960 ;
                0965 ; SET, RESET BIT IN P0
0045 E5         0970 FFLP   PUSH HL
0046 21000C     0975        LD   HL,PORT0
0049 AE         0980        XOR  (HL)
004A D300       0985        OUT  (0),A
```

when it returns from TRAP it returns to A instead of B



```
004C 7E       0990        LD   A,(HL)
004D D300     0995 FF2    OUT  (0),A
004F E1       1000        POP  HL
0050 C9       1005        RET
              1010 ;
              1015 ; FLIP BIT 4 IN P0
0051 3E10     1020 MFLP   LD   A,#10
0053 E5       1025        PUSH HL
0054 21000C   1030        LD   HL,PORT0
0057 AE       1035        XOR  (HL)
0058 77       1040        LD   (HL),A
0059 18F2     1045        JR   FF2
              1050 ;
              1055 ;
              1060 ; SERIAL OUTPUT TO P1
005B F5       1065 SRLX   PUSH AF
005C D301     1070        OUT  (1),A
              1075 ; WAIT UNTIL OUTPUT
005E DB02     1080 SRL4   IN   A,(2)
0060 CB77     1085        BIT  6,A
0062 28FA     1090        JR   Z,SRL4
0064 F1       1095        POP  AF
0065 C9       1100        RET
              1105 ;
              1110 ;
              1115 ; NMI RESTART
0066 C37D0C   1120 RNMI   JP   $NMI
              1125 ;
              1130 ;
              1135 ; GET INPUT
0069 1EC0     1140 BIN    LD   E,NC0
006B DF       1145 BIN2   RST  SCAL
006C 62       1150        DEFB ZIN
006D D8       1155        RET  C
006E 1D       1160        DEC  E
006F 20FA     1165        JR   NZ,BIN2
0071 C9       1170        RET
              1175 ;
              1180 ;
              1185 ; BLINK UNTIL INPUT
0072 2A290C   1190 BLINK  LD   HL,(CURSOR)
0075 56       1195        LD   D,(HL)
0076 365F     1200        LD   (HL),CU
0078 D7       1205        RST  RCAL
0079 EF       1210        DEFB BIN-$-1
007A 72       1215        LD   (HL),D
007B D8       1220        RET  C
007C D7       1225        RST  RCAL
007D EB       1230        DEFB BIN-$-1
007E 30F2     1235        JR   NC,BLINK
0080 C9       1240        RET
              1245 ;
              1250 ;
              1255 ; CHECK SERIAL IN
0081 DB02     1260 SRLIN  IN   A,(2)
0083 17       1265        RLA
0084 D0       1270        RET  NC
0085 DB01     1275        IN   A,(1)
0087 C9       1280        RET
              1285 ;
              1290 ;
              1295 ; KEYBOARD ROUTINE
              1300 ; RESET KBD COUNTER
0088 3E02     1305 KBD    LD   A,2
008A CD4500   1310        CALL FFLP
              1315 ; STORE ROW 0 IN MAP
008D 21010C   1320        LD   HL,KMAP
0090 DB00     1325        IN   A,(0)
0092 2F       1330        CPL
0093 77       1335        LD   (HL),A
              1340 ;
              1345 ; SCAN 8 ROWS
0094 0608     1350        LD   B,8
              1355 ; INC KBD COUNTER
0096 3E01     1360 KSC1   LD   A,1
0098 CD4500   1365        CALL FFLP
009B 23       1370        INC  HL
              1375 ; GET ROW STATUS
009C DB00     1380        IN   A,(0)
009E 2F       1385        CPL
009F 57       1390        LD   D,A
              1395 ; IF MAP DIFFERENT
              1400 ;  FIND OUT WHY
00A0 AE       1405        XOR  (HL)
00A1 2004     1410        JR   NZ,KSC2
              1415 ; SCAN NEXT ROW
00A3 10F1     1420 KSC1A  DJNZ KSC1
              1425 ; NO KEY PRESSED
00A5 B7       1430 KSC8   OR   A
00A6 C9       1435        RET
              1440 ;
              1445 ; WAIT, TO DEBOUNCE
00A7 AF       1450 KSC2   XOR  A
00A8 FF       1455        RST  RDEL
              1460 ; GET ROW AGAIN
00A9 DB00     1465        IN   A,(0)
00AB 2F       1470        CPL
00AC 5F       1475        LD   E,A
              1480 ; E = NEW STATE
00AD 7A       1485        LD   A,D
              1490 ; A = OLD STATE
00AE AE       1495        XOR  (HL)
```

```
                1500 ; A = CHANGES
                1505 ; FIND CHANGED BIT
00AF 0EFF       1510        LD   C,-1
00B1 1600       1515        LD   D,0
00B3 37         1520        SCF
00B4 CB12       1525 KSC4   RL   D
00B6 0C         1530        INC  C
00B7 1F         1535        RRA
00B8 30FA       1540        JR   NC,KSC4
                1545 ; C = COL CHANGED
                1550 ; D = MASK WITH 1 AT CHANGE
00BA 7A         1555        LD   A,D
00BB A3         1560        AND  E
00BC 5F         1565        LD   E,A
                1570 ; E = NEW STATE,
                1575 ;  MASKED BY CHANGE
                1580 ; IF MAP STATE AND NEW
                1585 ;  STATE EQUAL, IGNORE
00BD 7E         1590        LD   A,(HL)
00BE A2         1595        AND  D
00BF BB         1600        CP   E
00C0 28E1       1605        JR   Z,KSC1A
                1610 ; UPDATE MAP
00C2 7E         1615        LD   A,(HL)
00C3 AA         1620        XOR  D
00C4 77         1625        LD   (HL),A
                1630 ; IF NEW STATE IS 0, THEN
                1635 ;  KEY RELEASED, SO IGNORE
00C5 7B         1640        LD   A,E
00C6 B7         1645        OR   A
00C7 28DA       1650        JR   Z,KSC1A
                1655 ;
                1660 ; VALUE = SRRRRCCC
                1665 ;  S=1 IF SHIFT
                1670 ;  RRRR=9-ROW NUMBER
                1675 ;  CCC=COLUMN NUMBER
00C9 3A010C     1680        LD   A,(KMAP)
00CC E610       1685        AND  #10
00CE B0         1690        OR   B
00CF 87         1695        ADD  A,A
00D0 87         1700        ADD  A,A
00D1 87         1705        ADD  A,A
00D2 B1         1710        OR   C
                1715 ;
                1720 ; SEARCH TABLE
00D3 D7         1725        RST  RCAL
00D4 5B         1730        DEFB KSE-$-1
00D5 2806       1735        JR   Z,KSC5
                1740 ; CHECK FOR UNSHIFTED
00D7 E67F       1745        AND  #7F
00D9 D7         1750        RST  RCAL
00DA 55         1755        DEFB KSE-$-1
00DB 20CB       1760        JR   NZ,KSC8
                1765 ; CALC ASCII VALUE
00DD 37         1770 KSC5   SCF
00DE ED52       1775        SBC  HL,DE
00E0 7D         1780        LD   A,L
                1785 ;
                1790 ; SUPPORT LOWER CASE
00E1 FE41       1795        CP   "A
00E3 381E       1800        JR   C,K20
00E5 FE5B       1805        CP   "Z+1
00E7 301A       1810        JR   NC,K20
                1815 ; IF IS A LETTER
00E9 21010C     1820        LD   HL,KMAP
00EC CB66       1825        BIT  4,(HL)
                1830 ; 1 = SHIFT DOWN
00EE 21270C     1835        LD   HL,$KOPT
00F1 2008       1840        JR   NZ,K8
                1845 ; SHIFT NOT DOWN
00F3 CB46       1850        BIT  0,(HL)
00F5 280C       1855        JR   Z,K20
00F7 C620       1860        ADD  A,#20
00F9 1808       1865        JR   K20
                1870 ; SHIFT WAS DOWN
00FB C620       1875 K8     ADD  A,#20
00FD CB46       1880        BIT  0,(HL)
00FF 2802       1885        JR   Z,K20
0101 D620       1890        SUB  #20
                1895 ;
                1900 ; CONTROL KEYS
0103 21010C     1905 K20    LD   HL,KMAP
                1910 ; IF NOT @, MAY MODIFY
0106 FE40       1915        CP   "@
0108 2006       1920        JR   NZ,K30
                1925 ; IF SHIFT DOWN, NORMAL,
                1930 ;  OTHERWISE IGNORE
010A CB66       1935        BIT  4,(HL)
010C 2897       1940        JR   Z,KSC8
010E 1806       1945        JR   K35
                1950 ; IF @ DOWN, MODIFY
0110 CB6E       1955 K30    BIT  5,(HL)
0112 2802       1960        JR   Z,K35
0114 EE40       1965        XOR  #40
                1970 ; CONTROL
0116 CB5E       1975 K35    BIT  3,(HL)
0118 2802       1980        JR   Z,K40
011A EE40       1985        XOR  #40
                1990 ; GRAPHIC
011C 21060C     1995 K40    LD   HL,KMAP+5
011F CB76       2000        BIT  6,(HL)
0121 2802       2005        JR   Z,K55
```

```
0123 EE80      2010        XOR  N80
               2015 ;
               2020 ; K4 OPTION
               2025 ;  CHANGE BIT 7
0125 21270C    2030 K55    LD   HL,$KOPT
0128 CB56      2035        BIT  2,(HL)
012A 2002      2040        JR   Z,K60
012C EE80      2045        XOR  N80
               2050 ;
               2055 ; END
012E 37        2060 K60    SCF
012F C9        2065        RET
               2070 ;
               2075 ; SEARCH KEYBOARD TABLE
0130 2A6F0C    2080 KSE    LD   HL,($KTAB)
0133 54        2085        LD   D,H
0134 5D        2090        LD   E,L
0135 ED4B6D0C  2095        LD   BC,($KTABL)
0139 EDB1      2100        CPIR
013B C9        2105        RET
               2110 ;
               2115 ;
               2120 ; WORKSPACE INITIALISATION
               2125 ;  TABLE
013C           2130 INITT  EQU  $
               2135 ; USER STACK
013C 0010      2140 ISP    DEFW RAMTOP
               2145 ; LENGTH OF KTAB
013E 6000      2150 IKTABL DEFW KTABE-KTAB
               2155 ; KEYBOARD TABLE
0140 9E05      2160 IKTAB  DEFW KTAB
               2165 ; SUBROUTINE TABLE
0142 0607      2170 ISTAB  DEFW STABA-"A-"A
               2175 ; OUTPUT TABLE
0144 7F07      2180 IOUT   DEFW OUTT1
               2185 ; INPUT TABLE
0146 8207      2190 IIN    DEFW INT1
               2195 ; USER JUMPS
0148 C32F00    2200 IUOUT  JP   DRET
014B C32F00    2205 IUIN   JP   DRET
               2210 ; NMI JUMP
014E C3        2215 INMI   DEFB NC3
014F           2220 INITX  EQU  $
               2225 ;
               2230 ;
               2235 ; CRT ROUTINE
               2240 ; IGNORE NULL OR LF
014F B7        2245 CRT    OR   A
0150 C8        2250        RET  Z
0151 F5        2255        PUSH AF
0152 FE0A      2260        CP   LF
```

```
0154 2824      2265        JR   Z,CRT2
               2270 ;
               2275 ; CLEAR SCREEN
0156 FE0C      2280        CP   CS
0158 2022      2285        JR   NZ,CRT6
               2290 ; CLEAR TOP LINE
015A 210A08    2295        LD   HL,VL1
015D E5        2300        PUSH HL
015E 0630      2305        LD   B,48
0160 3620      2310 CR1    LD   (HL)," "
0162 23        2315        INC  HL
0163 10FB      2320        DJNZ CR1
               2325 ; SET MARGIN
0165 0610      2330        LD   B,16
0167 3600      2335 CR3    LD   (HL),0
0169 23        2340        INC  HL
016A 10FB      2345        DJNZ CR3
               2350 ; COPY DOWN SCREEN
016C EB        2355        EX   DE,HL
016D E1        2360        POP  HL
016E E5        2365        PUSH HL
016F 010003    2370        LD   BC,VEND-VRAM-64-10-6
0172 EDB0      2375        LDIR
               2380 ; SET TO TOP LEFT
0174 E1        2385        POP  HL
               2390 ;
               2395 ; SET HL TO LEFT SIDE
0175 DF        2400 CRT0   RST  SCAL
0176 7C        2405        DEFB ZCPOS
               2410 ;
               2415 ; SAVE CURSOR
0177 22290C    2420 CRT1   LD   (CURSOR),HL
               2425 ;
               2430 ; RETURN
017A F1        2435 CRT2   POP  AF
017B C9        2440        RET
               2445 ;
               2450 ; SET HL TO CURSOR
017C 2A290C    2455 CRT6   LD   HL,(CURSOR)
               2460 ;
               2465 ; BS, CUL
017F FE08      2470        CP   BS
0181 2011      2475        JR   NZ,CRT14
0183 F5        2480 CRT8   PUSH AF
               2485 ; IGNORE MARGINS
0184 2B        2490 CRT10  DEC  HL
0185 7E        2495        LD   A,(HL)
0186 B7        2500        OR   A
0187 28FB      2505        JR   Z,CRT10
0189 F1        2510        POP  AF
018A FE11      2515        CP   CUL
```

```
018C 2802      2520         JR   Z,CRT12
018E 3620      2525         LD   (HL),"
0190 D7        2530 CRT12   RST  RCAL
0191 63        2535         DEFB CTST-$-1
0192 18E6      2540         JR   CRT2
0194 FE11      2545 CRT14   CP   CUL
0196 28EB      2550         JR   Z,CRT8
               2555 ;
               2560 ; CURSOR HOME, ESC
0198 FE17      2565         CP   CH
019A 28D9      2570         JR   Z,CRT0
019C FE1B      2575         CP   ESC
019E 200B      2580         JR   NZ,CRT20
01A0 DF        2585         RST  SCAL
01A1 7C        2590         DEFB ZCPOS
01A2 0630      2595         LD   B,48
01A4 3620      2600 CRT18   LD   (HL),"
01A6 23        2605         INC  HL
01A7 10FB      2610         DJNZ CRT18
01A9 18CA      2615         JR   CRT0
               2620 ;
               2625 ; NEW LINE, CCR
01AB FE0D      2630 CRT20   CP   CR
01AD 2866      2635         JR   Z,CRT38
01AF FE18      2640         CP   CCR
01B1 200C      2645         JR   NZ,CRT25
01B3 E5        2650         PUSH HL
01B4 DF        2655         RST  SCAL
01B5 7C        2660         DEFB ZCPOS
01B6 D1        2665         POP  DE
01B7 B7        2670         OR   A
01B8 ED52      2675         SBC  HL,DE
01BA 19        2680         ADD  HL,DE
01BB 28BA      2685         JR   Z,CRT1
01BD 1856      2690         JR   CRT38
               2695 ;
               2700 ; CUU, CUD
01BF FE13      2705 CRT25   CP   CUU
01C1 2008      2710         JR   NZ,CRT28
01C3 11C0FF    2715         LD   DE,-64
01C6 19        2720 CRT26   ADD  HL,DE
01C7 D7        2725         RST  RCAL
01C8 2C        2730         DEFB CTST-$-1
01C9 18AF      2735         JR   CRT2
01CB FE14      2740 CRT28   CP   CUD
01CD 2005      2745         JR   NZ,CRT29
01CF 114000    2750         LD   DE,64
01D2 18F2      2755         JR   CRT26
               2760 ;
               2765 ; CSL, CSR
01D4 FE15      2770 CRT29   CP   CSL
01D6 200E      2775         JR   NZ,CRT32
01D8 23        2780 CRT30   INC  HL
01D9 7E        2785         LD   A,(HL)
01DA 2B        2790         DEC  HL
01DB B7        2795         OR   A
01DC 2004      2800         JR   NZ,CRT31
01DE 3620      2805         LD   (HL),"
01E0 1898      2810         JR   CRT2
01E2 77        2815 CRT31   LD   (HL),A
01E3 23        2820         INC  HL
01E4 18F2      2825         JR   CRT30
01E6 FE16      2830 CRT32   CP   CSR
01E8 201F      2835         JR   NZ,CRT34
01EA 0620      2840         LD   B,"
01EC 7E        2845 CRT33   LD   A,(HL)
01ED B7        2850         OR   A
01EE 280A      2855         JR   Z,CRT2
01F0 70        2860         LD   (HL),B
01F1 47        2865         LD   B,A
01F2 23        2870         INC  HL
01F3 18F7      2875         JR   CRT33
               2880 ;
               2885 ; TEST FOR ON SCREEN
01F5 110A08    2890 CTST    LD   DE,VL1
01F8 B7        2895         OR   A
01F9 ED52      2900         SBC  HL,DE
01FB 19        2905         ADD  HL,DE
01FC D8        2910         RET  C
01FD 110A0B    2915         LD   DE,VL15+48
0200 B7        2920         OR   A
0201 ED52      2925         SBC  HL,DE
0203 19        2930         ADD  HL,DE
0204 D0        2935         RET  NC
0205 F1        2940         POP  AF
0206 C37701    2945 CT8     JP   CRT1
               2950 ;
               2955 ; CUR, OTHERS
0209 FE12      2960 CRT34   CP   CUR
020B 2801      2965         JR   Z,CRT36
020D 77        2970         LD   (HL),A
               2975 ; IGNORE MARGINS
020E 23        2980 CRT36   INC  HL
020F 7E        2985         LD   A,(HL)
0210 B7        2990         OR   A
0211 28FB      2995         JR   Z,CRT36
0213 D7        3000         RST  RCAL
0214 E0        3005         DEFB CTST-$-1
               3010 ;
               3015 ; DO NEW LINE
0215 DF        3020 CRT38   RST  SCAL
0216 7C        3025         DEFB ZCPOS
```

```
0217 114000     3030        LD   DE,64
021A 19         3035        ADD  HL,DE
021B D7         3040        RST  RCAL
021C D8         3045        DEFB CTST-$-1
                3050 ;
                3055 ; SCROLL UP
021D 110A08     3060 CRT40  LD   DE,VL1
0220 214A08     3065        LD   HL,VL2
0223 017003     3070        LD   BC,VEND-VRAM-64-64-16
0226 EDB0       3075        LDIR
                3080 ; CLEAR BOTTOM LINE
0228 0630       3085        LD   B,48
022A 2B         3090 CRT50  DEC  HL
022B 3620       3095        LD   (HL),"
022D 10FB       3100        DJNZ CRT50
022F 218A0B     3105        LD   HL,VL15
0232 18D2       3110        JR   CT8
                3115 ;
                3120 ; SET HL TO START OF LINE
0234 7D         3125 CPOS   LD   A,L
0235 D640       3130 CP2    SUB  #40
0237 30FC       3135        JR   NC,CP2
0239 C636       3140        ADD  A,#36
023B 5F         3145        LD   E,A
023C 7D         3150        LD   A,L
023D 93         3155        SUB  E
023E 6F         3160        LD   L,A
023F C9         3165        RET
                3170 ;
                3175 ;
                3180 ; MODIFY COMMAND
0240 DF         3185 MODIFY RST  SCAL
0241 60         3190        DEFB ZARGS
                3195 ; OUTPUT ADDRESS
0242 220C0C     3200 MOD1   LD   (ARG1),HL
0245 DF         3205        RST  SCAL
0246 66         3210        DEFB ZTBCD3
0247 7E         3215        LD   A,(HL)
0248 DF         3220        RST  SCAL
0249 68         3225        DEFB Z82HEX
                3230 ; GET INPUT LINE
024A EF         3235        RST  PRS
024B 20         3240        DEFB " ,CUL,CUL,CUL,0
024F D7         3245        RST  RCAL
0251 54         3250        DEFB INLS-$-1
                3255 ; GET ADDRESS
0252 DF         3260        RST  SCAL
0253 64         3265        DEFB ZNUM
0254 384C       3270        JR   C,MOD9
0256 7E         3275        LD   A,(HL)
0257 B7         3280        OR   A
0258 2848       3285        JR   Z,MOD9
025A 23         3290        INC  HL
025B D5         3295        PUSH DE
025C 5E         3300        LD   E,(HL)
025D 23         3305        INC  HL
025E 56         3310        LD   D,(HL)
025F EB         3315        EX   DE,HL
0260 D1         3320        POP  DE
0261 0600       3325        LD   B,0
                3330 ; GET EACH ENTRY
0263 E5         3335 MOD2   PUSH HL
0264 DF         3340        RST  SCAL
0265 64         3345        DEFB ZNUM
0266 7E         3350        LD   A,(HL)
0267 B7         3355        OR   A
0268 2807       3360        JR   Z,MOD3
                3365 ; PUT INTO MEMORY
026A 23         3370        INC  HL
                3375 ; HL = NUMN+1 = NUMV
026B 7E         3380        LD   A,(HL)
026C E1         3385        POP  HL
026D 77         3390 MOD2A  LD   (HL),A
026E 04         3395        INC  B
026F 23         3400        INC  HL
0270 E5         3405        PUSH HL
0271 E1         3410 MOD3   POP  HL
0272 1A         3415        LD   A,(DE)
                3420 ; IF "." RETURN
0273 FE2E       3425        CP   "."
0275 C8         3430        RET  Z
                3435 ; IF "," SET CHAR
0276 FE2C       3440        CP   ",
0278 2005       3445        JR   NZ,MOD4
027A 13         3450        INC  DE
027B 1A         3455        LD   A,(DE)
027C 13         3460        INC  DE
027D 18EE       3465        JR   MOD2A
                3470 ; INC IF NONE
027F 78         3475 MOD4   LD   A,B
0280 B7         3480        OR   A
0281 2001       3485        JR   NZ,MOD5
0283 23         3490        INC  HL
                3495 ; IF ":" GO BACK
0284 1A         3500 MOD5   LD   A,(DE)
0285 FE3A       3505        CP   ":
0287 2004       3510        JR   NZ,MOD7
0289 2B         3515        DEC  HL
028A 2B         3520        DEC  HL
028B 18B5       3525        JR   MOD1
                3530 ; IF "/" SET TO VALUE
028D FE2        3535 MOD7   CP   "/
```

```
028F 200A     3540        JR   NZ,MOD8
0291 13       3545        INC  DE
0292 DF       3550        RST  SCAL
0293 64       3555        DEFB ZNUM
0294 380C     3560        JR   C,MOD9
0296 2A210C   3565        LD   HL,(NUMV)
0299 18A7     3570        JR   MOD1
029B B7       3575 MOD8   OR   A
029C 28A4     3580        JR   Z,MOD1
029E FE20     3585        CP   " "
02A0 28C1     3590        JR   Z,MOD2
02A2 DF       3595 MOD9   RST  SCAL
02A3 6B       3600        DEFB ZERRM
02A4 189A     3605        JR   MODIFY
              3610 ;
              3615 ;
              3620 ; ROUTINE TO GET INPUT LINE
              3625 ; STORE BPT BYTE
02A6 E5       3630 INLS   PUSH HL
02A7 CD8A03   3635        CALL BRSTO
              3640 ; SET CONFLG TO 0
02AA AF       3645        XOR  A
02AB 32260C   3650        LD   (CONFLG),A
              3655 ; RESET NMI ADDRESS
02AE 211A04   3660        LD   HL,TRAP
02B1 227E0C   3665        LD   ($NMI+1),HL
02B4 E1       3670        POP  HL
              3675 ; NORMAL START OF ROUTINE
              3680 ; GET INPUT CHAR
02B5 E5       3685 INLIN  PUSH HL
02B6 DF       3690 INL2   RST  SCAL
02B7 7B       3695        DEFB ZBLINK
02B8 F7       3700        RST  ROUT
02B9 FE0D     3705        CP   CR
02BB 20F9     3710        JR   NZ,INL2
              3715 ; SET DE TO START OF INPUT
02BD 2A290C   3720        LD   HL,(CURSOR)
02C0 11C0FF   3725        LD   DE,-64
02C3 19       3730        ADD  HL,DE
02C4 EB       3735        EX   DE,HL
02C5 E1       3740        POP  HL
02C6 C9       3745        RET
              3750 ;
              3755 ;
              3760 ; TABULATE COMMAND
02C7 C5       3765 TABCDE PUSH BC
02C8 1817     3770        JR   TB3
              3775 ; IF HL<DE GO TO TB2
02CA B7       3780 TB1    OR   A
02CB ED52     3785        SBC  HL,DE
02CD 19       3790        ADD  HL,DE
02CE 3806     3795        JR   C,TB2
              3800 ; END, SO OUTPUT ".",CR
02D0 C1       3805 TB0    POP  BC
02D1 EF       3810        RST  PRS
02D2 2E       3815        DEFB ".,CR,0
02D5 C9       3820        RET
              3825 ; CONTROL SCROLLING
02D6 78       3830 TB2    LD   A,B
02D7 B1       3835        OR   C
02D8 2007     3840        JR   NZ,TB3
02DA CF       3845        RST  RIN
02DB FE1B     3850        CP   ESC
02DD 28F1     3855        JR   Z,TB0
02DF C1       3860        POP  BC
02E0 C5       3865        PUSH BC
02E1 0B       3870 TB3    DEC  BC
02E2 C5       3875        PUSH BC
              3880 ; INIT CHECKSUM
02E3 0E00     3885        LD   C,0
              3890 ; OUTPUT ADDRESS
02E5 EF       3895        RST  PRS
02E6 20       3900        DEFB " ," ,0
02E9 DF       3905        RST  SCAL
02EA 66       3910        DEFB ZTBCD3
              3915 ; OUTPUT 8 BYTES
02EB 0608     3920        LD   B,8
02ED 7E       3925 TB4    LD   A,(HL)
02EE DF       3930        RST  SCAL
02EF 67       3935        DEFB ZTBCD2
02F0 23       3940        INC  HL
02F1 DF       3945        RST  SCAL
02F2 69       3950        DEFB ZSPACE
02F3 10F8     3955        DJNZ TB4
              3960 ; OUTPUT CHECKSUM ETC
02F5 79       3965        LD   A,C
02F6 DF       3970        RST  SCAL
02F7 68       3975        DEFB ZB2HEX
02F8 EF       3980        RST  PRS
02F9 08       3985        DEFB BS,BS,CR,0
02FD C1       3990        POP  BC
02FE 18CA     3995        JR   TB1
              4000 ;
              4005 ;
              4010 ; OUTPUT HL THEN SPACE
0300 7C       4015 TBCD3  LD   A,H
0301 DF       4020        RST  SCAL
0302 67       4025        DEFB ZTBCD2
0303 7D       4030        LD   A,L
0304 DF       4035        RST  SCAL
0305 67       4040        DEFB ZTBCD2
              4045
```

```
                4050 ;
                4055 ; OUTPUT SPACE
0306 3E20       4060 SPACE  LD   A,"
0308 F7         4065        RST  ROUT
0309 C9         4070        RET
                4075 ;
                4080 ;
                4085 ; ERROR MESSAGE
030A EF         4090 ERRM   RST  PRS
030B 45         4095        DEFM /Error/
0310 00         4100        DEFB 0
                4105 ;
                4110 ;
                4115 ; OUTPUT CR
0311 3E0D       4120 CRLF   LD   A,CR
0313 F7         4125        RST  ROUT
0314 C9         4130        RET
                4135 ;
                4140 ;
                4145 ; ADD TO CHECKSUM, OUTPUT
0315 F5         4150 TBCD2  PUSH AF
0316 81         4155        ADD  A,C
0317 4F         4160        LD   C,A
0318 F1         4165        POP  AF
                4170 ;
                4175 ;
                4180 ; OUTPUT A
0319 F5         4185 B2HEX  PUSH AF
031A 1F         4190        RRA
031B 1F         4195        RRA
031C 1F         4200        RRA
031D 1F         4205        RRA
031E D7         4210        RST  RCAL
031F 01         4215        DEFB B1HEX-$-1
0320 F1         4220        POP  AF
                4225 ;
                4230 ; OUTPUT LOW HALF A
0321 E60F       4235 B1HEX  AND  #0F
0323 C630       4240        ADD  A,"0
0325 FE3A       4245        CP   "9+1
0327 3802       4250        JR   C,B1H4
0329 C607       4255        ADD  A,"A-"0-10
                4260 ; OUTPUT CHAR
032B F7         4265 B1H4   RST  ROUT
032C C9         4270        RET
                4275 ;
                4280 ;
                4285 ; OUTPUT HL DE, ADD TO SUM
032D D7         4290 TX1    RST  RCAL
032E 00         4295        DEFB TX2-$-1
032F DF         4300 TX2    RST  SCAL
0330 66         4305        DEFB ZTBCD3
0331 EB         4310        EX   DE,HL
0332 C9         4315        RET
                4320 ;
                4325 ;
                4330 ; READ IN HEX VALUE
                4335 ; DE = INPUT LINE
                4340 ; NUMN = NO OF CHARS
                4345 ; NUMV = VALUE
0333 1A         4350 NUM    LD   A,(DE)
                4355 ; IGNORE BLANKS
0334 FE20       4360        CP   "
0336 13         4365        INC  DE
0337 28FA       4370        JR   Z,NUM
0339 1B         4375        DEC  DE
                4380 ; NUMV, NUMN = 0
033A 210000     4385        LD   HL,0
033D 22210C     4390        LD   (NUMV),HL
0340 AF         4395        XOR  A
0341 21200C     4400        LD   HL,NUMN
0344 77         4405        LD   (HL),A
                4410 ; GET CHAR
0345 1A         4415 NN1    LD   A,(DE)
                4420 ; CHECK FOR END
0346 B7         4425        OR   A
0347 C8         4430        RET  Z
0348 FE20       4435        CP   "
034A C8         4440        RET  Z
                4445 ; CONVERT FROM ASCII
                4450 ; IF LT 0 INVALID
034B D630       4455        SUB  "0
034D D8         4460        RET  C
                4465 ; IF LT 10 THEN OK, SO NN2
034E FE0A       4470        CP   10
0350 3808       4475        JR   C,NN2
                4480 ; CONVERT A/F FROM ASCII
0352 D607       4485        SUB  "A-"0-10
                4490 ; IF LT 10 INVALID
0354 FE0A       4495        CP   10
0356 D8         4500        RET  C
                4505 ; IF GE 16 INVALID
0357 FE10       4510        CP   16
0359 3802       4515        JR   C,NN2
                4520 ; INVALID
035B 37         4525        SCF
035C C9         4530        RET
                4535 ;
                4540 ; VALID CHAR FOUND
                4545 ; POINT TO NEXT CHAR
035D 13         4550 NN2    INC  DE
                4555 ; INC NUMN
```

```
035E 34        4560        INC  (HL)
               4565 ; PUT VALUE IN NUMV, ROTATING
               4570 ;  PREVIOUS CONTENTS
035F 23        4575        INC  HL
0360 ED6F      4580        RLD
0362 23        4585        INC  HL
0363 ED6F      4590        RLD
0365 2B        4595        DEC  HL
0366 2B        4600        DEC  HL
0367 28DC      4605        JR   Z,NNI
0369 1B        4610        DEC  DE
036A 37        4615        SCF
036B C9        4620        RET
               4625 ;
               4630 ;
               4635 ; GET ARGUMENTS
036C 010B0C    4640 RLIN   LD   BC,ARGN
036F AF        4645        XOR  A
0370 02        4650        LD   (BC),A
               4655 ; GET VALUE
               4660 ; C SET IF INVALID
0371 DF        4665 RL2    RST  SCAL
0372 64        4670        DEFB ZNUM
0373 D8        4675        RET  C
               4680 ; CHECK FOR END
0374 7E        4685        LD   A,(HL)
0375 B7        4690        OR   A
0376 C8        4695        RET  Z
               4700 ; COPY TO ARG1/10
0377 23        4705        INC  HL
0378 03        4710        INC  BC
0379 7E        4715        LD   A,(HL)
037A 02        4720        LD   (BC),A
037B 23        4725        INC  HL
037C 03        4730        INC  BC
037D 7E        4735        LD   A,(HL)
037E 02        4740        LD   (BC),A
               4745 ; INC ARGN
037F 210B0C    4750        LD   HL,ARGN
0382 34        4755        INC  (HL)
0383 7E        4760        LD   A,(HL)
0384 FE0B      4765        CP   11
0386 38E9      4770        JR   C,RL2
0388 37        4775        SCF
0389 C9        4780        RET
               4785 ;
               4790 ;
               4795 ; STORE BPT BYTE
038A 2A230C    4800 BRSTO  LD   HL,(BRKADR)
038D 7E        4805        LD   A,(HL)
038E 32250C    4810        LD   (BRKVAL),A
0391 C9        4815        RET
               4820 ;
               4825 ;
               4830 ; RESTORE BPT BYTE
0392 2A230C    4835 BRRES  LD   HL,(BRKADR)
0395 3A250C    4840        LD   A,(BRKVAL)
0398 77        4845        LD   (HL),A
0399 C9        4850        RET
               4855 ;
               4860 ;
               4865 ; MONITOR INITIALISATION
               4870 ; RESTORE BPT BYTE
039A D7        4875 STRTB  RST  RCAL
039B F6        4880        DEFB BRRES-$-1
               4885 ; SET WORKSPACE TO 0
039C 11000C    4890        LD   DE,INITZ
039F 0660      4895        LD   B,INITR-INITZ
03A1 AF        4900        XOR  A
03A2 12        4905 ST4    LD   (DE),A
03A3 13        4910        INC  DE
03A4 10FC      4915        DJNZ ST4
               4920 ; SET WORKSPACE FROM TABLE
03A6 213C01    4925        LD   HL,INITT
03A9 011500    4930        LD   BC,INITE-INITR
03AC EDB0      4935        LDIR
               4940 ; CLEAR SCREEN
03AE EF        4945        RST  PRS
03AF 0C        4950        DEFB CS,0
03B1 C9        4955        RET
               4960 ;
               4965 ;
               4970 ; USER RETURN
               4975 ; RESET STACKS
03B2 31610C    4980 MRET   LD   SP,STACK
03B5 2A3C01    4985        LD   HL,(ISP)
03B8 22B00C    4990        LD   (RSP),HL
03BB EF        4995        RST  PRS
03BC 4E        5000        DEFM /NAS-SYS 1/
03C5 0D        5005        DEFB CR,0
               5010 ;
               5015 ;
               5020 ; BPT OR NMI
               5025 ; RESTORE BPT BYTE
03C7 D7        5030 STRTD  RST  RCAL
03C8 C9        5035        DEFB BRRES-$-1
               5040 ;
               5045 ;
               5050 ; MAIN MONITOR LOOP
               5055 ; GET LINE AND OBEY
03C9 CDA602    5060 PARSE  CALL INLS
03CC 01280C    5065        LD   BC.ARGX
```

```
                5070 ; IF COMMAND IS BLANK, AND
                5075 ;  PREVIOUS COMMAND NOT S,
                5080 ;  IGNORE IT
03CF 1A         5085        LD   A,(DE)
03D0 FE20       5090        CP   " 
03D2 2005       5095        JR   NZ,PA2
03D4 0A         5100        LD   A,(BC)
03D5 FE53       5105        CP   "S
03D7 20F0       5110        JR   NZ,PARSE
                5115 ; CHECK AND STORE
03D9 FE41       5120 PA2    CP   "A
03DB 380D       5125        JR   C,PERR
03DD FE5B       5130        CP   "Z+1
03DF 3009       5135        JR   NC,PERR
03E1 02         5140        LD   (BC),A
03E2 320A0C     5145        LD   (ARGC),A
                5150 ; POINT TO NEXT CHAR
03E5 13         5155        INC  DE
                5160 ; GET ARGS
03E6 DF         5165        RST  SCAL
03E7 79         5170        DEFB ZRLIN
03E8 3004       5175        JR   NC,PEND
03EA DF         5180 PERR   RST  SCAL
03EB 6B         5185        DEFB ZERRM
03EC 18DB       5190        JR   PARSE
                5195 ; CALL COMMAND ROUTINE
03EE DF         5200 PEND   RST  SCAL
03EF 60         5205        DEFB ZARGS
03F0 DF         5210        RST  SCAL
03F1 5C         5215        DEFB ZSCALJ
03F2 18D5       5220        JR   PARSE
                5225 ;
                5230 ;
                5235 ; THE EXECUTE COMMAND
                5240 ; CONFLG NOT 0 IF E COMMAND
03F4 3EFF       5245 EXEC   LD   A,-1
03F6 32260C     5250        LD   (CONFLG),A
                5255 ;
                5260 ; EXECUTE AND STEP COMMANDS
                5265 ; DISCARD RETURN
03F9 F1         5270 STEP   POP  AF
                5275 ; IF NO ADDRESS ENTERED,
                5280 ;  USE STORED USER PC
03FA 3A0B0C     5285        LD   A,(ARGN)
03FD B7         5290        OR   A
03FE 2803       5295        JR   Z,EXEC2
                5300 ; USER PC = NEW ADDRESS
0400 22690C     5305        LD   (RPC),HL
                5310 ; RESTORE USER REGS BC DE AF
0403 C1         5315 EXEC2  POP  BC
0404 D1         5320        POP  DE
0405 F1         5325        POP  AF; IN FACT RHL
0406 F1         5330        POP  AF
                5335 ; RESTORE USER SP
0407 2A6B0C     5340        LD   HL,(RSP)
040A F9         5345        LD   SP,HL
                5350 ; PUT USER PC ON TOP OF STACK
040B 2A690C     5355        LD   HL,(RPC)
040E E5         5360        PUSH HL
                5365 ; RESTORE USER HL
040F 2A650C     5370        LD   HL,(RHL)
                5375 ; SET BIT 3 OF P0, TO
                5380 ;  ACTIVATE NMI
0412 F5         5385        PUSH AF
0413 3E08       5390        LD   A,8
0415 D300       5395        OUT  (0),A
0417 F1         5400        POP  AF
                5405 ; EXECUTE ONE STEP OF PROGRAM
0418 ED45       5410        RETN
                5415 ;
                5420 ;
                5425 ; COME HERE AFTER NMI OR BPT
041A F5         5430 TRAP   PUSH AF
041B E5         5435        PUSH HL
                5440 ; RESET NMI BIT IN P0
041C 3A000C     5445        LD   A,(PORT0)
041F D300       5450        OUT  (0),A
                5455 ; IF CONFLG NOT 0 THEN E
                5460 ;  SO EXECUTE NORMALLY
0421 3A260C     5465        LD   A,(CONFLG)
0424 B7         5470        OR   A
0425 280D       5475        JR   Z,ER1
                5480 ; STORE BPT BYTE
                5485 ;  AND INSERT RESTART
0427 CD8A03     5490        CALL BRSTO
042A 36E7       5495        LD   (HL),#E7
                5500 ; SET CONFLG TO 0, FOR
                5505 ;  NMI OR BPT
042C AF         5510        XOR  A
042D 32260C     5515        LD   (CONFLG),A
                5520 ; EXECUTE PROGRAM NORMALLY
0430 E1         5525        POP  HL
0431 F1         5530        POP  AF
0432 ED45       5535        RETN
                5540 ;
                5545 ; STORE USER REGISTERS
0434 D5         5550 ER1    PUSH DE
0435 C5         5555        PUSH BC
                5560 ; STACK HAS: PC AF HL DE BC
                5565 ; SET HL TO USER SP
0436 210000     5570        LD   HL,0
0439 39         5575        ADD  HL,SP
```

($NMI) → 041A

```
                5580 ; SET MONITOR SP
043A 31610C     5585        LD   SP,STACK
                5590 ; COPY USER REGS FROM USER
                5595 ;  STACK TO REG SAVE AREA
043D 11610C     5600        LD   DE,STACK
0440 010800     5605        LD   BC,8
0443 EDB0       5610        LDIR
                5615 ; SET DE TO PC ON USER STACK
0445 5E         5620        LD   E,(HL)
0446 23         5625        INC  HL
0447 56         5630        LD   D,(HL)
0448 23         5635        INC  HL
                5640 ; STORE USER PC
0449 ED53690C   5645        LD   (RPC),DE
                5650 ; STORE USER SP
044D 22680C     5655        LD   (RSP),HL
                5660 ;
                5665 ; OUTPUT REGISTERS
0450 EF         5670        RST  PRS
0451 18         5675        DEFB CCR,8
                5680 ;  SP PC AF HL DE BC
0453 21690C     5685        LD   HL,RSAE
0456 0606       5690        LD   B,6
0458 2B         5695 ER2    DEC  HL
0459 7E         5700        LD   A,(HL)
045A DF         5705        RST  SCAL
045B 68         5710        DEFB ZB2HEX
045C 2B         5715        DEC  HL
045D 7E         5720        LD   A,(HL)
045E DF         5725        RST  SCAL
045F 68         5730        DEFB ZB2HEX
0460 DF         5735        RST  SCAL
0461 69         5740        DEFB ZSPACE
0462 10F4       5745        DJNZ ER2
                5750 ; I REG
0464 ED57       5755        LD   A,I
0466 DF         5760        RST  SCAL
0467 68         5765        DEFB ZB2HEX
0468 DF         5770        RST  SCAL
0469 69         5775        DEFB ZSPACE
                5780 ; IX IY REGS
046A DDE5       5785        PUSH IX
046C E1         5790        POP  HL
046D DF         5795        RST  SCAL
046E 66         5800        DEFB ZTBCD3
046F FDE5       5805        PUSH IY
0471 E1         5810        POP  HL
0472 DF         5815        RST  SCAL
0473 66         5820        DEFB ZTBCD3
                5825 ; F REG
0474 3A670C     5830        LD   A,(RAF)
0477 110004     5835        LD   DE,ESTR-1
047A 0608       5840        LD   B,8
047C 13         5845 ER4    INC  DE
047D 17         5850        RLA
047E F5         5855        PUSH AF
047F 1A         5860        LD   A,(DE)
0480 3001       5865        JR   NC,ER6
0482 F7         5870        RST  ROUT
0483 F1         5875 ER6    POP  AF
0484 10F6       5880        DJNZ ER4
0486 EF         5885        RST  PRS
0487 18         5890        DEFB CCR,8
0489 C3C703     5895        JP   STRTD
                5900 ;
                5905 ; STRING FOR FLAGS
048C 53         5910 ESTR   DEFB "S,"Z,8,"H
0490 00         5915        DEFB 8,"P,"N,"C
                5920 ;
                5925 ;
                5930 ; LOAD COMMAND
0494 DF         5935 LOAD   RST  SCAL
0495 5F         5940        DEFB ZNFLP
                5945 ; NORMAL TABLES
0496 DF         5950        RST  SCAL
0497 77         5955        DEFB ZNNOM
0498 E5         5960        PUSH HL
0499 DF         5965        RST  SCAL
049A 78         5970        DEFB ZNNIM
049B E5         5975        PUSH HL
                5980 ; GET INPUT
049C CF         5985 LOD1   RST  RIN
                5990 ; STRIP PARITY
049D E67F       5995 LOD1A  AND  #7F
                6000 ; IF "." THEN END
049F FE2E       6005        CP   "
04A1 283A       6010        JR   Z,LOD9
                6015 ; IF CR THEN END OF LINE,
                6020 ;  SO LOD2 TO PROCESS IT
04A3 FE0D       6025        CP   CR
04A5 2807       6030        JR   Z,LOD2
                6035 ; IGNORE DS LF ETC
04A7 FE20       6040        CP   "
04A9 38F1       6045        JR   C,LOD1
04AB F7         6050        RST  ROUT
04AC 18EE       6055        JR   LOD1
                6060 ;
                6065 ; CONVERT AND CHECK
04AE 2A299C     6070 LOD2   LD   HL,(CURSOR)
04B1 DF         6075        RST  SCAL
04B2 7C         6080        DEFB ZCPOS
04B3 EB         6085        EX   DE,HL
```

```
04B4 DF       6090        RST  SCAL
04B5 79       6095        DEFB ZRLIN
04B6 3821     6100        JR   C,LOD8
              6105 ; CHECKSUM
04B8 210C0C   6110        LD   HL,ARG1
04BB AF       6115        XOR  A
04BC 0612     6120        LD   B,18
04BE 86       6125 LOD3   ADD  A,(HL)
04BF 23       6130        INC  HL
04C0 10FC     6135        DJNZ LOD3
04C2 BE       6140        CP   (HL)
04C3 2014     6145        JR   NZ,LOD8
              6150 ; COPY TO MEMORY
04C5 2A0C0C   6155        LD   HL,(ARG1)
04C8 110E0C   6160        LD   DE,ARG2
04CB 0608     6165        LD   B,8
04CD 1A       6170 LOD5   LD   A,(DE)
04CE 77       6175        LD   (HL),A
04CF 23       6180        INC  HL
04D0 13       6185        INC  DE
04D1 13       6190        INC  DE
04D2 10F9     6195        DJNZ LOD5
              6200 ; CURSOR HOME
04D4 EF       6205        RST  PRS
04D5 1B       6210        DEFB ESC,0
04D7 18C3     6215        JR   LOD1
              6220 ; BAD DATA, SCROLL UP
04D9 DF       6225 LOD8   RST  SCAL
04DA 6A       6230        DEFB ZCRLF
04DB 18BF     6235        JR   LOD1
              6240 ; END
04DD CF       6245 LOD9   RST  RIN
04DE E67F     6250        AND  7FH
04E0 FE0D     6255        CP   CR
04E2 20B9     6260        JR   NZ,LOD1A
04E4 F7       6265        RST  ROUT
04E5 C38606   6270        JP   R1X
              6275 ;
              6280 ;
              6285 ; WRITE COMMAND
04E8 DF       6290 WRITE  RST  SCAL
04E9 5F       6295        DEFB ZMFLP
              6300 ; WAIT
04EA DF       6305        RST  SCAL
04EB 5D       6310        DEFB ZTDEL
              6315 ; OUTPUT TO CRT ONLY
04EC DF       6320        RST  SCAL
04ED 77       6325        DEFB ZNNON
04EE E5       6330        PUSH HL
              6335 ; OUTPUT 256 NULLS
04EF AF       6340        XOR  A
```

```
04F0 47       6345        LD   B,A
04F1 DF       6350 W3     RST  SCAL
04F2 6F       6355        DEFB ZSRLX
04F3 10FC     6360        DJNZ W3
              6365 ; CALCULATE LENGTH-1
04F5 DF       6370        RST  SCAL
04F6 60       6375        DEFB ZARGS
04F7 ED5B0E0C 6380 W4     LD   DE,(ARG2)
04FB EB       6385        EX   DE,HL
04FC 37       6390        SCF
04FD ED52     6395        SBC  HL,DE
              6400 ; IF LEN-1 IS NEG, END
04FF DA0A06   6405        JP   C,R1Y
0502 EB       6410        EX   DE,HL
              6415 ; HL = START
              6420 ; DE = LENGTH-1
              6425 ; WAIT
0503 AF       6430        XOR  A
0504 FF       6435        RST  RDEL
              6440 ; OUTPUT 00 FF FF FF FF
0505 0605     6445        LD   B,5
0507 DF       6450 W5     RST  SCAL
0508 6F       6455        DEFB ZSRLX
0509 3EFF     6460        LD   A,0FFH
050B 10FA     6465        DJNZ W5
              6470 ; IF BLOCK 0, SET LEN TO E+1
050D AF       6475        XOR  A
050E BA       6480        CP   D
050F 2002     6485        JR   NZ,W6
0511 43       6490        LD   B,E
0512 04       6495        INC  B
              6500 ; SET E TO LENGTH
0513 58       6505 W6     LD   E,B
              6510 ; OUTPUT START ADDRESS
0514 7D       6515        LD   A,L
0515 DF       6520        RST  SCAL
0516 6F       6525        DEFB ZSRLX
0517 7C       6530        LD   A,H
0518 DF       6535        RST  SCAL
0519 6F       6540        DEFB ZSRLX
              6545 ; OUTPUT LENGTH OF DATA
051A 7B       6550        LD   A,E
051B DF       6555        RST  SCAL
051C 6F       6560        DEFB ZSRLX
              6565 ; OUTPUT BLOCK NUMBER
051D 7A       6570        LD   A,D
051E DF       6575        RST  SCAL
051F 6F       6580        DEFB ZSRLX
              6585 ; NOW DISPLAY ALL THIS
              6590 ; AND OUTPUT HEADER CHECKSUM
0520 0E00     6595        LD   C,0
```

```
0522 DF       6600        RST  SCAL
0523 6C       6605        DEFB ZTX1
0524 79       6610        LD   A,C
0525 DF       6615        RST  SCAL
0526 6F       6620        DEFB ZSRLX
              6625 ; OUTPUT THE BLOCK
0527 DF       6630        RST  SCAL
0528 6D       6635        DEFB ZSOUT
              6640 ; OUTPUT CHECKSUM AND NULLS
0529 060B     6645        LD   B,11
052B 79       6650        LD   A,C
052C DF       6655 W9     RST  SCAL
052D 6F       6660        DEFB ZSRLX
052E AF       6665        XOR  A
052F 10FB     6670        DJNZ W9
              6675 ; CRLF (READ HAS SAME TIMING)
0531 DF       6680        RST  SCAL
0532 6A       6685        DEFB ZCRLF
0533 18C2     6690        JR   W4
              6695 ;
              6700 ;
              6705 ; ICOPY COMMAND
              6710 ; IF ARG1 GE ARG2, GO TO
              6715 ;  LDIR COPY
0535 B7       6720 ICOPY  OR   A
0536 ED52     6725        SBC  HL,DE
0538 19       6730        ADD  HL,DE
0539 3009     6735        JR   NC,COPY
              6740 ; SET TO END NOT START
053B 0B       6745        DEC  BC
053C EB       6750        EX   DE,HL
053D 09       6755        ADD  HL,BC
053E EB       6760        EX   DE,HL
053F 09       6765        ADD  HL,BC
0540 03       6770        INC  BC
0541 EDB8     6775        LDDR
0543 C9       6780        RET
              6785 ;
              6790 ;
              6795 ; COPY COMMAND
0544 EDB0     6800 COPY   LDIR
0546 C9       6805        RET
              6810 ;
              6815 ;
              6820 ; ARITHMETIC COMMAND
0547 EB       6825 ARITH  EX   DE,HL
0548 E5       6830        PUSH HL
              6835 ; SUM
0549 19       6840        ADD  HL,DE
054A DF       6845        RST  SCAL
054B 66       6850        DEFB ZTBCD3
              6855 ; DIFFERENCE
054C E1       6860        POP  HL
054D B7       6865        OR   A
054E ED52     6870        SBC  HL,DE
0550 DF       6875        RST  SCAL
0551 66       6880        DEFB ZTBCD3
              6952 ; OFFSET
0552 2B       6890        DEC  HL
0553 2B       6895        DEC  HL
0554 7C       6900        LD   A,H
0555 FEFF     6905        CP   0FF
0557 200A     6910        JR   NZ,A2
0559 CB7D     6915        BIT  7,L
055B 2000     6920        JR   NZ,AOK
              6925 ; NO GOOD SO ??
055D EF       6930 ANG    RST  PRS
055E 3F       6935        DEFB "?,"?,CR,0
0562 C9       6940        RET
0563 B7       6945 A2     OR   A
0564 20F7     6950        JR   NZ,ANG
0566 CB7D     6955        BIT  7,L
0568 20F3     6960        JR   NZ,ANG
              6965 ; OUTPUT OFFSET
056A 7D       6970 AOK    LD   A,L
056B DF       6975        RST  SCAL
056C 68       6980        DEFB ZB2HEX
056D C31103   6985        JP   CRLF
              6990 ;
              6995 ;
              7000 ; RELATIVE CALL RESTART
0570 2B       7005 RCALB  DEC  HL
0571 3B       7010        DEC  SP
0572 3B       7015        DEC  SP
0573 F5       7020        PUSH AF
0574 D5       7025        PUSH DE
0575 5E       7030        LD   E,(HL)
              7035 ; E = OFFSET, SET D
0576 7B       7040        LD   A,E
0577 17       7045        RLA
0578 9F       7050        SBC  A,A
0579 57       7055        LD   D,A
057A 23       7060        INC  HL
057B 19       7065        ADD  HL,DE
057C D1       7070 RCAL4  POP  DE
057D F1       7075        POP  AF
057E E3       7080        EX   (SP),HL
              7085 ; FAKE JUMP TO ROUTINE
057F C9       7090        RET
              7095 ;
              7100 ;
              7105 ; SUBROUTINE CALL RESTART
```

```
0580 2B       7110 SCALB  DEC  HL
0581 3B       7115        DEC  SP
0582 3B       7120        DEC  SP
0583 F5       7125        PUSH AF
0584 D5       7130        PUSH DE
0585 5E       7135        LD   E,(HL)
0586 1600     7140        LD   D,0
0588 2A710C   7145 SCAL2  LD   HL,($STAB)
058B 19       7150        ADD  HL,DE
058C 19       7155        ADD  HL,DE
058D 5E       7160        LD   E,(HL)
058E 23       7165        INC  HL
058F 56       7170        LD   D,(HL)
0590 EB       7175        EX   DE,HL
0591 18E9     7180        JR   RCAL4
              7185 ;
              7190 ;
              7195 ; SUBROUTINE FOR CALL
0593 E5       7200 SCALJ  PUSH HL
0594 F5       7205        PUSH AF
0595 D5       7210        PUSH DE
0596 3A0A0C   7215        LD   A,(ARGC)
0599 5F       7220        LD   E,A
059A 1600     7225        LD   D,0
059C 18EA     7230        JR   SCAL2
              7235 ;
              7240 ;
              7245 ; KEYBOARD TABLE
059E FF       7250 KTAB   DEFB HFF,HFF,HFF,HFF; H00
05A2 FF       7255        DEFB HFF,HFF,HFF,HFF; H04
05A6 08       7260        DEFB H08,HFF,H0E,HFF; H08 BS,LF
05AA 08       7265        DEFB H08,H09,HFF,HFF; H0C CS,CR
05AE FF       7270        DEFB HFF,H3E,H2E,H46; H10  LRU
05B2 36       7275        DEFB H36,HBE,HAE,H0E; H14 DLR,CH
05B6 FF       7280        DEFB HFF,HFF,HFF,H89; H18 ESC
05BA FF       7285        DEFB HFF,HFF,HFF,HFF; H1C
05BE 14       7290        DEFB H14,H9C,H9D,HA3; H20   "H
05C2 92       7295        DEFB H92,HC2,HBA,HB2; H24 $%&'
05C6 AA       7300        DEFB HAA,HA2,H98,HA0; H28 ()*+
05CA 29       7305        DEFB H29,H0A,H21,H19; H2C ,-./
05CE 1A       7310        DEFB H1A,H1C,H1B,H23; H30 0123
05D2 12       7315        DEFB H12,H42,H3A,H32; H34 4567
05D6 2A       7320        DEFB H2A,H22,H18,H20; H38 89:;
05DA A9       7325        DEFB HA9,H0A,HA1,H99; H3C <= ?
05DE 0D       7330        DEFB H0D,H2C,H41,H13; H40 @ABC
05E2 3B       7335        DEFB H3B,H33,H43,H10; H44 DEFG
05E6 40       7340        DEFB H40,H2D,H38,H30; H48 HIJK
05EA 28       7345        DEFB H28,H31,H39,H25; H4C LMNO
05EE 1D       7350        DEFB H1D,H24,H15,H34; H50 PQRS
05F2 45       7355        DEFB H45,H35,H11,H2B; H54 TUVW
05F6 44       7360        DEFB H44,H3D,H3C,H1E; H58 XYZ[
```

```
05FA 9E       7365        DEFB H9E,H16,H9A,H96; H5C \]^←
05FE          7370 KTABE  EQU  $
              7375 ;
              7380 ;
              7385 ; KEYBOARD COMMAND
              7390 ; STORE K OPTIONS
05FE 7D       7395 KOP    LD   A,L
05FF 32270C   7400        LD   ($KOPT),A
0602 C9       7405        RET
              7410 ;
              7415 ;
              7420 ; BPT COMMAND
              7425 ; STORE BPT ADDRESS
0603 22230C   7430 BREAK  LD   (BRKADR),HL
0606 C9       7435        RET
              7440 ;
              7445 ;
              7450 ; OUTPUT COMMAND
0607 44       7455 O      LD   B,H
0608 4D       7460        LD   C,L
0609 ED59     7465        OUT  (C),E
060B C9       7470        RET
              7475 ;
              7480 ;
              7485 ; QUERY COMMAND
060C 44       7490 Q      LD   B,H
060D 4D       7495        LD   C,L
060E ED78     7500        IN   A,(C)
0610 DF       7505        RST  SCAL
0611 68       7510        DEFB ZB2HEX
0612 C31103   7515        JP   CRLF
              7520 ;
              7525 ;
              7530 ; GET ARGUMENTS
0615 ED4B100C 7535 ARGS   LD   BC,(ARG3)
0619 ED5B0E0C 7540        LD   DE,(ARG2)
061D 2A0C0C   7545        LD   HL,(ARG1)
0620 C9       7550        RET
              7555 ;
              7560 ;
              7565 ; GENERATE COMMAND
              7570 ; OUTPUT COMMANDS TO BOTH
0621 217A07   7575 G      LD   HL,OUTT2
0624 DF       7580        RST  SCAL
0625 71       7585        DEFB ZNOM
0626 E5       7590        PUSH HL
0627 214C06   7595        LD   HL,GDS
062A 0606     7600        LD   B,GDSE-GDS
062C 7E       7605 G2     LD   A,(HL)
062D F7       7610        RST  ROUT
              7615 ; WAIT
```

```
w62E 0E1+     7620        LD   C,20
0630 AF       7625        XOR  A
0631 FF       7630 G4     RST  RDEL
0632 0D       7635        DEC  C
0633 20FC     7640        JR   NZ,G4
0635 23       7645        INC  HL
0636 10F4     7650        DJNZ G2
              7655 ; OUTPUT THE DATA
0638 DF       7660        RST  SCAL
0639 57       7665        DEFB "W
              7670 ; WAIT
063A AF       7675        XOR  A
063B FF       7680        RST  RDEL
              7685 ; OUTPUT "E"
063C 3E45     7690        LD   A,"E
063E F7       7695        RST  ROUT
              7700 ; OUTPUT EXECUTION ADDRESS
063F 2A100C   7705        LD   HL,(ARG3)
0642 DF       7710        RST  SCAL
0643 66       7715        DEFB ZTBCD3
0644 3E0D     7720        LD   A,CR
              7725 ; FINAL CR, END
0646 F7       7730        RST  ROUT
0647 E1       7735        POP  HL
0648 22730C   7740        LD   ($OUT),HL
064B C9       7745        RET
              7750 ;
              7755 ; COMMANDS OUTPUT BY GENERATE
064C 0D       7760 GDS    DEFB CR,"E,"0,CR,"R,CR
0652          7765 GDSE   EQU  $
              7770 ;
              7775 ;
              7780 ; STRING TO SERIAL OUTPUT
              7785 ; HL = ADDRESS
              7790 ; B = LENGTH
              7795 ; C = CHECKSUM
0652 0E00     7800 SOUT   LD   C,0
0654 7E       7805 S01    LD   A,(HL)
0655 81       7810        ADD  A,C
0656 4F       7815        LD   C,A
0657 7E       7820        LD   A,(HL)
0658 DF       7825        RST  SCAL
0659 6F       7830        DEFB ZSRLX
065A 23       7835        INC  HL
065B 10F7     7840        DJNZ S01
065D C9       7845        RET
              7850 ;
              7855 ;
              7860 ; READ ROUTINE
065E DF       7865 READ   RST  SCAL
065F 5F       7870        DEFB ZMFLP
              7875 ; NORMAL TABLES
0660 DF       7880        RST  SCAL
0661 77       7885        DEFB ZNNOM
0662 E5       7890        PUSH HL
0663 DF       7895        RST  SCAL
0664 78       7900        DEFB ZNNIM
0665 E5       7905        PUSH HL
              7910 ; LOOK FOR 4 #FF CHARS
0666 CF       7915 R1     RST  RIN
0667 FEFF     7920 R1A    CP   #FF
0669 200B     7925        JR   NZ,R1D
066B 0603     7930        LD   B,3
066D CF       7935 R1C    RST  RIN
066E FEFF     7940        CP   #FF
0670 2004     7945        JR   NZ,R1D
0672 10F9     7950        DJNZ R1C
0674 181B     7955        JR   R3
              7960 ; LOOK FOR 4 ESC CHARS
0676 FE1B     7965 R1D    CP   ESC
0678 20EC     7970        JR   NZ,R1
067A 0603     7975        LD   B,3
067C CF       7980 R1F    RST  RIN
067D FE1B     7985        CP   ESC
067F 20E6     7990        JR   NZ,R1A
0681 10F9     7995        DJNZ R1F
              8000 ; END, RESTORE TABLES
0683 EF       8005 R1W    RST  PRS
0684 18       8010        DEFB CCR,0
0686 E1       8015 R1X    POP  HL
0687 22750C   8020        LD   ($IN),HL
068A E1       8025 R1Y    POP  HL
068B 22730C   8030        LD   ($OUT),HL
068E C35100   8035        JP   MFLP
              8040 ; GET HEADER DATA
0691 CF       8045 R3     RST  RIN
0692 6F       8050        LD   L,A
0693 CF       8055        RST  RIN
0694 67       8060        LD   H,A
0695 CF       8065        RST  RIN
0696 5F       8070        LD   E,A
0697 CF       8075        RST  RIN
0698 57       8080        LD   D,A
              8085 ; DISPLAY AND CHECK
0699 0E00     8090        LD   C,0
069B DF       8095        RST  SCAL
069C 6C       8100        DEFB ZTXI
069D CF       8105        RST  RIN
069E B9       8110        CP   C
069F 201E     8115        JR   NZ,R6
              8120 ; SET B TO LENGTH
06A1 43       8125        LD   B,E
```

```
                    8130 ; LOAD THE DATA
06A2 0E00           8135        LD   C,0
06A4 3A2B0C         8140 R4     LD   A,(ARGX)
06A7 FE52           8145        CP   "R
06A9 2803           8150        JR   Z,R4A
06AB CF             8155        RST  RIN
06AC 1802           8160        JR   R4C
06AE CF             8165 R4A    RST  RIN
06AF 77             8170        LD   (HL),A
06B0 E5             8175 R4C    PUSH HL
06B1 2A290C         8180        LD   HL,(CURSOR)
06B4 77             8185        LD   (HL),A
06B5 E1             8190        POP  HL
06B6 81             8195        ADD  A,C
06B7 4F             8200        LD   C,A
06B8 23             8205        INC  HL
06B9 10E9           8210        DJNZ R4
                    8215 ; CHECK AGAINST CHECKSUM
06BB CF             8220        RST  RIN
06BC B9             8225        CP   C
06BD 2806           8230        JR   Z,R7
                    8235 ; ERROR FOUND
06BF EF             8240 R6     RST  PRS
06C0 3F             8245        DEFB "?," ,0
06C3 18A1           8250        JR   R1
                    8255 ; CR, TEST FOR END
06C5 EF             8260 R7     RST  PRS
06C6 2E             8265        DEFB "." ,0
06C9 AF             8270        XOR  A
06CA BA             8275        CP   D
06CB 2099           8280        JR   NZ,R1
06CD 18B4           8285        JR   R1W
                    8290 ;
                    8295 ;
                    8300 ; USER I/O COMMAND
06CF 218107         8305 UP     LD   HL,INTU
06D2 DF             8310        RST  SCAL
06D3 72             8315        DEFB ZNIN
06D4 217E07         8320        LD   HL,OUTTU
06D7 DF             8325        RST  SCAL
06D8 71             8330        DEFB ZNOM
06D9 C9             8335        RET
                    8340 ;
                    8345 ;
                    8350 ; EXTERNAL (X) COMMAND
06DA 7D             8355 XP     LD   A,L
06DB 32280C         8360        LD   ($XOPT),A
06DE 218507         8365        LD   HL,INTX
06E1 DF             8370        RST  SCAL
06E2 72             8375        DEFB ZNIN
06E3 217D07         8380        LD   HL,OUTTX
```

```
06E6 DF             8385        RST  SCAL
06E7 71             8390        DEFB ZNOM
06EB C9             8395        RET
                    8400 ;
                    8405 ;
                    8410 ; X INPUT ROUTINE
                    8415 ; CHECK FOR INPUT
06E9 DF             8420 XKBD   RST  SCAL
06EA 70             8425        DEFB ZSRLIN
06EB D0             8430        RET  NC
                    8435 ; STRIP PARITY
06EC E67F           8440        AND  #7F
06EE F5             8445        PUSH AF
                    8450 ; IF FULL DUPLEX, SEND BACK
06EF 21280C         8455        LD   HL,$XOPT
06F2 CB6E           8460        BIT  5,(HL)
06F4 CC2107         8465        CALL Z,XSOPO
06F7 D7             8470        RST  RCAL
06F8 20             8475        DEFB XSOPL-$-1
06F9 F1             8480        POP  AF
                    8485 ; IF DEL, MAKE NULL
06FA FE7F           8490        CP   DEL
06FC 2001           8495        JR   NZ,XK2
06FE AF             8500        XOR  A
                    8505 ; IF ESCAPE OR NOW NULL,
                    8510 ;  ASSUME PROGRAM WILL NOT
                    8515 ;  OUTPUT THE CHAR
06FF FE1B           8520 XK2    CP   ESC
0701 2805           8525        JR   Z,XK4
0703 B7             8530        OR   A
0704 2802           8535        JR   Z,XK4
0706 CBFE           8540        SET  7,(HL)
0708 37             8545 XK4    SCF
0709 C9             8550        RET
                    8555 ;
                    8560 ;
                    8565 ; X OUTPUT ROUTINE
070A F5             8570 XOUT   PUSH AF
                    8575 ; OUTPUT UNLESS BIT 7 SET
                    8580 ;  TO SUPPRESS SERIAL OUTPUT
070B 21280C         8585        LD   HL,$XOPT
070E CB7E           8590        BIT  7,(HL)
0710 CC1707         8595        CALL Z,XSOP
                    8600 ; TURN OFF SUPPRESSION
0713 CBBE           8605        RES  7,(HL)
0715 F1             8610        POP  AF
0716 C9             8615        RET
                    8620 ;
                    8625 ; OUTPUT CHAR AND LF
0717 D7             8630 XSOP   RST  RCAL
0718 00             8635        DEFB XSOPO-$-1
```

```
                8640 ; IF IT WAS A CR AND BIT 4
                8645 ;  OF $XOPT = 0, OUTPUT LF
0719 FE0D       8650 XSOPL  CP   CR
071B C0         8655        RET  NZ
071C CB66       8660        BIT  4,(HL)
071E C0         8665        RET  NZ
071F 3E0A       8670        LD   A,LF
                8675 ;
                8680 ; OUTPUT ASCII CHAR
                8685 ; SET PARITY FLAG ETC
0721 B7         8690 XSOPO  OR   A
                8695 ; IGNORE NULLS
0722 C8         8700        RET  Z
0723 F5         8705 XSOPC  PUSH AF
                8710 ; MAKE PARITY EVEN
0724 EA2907     8715        JP   PE,XSOP2
0727 EE80       8720        XOR  #80
                8725 ; IF BIT 0 SET, MAKE IT ODD
0729 CB46       8730 XSOP2  BIT  0,(HL)
072B 2802       8735        JR   Z,XSOP4
072D EE80       8740        XOR  #80
                8745 ; OUTPUT IT
072F DF         8750 XSOP4  RST  SCAL
0730 6F         8755        DEFB ZSRLX
                8760 ; RESTORE ORIGINAL VALUE
0731 F1         8765        POP  AF
0732 C9         8770        RET
                8775 ;
                8780 ;
                8785 ; TERMINAL PROGRAM
0733 DF         8790 XN     RST  SCAL
0734 7B         8795        DEFB ZBLINK
0735 F7         8800        RST  ROUT
0736 18FB       8805        JR   XN
                8810 ;
                8815 ;
                8820 ; MAKE $IN AND $OUT NORMAL
0738 DF         8825 NORMAL RST  SCAL
0739 7B         8830        DEFB ZNNIM
                8835 ;
                8840 ; SET NEW OUTPUT TABLE
073A 217F07     8845 NNOM   LD   HL,OUTT1
073D E5         8850 NOM    PUSH HL
073E 2A730C     8855        LD   HL,($OUT)
0741 E3         8860        EX   (SP),HL
0742 22730C     8865        LD   ($OUT),HL
0745 E1         8870        POP  HL
0746 C9         8875        RET
                8880 ;
                8885 ;
                8890 ; SET NEW INPUT TABLE
0747 216207     8895 NNIM   LD   HL,INT1
074A E5         8900 NIM    PUSH HL
074B 2A750C     8905        LD   HL,($IN)
074E E3         8910        EX   (SP),HL
074F 22750C     8915        LD   ($IN),HL
0752 E1         8920        POP  HL
0753 C9         8925        RET
                8930 ;
                8935 ;
                8940 ; ADDRESS TABLE EXECUTION
0754 E5         8945 IN     PUSH HL
0755 21750C     8950        LD   HL,$IN
0758 1803       8955        JR   ATE
075A 21730C     8960 AOUT   LD   HL,$OUT
                8965 ; GET START OF TABLE
075D D5         8970 ATE    PUSH DE
075E C5         8975        PUSH BC
075F 5E         8980        LD   E,(HL)
0760 23         8985        INC  HL
0761 56         8990        LD   D,(HL)
                8995 ; GET ROUTINE NUMBER
0762 F5         9000 AT4    PUSH AF
0763 1A         9005        LD   A,(DE)
0764 13         9010        INC  DE
                9015 ; CHECK FOR END
0765 B7         9020        OR   A
0766 280D       9025        JR   Z,AT6
0768 320A0C     9030        LD   (ARGC),A
076B F1         9035        POP  AF
                9040 ; CALL ROUTINE
076C D5         9045        PUSH DE
076D D7         9050        OR   A
076E CD9305     9055        CALL SCALJ
0771 D1         9060        POP  DE
0772 30EE       9065        JR   NC,AT4
0774 F5         9070        PUSH AF
0775 F1         9075 AT6    POP  AF
0776 C1         9080        POP  BC
0777 D1         9085        POP  DE
0778 E1         9090        POP  HL
0779 C9         9095        RET
                9100 ;
                9105 ;
                9110 ; OUTPUT TABLES
077A 65         9115 OUTT2  DEFB ZCRT
077B 6F         9120 OUTT3  DEFB ZSRLX
077C 00         9125        DEFB 0
077D 6E         9130 OUTTX  DEFB ZXOUT
077E 75         9135 OUTTU  DEFB ZUOUT
077F 65         9140 OUTT1  DEFB ZCRT
0780 00         9145        DEFB 0
```

```
                        9150 ;
                        9155 ;
                        9160 ; INPUT TABLES
0781 76                 9165 INTU   DEFB ZUIN
0782 61                 9170 INT1   DEFB ZKBD
0783 70                 9175 INT3   DEFB ZSRLIN
0784 00                 9180        DEFB 0
0785 74                 9185 INTX   DEFB ZXKBD
0786 61                 9190 INT4   DEFB ZKBD
0787 00                 9195        DEFB 0
                        9200 ;
                        9205 ;
                        9210 ; SUBROUTINE TABLE
                        9215 ; STARTS WITH "A"
0788 4705               9220 STABA  DEFW ARITH
078A 0306               9225        DEFW BREAK
078C 4405               9230        DEFW COPY
078E 0A03               9235        DEFW ERRM - D
0790 F403               9240        DEFW EXEC
0792 0A03               9245        DEFW ERRM - F
0794 2106               9250        DEFW G
0796 3307               9255        DEFW XN
0798 3505               9260        DEFW ICOPY
079A FAFF               9265        DEFW BPRC
079C FE05               9270        DEFW KOP
079E 9404               9275        DEFW LOAD
07A0 4002               9280        DEFW MODIFY
07A2 3807               9285        DEFW NORMAL
07A4 0706               9290        DEFW 0
07A6 0A03               9295        DEFW ERRM - P
07A8 0C06               9300        DEFW 0
07AA 5E06               9305        DEFW READ
07AC F903               9310        DEFW STEP
07AE C702               9315        DEFW TABCDE
07B0 CF06               9320        DEFW UP
07B2 5E06               9325        DEFW READ
07B4 E804               9330        DEFW WRITE
07B6 DA06               9335        DEFW XP
07B8 0A03               9340        DEFW ERRM
07BA FDFF               9345        DEFW BPRW
07BC B203               9350        DEFW MRET;   #5B
07BE 9305               9355        DEFW SCALJ;  #5C
07C0 3E00               9360        DEFW TDEL;   #5D
07C2 4500               9365        DEFW FFLP;   #5E
07C4 5100               9370        DEFW MFLP;   #5F
07C6 1506               9375        DEFW ARGS;   #60
07C8 8800               9380        DEFW KBD;    #61
07CA 5407               9385        DEFW IN;     #62
07CC B502               9390        DEFW INLIN;  #63
07CE 3303               9395        DEFW NUM;    #64
07D0 4F01               9400        DEFW CRT;    #65
07D2 0003               9405        DEFW TBCD3;  #66
07D4 1503               9410        DEFW TBCD2;  #67
07D6 1903               9415        DEFW B2HEX;  #68
07D8 0603               9420        DEFW SPACE;  #69
07DA 1103               9425        DEFW CRLF;   #6A
07DC 0A03               9430        DEFW ERRM;   #6B
07DE 2003               9435        DEFW TX1;    #6C
07E0 5206               9440        DEFW SOUT;   #6D
07E2 0A07               9445        DEFW XOUT;   #6E
07E4 5000               9450        DEFW SRLX;   #6F
07E6 8100               9455        DEFW SRLIN;  #70
07E8 3D07               9460        DEFW NOM;    #71
07EA 4A07               9465        DEFW NIM;    #72
07EC 5D07               9470        DEFW ATE;    #73
07EE E906               9475        DEFW XKBD;   #74
07F0 770C               9480        DEFW $UOUT;  #75
07F2 7A0C               9485        DEFW $UIN;   #76
07F4 3A07               9490        DEFW NNOM;   #77
07F6 4707               9495        DEFW NNIM;   #78
07F8 6C03               9500        DEFW RLIN;   #79
07FA 2103               9505        DEFW B1HEX;  #7A
07FC 7200               9510        DEFW BLINK;  #7B
07FE 3402               9515        DEFW CPOS;   #7C
                        9520 ;
                        9525 ;
                        9530 ; SUBROUTINE CALL TABLE
005B                    9535 ZMRET  EQU  #5B
005C                    9540 ZSCALJ EQU  #5C
005D                    9545 ZTDEL  EQU  #5D
005E                    9550 ZFFLP  EQU  #5E
005F                    9555 ZMFLP  EQU  #5F
0060                    9560 ZARGS  EQU  #60
0061                    9565 ZKBD   EQU  #61
0062                    9570 ZIN    EQU  #62
0063                    9575 ZINLIN EQU  #63
0064                    9580 ZNUM   EQU  #64
0065                    9585 ZCRT   EQU  #65
0066                    9590 ZTBCD3 EQU  #66
0067                    9595 ZTBCD2 EQU  #67
0068                    9600 ZB2HEX EQU  #68
0069                    9605 ZSPACE EQU  #69
006A                    9610 ZCRLF  EQU  #6A
006B                    9615 ZERRM  EQU  #6B
006C                    9620 ZTX1   EQU  #6C
006D                    9625 ZSOUT  EQU  #6D
006E                    9630 ZXOUT  EQU  #6E
006F                    9635 ZSRLX  EQU  #6F
0070                    9640 ZSRLIN EQU  #70
0071                    9645 ZNOM   EQU  #71
0072                    9650 ZNIM   EQU  #72
0073                    9655 ZATE   EQU  #73
```

```
0074          9660 ZXKBD  EQU  #74
0075          9665 ZUOUT  EQU  #75
0076          9670 ZUIN   EQU  #76
0077          9675 ZNNOM  EQU  #77
0078          9680 ZNNIN  EQU  #78
0079          9685 ZRLIN  EQU  #79
007A          9690 ZB1HEX EQU  #7A
007B          9695 ZBLINK EQU  #7B
007C          9700 ZCPOS  EQU  #7C
              9705 ;
              9710 ;
0800          9715 NEND   EQU  $
              9720 ; END OF LISTING
```

Simple Demonstration Progams Using NAS-SYS 1.

These programs set out to demonstrate some of the features of NAS-SYS 1. Initially the machine code instructions are explained in detail (with the actual instruction mnemonics bracketed), later the instruction mnemonics only will be used. These simple demonstrations do not try to cover the subject thoroughly but are intended to give some indication as to the use of machine code assembly.

The first program is given as an example of how to overcome one of the slight disadvantages of NAS-SYS. There are two ways of printing a text string on the monitor screen:

1) The normal way using the CRT routines; 'PRS' called using restart 28H which puts the string which follows it on the screen until 'PRS' sees '00' which terminates it. Or 'ROUT', called using restart 30H, which prints the contents of the A register on the screen (in ASCII) each time it is called.
2) Copying characters directly into the video RAM.

In NAS-SYS, the extensive screen editing commands do not allow direct access to line 16 (the top line of the monitor screen) using the normal CRT routines. As the top line is an ideal location for titles etc, addressing the top line must be achieved in some other fashion.

Program 1

In this program a title is copied directly into the video RAM.

0C80	3E 0C	Load the accumulator with the code to clear the screen. (LD A, 0CH)
0C82	F7	Call the routine at 30H labelled 'ROUT'. (RST 30H)
0C83	21 8F 0C	Load the HL register pair with the start address of the title. (LD HL, 0C8FH)
0C86	11 D6 0B	Load the DE register pair with the start location on the screen. (LD DE, 0BD6H)
0C89	01 11 00	Load the BC register pair with the length of the title. (LD BC, 0011H)
0C8C	ED B0	Copy the title using a copy instruction. (LDIR)
0C8E	76	Stop the Nascom. (HALT)
0C8F 0C93 0C97 0C9B 0C9F	54 48 49 53 20 49 53 20 54 48 45 20 54 49 54 4C 45	Title as an ASCII string.

The next five programs are designed to be built up into one continuous program. Having entered the first program, and learned how it works, the second program is added to it, likewise with the third, etc.

Program 2

Program 2 clears the screen and loads the title in similar way to program 1. There is a difference, as this program does not copy the title directly, instead, each character is copied as before, but a delay (called as a subroutine) is inserted between each character. As the copy routine used is not automatic, checks have to be made to determine when the title is fully loaded.

0C80	3E 0C	LD A, 0CH Load A with a clear screen symbol.
0C82	F7	RST 30H Print it using restart labelled 'ROUT'.
0C83	21 EE 0C	LD HL, 0CEEH Point HL to the start of the title.
0C86	11 DE 0B	LD DE, 0BDEH Point DE to the screen location.
0C89	01 05 00	LD BC, 0005H Load BC with the length of the title.
0C8C	ED A0	LDI Copy one character.
0C8E	CD E4 0C	CALL 0CE4H Call the delay subroutine.
0C91	AF	XOR A Exclusive OR A to clear it, making it 00.
0C92	B1	OR C OR C with A. If C was 00, then the Z flag is set.
0C93	20 F7	JR NZ, -7 If the Z flag was not set, jump back to 0C8CH.
0C95	76	HALT Stop the Nascom.

The next part of program 2 is the delay subroutine, which makes use of the delay in NAS-SYS, labelled 'RDEL', called by restart 38H, followed by the title.

Note that 'RDEL' is 2.5mS when using a 4MHz clock, and 5mS when using a 2MHz clock. When using a 2MHz clock (Nascom 1), the B register should be loaded with 10H (at 0CE7H) to halve the length of the delay loop.

Note that this next part does not follow directly after the above, but must be typed in before program 2 is used.

0CE4	F5	PUSH AF Save the contents of the AF register pair.
0CE5	C5	PUSH BC Save the contents of the BC register pair.

The contents of these registers must be saved, as they contain information to be used later, which would otherwise be destroyed by the subroutine.

0CE6	06 20	LD B, 20H 32 times the delay is required, so as a DJNZ loop is to be used B is loaded with 32.

```
0CE8  FF          RST 38H  Call the delay
                  routine labelled 'RDEL'.

0CE9  10 FD       DJNZ -1  Decrement B by 1.
                  If B not zero, jump to
                  0CE8H.

0CEB  C1          POP BC   Restore the BC
                  register pair.

0CEC  F1          POP AF   Restore the AF
                  register pair.

0CED  C9          RET  Return from the
                  subroutine.

0CEE  41 20 42 4F The title as an ASCII
0CF2  58          string.
```

Program 3
=========

In this, the next part of the program we propose to draw a vertical column of X's from a location near the bottom of the screen up towards the top. To do this the 'ROUT' routine will be used, having first located the cursor at the desired position. A 'DJNZ' loop is set up which sequentially prints an X, moves the cursor up to the next line, then prints a backspace.

Note that having printed a character, the cursor is automatically moved on to the next position. Hence the backspace.

This is not the most economic way to construct this routine, but serves by way of demonstration.

```
0C95  21 50 08    LD HL, 0850H  Point HL
                  to the cursor position
                  required on the screen.

0C98  22 29 0C    LD (0C29H), HL  Load HL
                  into the cursor store, thus
                  altering the cursor on the
                  screen.
```

This simple little routine may be used at any time to locate the cursor at a desired position on the screen.

```
0C9B  06 0D       LD B, 0DH  Load B with 14
                  as 14 X's are required.

0C9D  3E 58       LD A, 58H  Load A with the
                  code for an X.

0C9F  F7          RST 30H  Call the CRT
                  routine labelled 'ROUT' to
                  print the character.

0CA0  3E 13       LD A, 13H  Load A with the
                  code for a cursor 'up move'.

0CA2  F7          RST 30H  Print it.

0CA3  3E 08       LD A, 08H  Load A with the
                  code for a backspace.

0CA5  F7          RST 30H  Print it.

0CA6  CD E4 0C    CALL 0CE4H  Call the delay
                  subroutine.

0CA9  10 F2       DJNZ -12  Decrement B, if
                  not zero, jump to 0C9DH.

0CAB  76          HALT  Stop the Nascom.
```

Program 4
=========

Program 4 uses the string print routine called by restart 28H, this restart is labelled 'PRS'. The string to be printed is a space followed by an X. As the string is enclosed within a DJNZ loop, and the cursor is not manipulated by an 'up move' command as in the last routine, a horizontal row of X's is printed. Note that as the last screen commands in the previous program were an 'up move' and a backspace, it is appropriate to print one more X before entering the loop. Although this program has much the same effect as the previous program, it is much shorter because of the use of the 'PRS' routine.

```
0CAB  3E 58       LD A, 58H  Load A with the
                  code for an X.

0CAD  F7          RST 30H  Print it.

0CAE  06 10       LD B, 10H  Load B with 16
                  as 16 X's are required.

0CB0  EF          RST 28H  Call the CRT
                  routine labelled 'PRS'.

0CB1  20 58 00    ASCII codes for a space and
                  an X. The 00 tells 'PRS'
                  that this is the end of the
                  string.

0CB4  CD E4 0C    CALL 0CE4H  Call the delay
                  subroutine.

0CB7  10 F7       DJNZ -7  Decrement B. If B
                  not zero jump to 0CB0H.

0CB9  76          HALT  Stop the Nascom.
```

Program 5
=========

The next program is similar to program 3. In fact this prints a second column of X's at the end of the horizontal row of X's. No explanation will be given as this is so similar to the other program.

```
0CB9  06 0D       LD B, 06H
0CBB  3E 14       LD A, 14H
0CBD  F7          RST 30H
0CBE  3E 08       LD A, 08H
0CC0  F7          RST 30H
0CC1  3E 58       LD A, 58H
0CC3  F7          RST 30H
0CC4  CD E4 0C    CALL 0CE4H
0CC7  10 F2       DJNZ -12
0CC9  76          HALT
```

Program 6
=========

The last program in this group gives a good demonstration of the use of the 'PRS' routine. In many ways this is similar to program 4. However, here, the line is printed backwards, using the cursor 'left move' command. Remember that in printing a character the cursor moves one space to the right, hence the three 'left moves' to reach the correct position for the next X.

The small routine at the end is a loop calling the delay subroutine. Note that the

delay routine is also a loop. As this is a loop with a lesser loop inside it, it is known as a 'nested loop'. After the delay, the program returns to the start and repeats itself.

```
0CC9  EF              RST 28H  Call the routine
                      labelled 'PRS'..

0CCA  11 11 11 00     ASCII string moving the
                      cursor back three places.

0CCE  06 10           LD B, 10H  Load B with 16
                      as 16 X's are required.

0CD0  EF              RST 28H

0CD1  58 11 11 11     ASCII string of an X then
0CD5  00              three cursor 'left moves'.

0CD6  CD E4 0C        CALL 0CE4H  Call delay.

0CD9  10 F5           DJNZ -9

0CDB  06 10           LD B, 10H  Load B with 16
                      to loop the delay 16 times.

0CDD  CD E4 0C        CALL 0CE4H  Call the delay.

0CE0  10 F8           DJNZ -8

0CE2  18 9C           JR -115  Jump back to start
                      of program, 0C30H.
```

The last four programs give a simple demonstration of the use of NAS-SYS internal subroutines, which are accessed from a table of numbers called by the restart labelled 'SCAL'. To use an internal subroutine the appropriate restart code (in this case 'DF') is followed by the table number. It will be noticed that some of the table numbers are marked 'not normally used', this is because it is usually easier to use the 'Input/Output' restarts (RIN and ROUT).

In the next two programs the Input Output routines are not used (except to print on the monitor screen in one instance), and the functions of 'RIN' and 'ROUT' are replaced by internal subroutine calls from the table.

From now on, the operands of the instruction mnemonics will be replaced by the labels assigned to the operands; thus, RST 30H will be refered to by its label and will be written RST ROUT, likewise defined bytes (DEFB) will be refered to by label, DEFB SRLX means the byte in the table which points to the subroutine labeled SRLX.

Program 7
=========

This little program outputs the characters typed on the keyboard to the monitor screen and the tape recorder. In this way a tape record of what was typed is preserved.

The first thing the program does is to output a string of characters, which when replayed put the Nascom in the 'H' mode. This can be done, as, when the Nascom is waiting for a key press, it is in fact scanning for an input from either the keyboard or the tape recorder. Refer to the descriptions of the subroutines used.

```
0C80  21 8E 0C        LD HL, TABLE  Point
                      HL at the table of
                      characters to be sent
                      out.

0C83  06 06           LD B, TABLE LENGTH

0C85  DF 6D           RST SCAL, DEFB SOUT
                      Call SOUT and send
                      the characters.

0C87  DF 7B     LOOP  RST SCAL, DEFB BLINK
                      Call BLINK routine
                      to get a character.

0C89  DF 65           RST SCAL, DEFB CRT
                      Call CRT to print it.

0C8B  DF 6F           RST SCAL, DEFB SRLX
                      Call SRLX to send it
                      to the tape recorder.

0C8D  18 F8           JR LOOP  Jump back to
                      LOOP.

0C8F  0C 45 30 0D TABLE Table of characters
0C93  48 0D           to be sent.
```

Now this routine is very inefficient, as the tape recorder is running all the time, and as minimum speed on the Nascom is about 30 characters a second, a significant improvement in tape economy could be achieved if the message were first stored in the memory then sent to the tape recorder all at once.

Program 8
=========

This program sets B to account for the characters to be sent before the start of the text (the prefix), then points HL at the location where the text is to start. It then enters a loop, first saving HL (as this is lost when getting a character), then checking if the character is an '@'. If an '@' is found then the program branches to 'END'. If the character is not an '@' then the character is printed on the screen. Next HL is restored, and the character saved in memory at the location 'pointed to' by HL (labelled 'BUFFER'). HL is incremented by one, and B is incremented by one. The program then loops back for another character.

When an '@' is encountered, the program branches to 'END'. At this instant, HL is pointing to the the location of the '@' on the screen (a function of 'BLINK'), and B contains a count of the characters (plus 6 for the prefix).

First HL is 'POPped' to 'throw away' the PUSH at 0CE5H. In this program there is no real nesseccity for this, as HL is not required, but as this would leave the stack two down, it is both untidy, and, in a different program, could lead to serious problems. Therefore, the rule; if the stack has been 'PUSHed', and this is later not required, 'throw away' the stack.

The program then outputs a message to the screen reminding you to turn on the tape recorder then waits for a key press before continuing. Routine 'KBD' was chosen for the wait, as using 'RIN' may cause a false start because 'RIN' scans the tape input as well as

the keyboard, and the tape recorder may well output a few false characters as it starts up.

Having seen a key press, the routine outputs the characters to the tape recorder using 'SOUT'. When the output is complete, the program outputs a newline to the screen, and returns to the monitor using subroutine 'MRET'.

Note that this routine does not contain any checks as to the quantity of characters stored, and as the program uses 'SOUT' which can only count up to 256, the number of characters should not exceed this ammount (minus 6 for the prefix).

```
0C80  06 06              LD B, TABLE LENGTH
0C82  21 BE 0C           LD HL, BUFFER  Point
                         HL at text space.
0C85  E5          LOOP   PUSH HL  Save HL.
0C86  DF 78              RST SCAL, DEFB BLINK
                         Call BLINK to get a
                         character.
0C88  FE 40              CP 40H  Compare with
                         the ASCII code for an
                         '@'.
0C3A  28 07              JR Z END  If Z flag
                         set, jump to END.
0C3C  F7                 RST ROUT  Call ROUT
                         to print it.
0C3D  E1                 POP HL  Restore HL
0C3E  77                 LD (HL), A  Save the
                         character at HL.
0C3F  23                 INC HL  Increment HL
0C90  04                 INC B  Increment B
0C91  18 F2              JR LOOP  Go get
                         another character.
0C93  E1          END    POP HL  Throw away
                         stack.
0C94  EF                 RST PRS  Print the
                         following string.
0C95  0D 54 75 72        The message.
0C99  6E 20 6F 6E
0C9D  20 72 65 63
0CA1  6F 72 64 65
0CA5  72 2E 00
0CA8  C5                 PUSH BC  Save BC.
0CA9  DF 61       LOOP1  RST SCAL, DEFB KBD
                         Scan the keyboard for
                         a key press.
0CAB  30 FC              JR NC LOOP1  If no
                         key down, jump back
                         to LOOP1.
0CAD  C1                 POP BC  Restore BC
0CAE  21 B7 0C           LD HL, TABLE  Point
                         HL at the prefix.
0CB1  DF 6D              RST SCAL, DEFB SOUT
                         Send to tape.
0CB3  EF                 RST PRS  Print the
                         following string.
0CB4  0D 00              'newline'
0CB6  DF 5B              RST SCAL, DEFB MRET
                         Call MRET to return
                         to NAS-SYS.
0CB8  0C 45 30 0D TABLE  The prefix.
0CBC  48 0D
0CBE              BUFFER Text space.
```

Program 9
=========

This next program gives an insight into decimal to binary conversions, and also demonstrates the more normal use of the 'PRS', 'BLINK' and 'ROUT' routines. The program also uses another routine, 'B2HEX' (refer to the description of this routine).

The program first puts out a messa and then scans the keyboard for an input. Having received an input, the character is displayed on the monitor, then converted from ASCII to decimal by the simple expedient of subtracting 30H from it.

In this instance no checks are made to test the validity of the character, which must be a decimal number. In practise this sort of programming is very bad, as invalid inputs should be trapped, and a backspace allowed for correcting the input in the event of a mistake.

Having converted the number from ASCII to decimal, the number is saved in B. The routine is then repeated to get another number which is saved in C. The A register is then cleared, and the B register multiplied by 2 which is done by shifting the binary number left by one bit. The new number formed is added to A. The B register is now multiplied by 4 (two left shifts), and the result again added to A. C is then added to A. The number has now been converted to pure binary.

The contents of the A register are saved whilst a further message is put out, then restored, and A printed using 'B2HEX', followed by a further message. The routine then jumps back to be repeated until terminated by a RESET.

From now on the programs will be printed in a more compact form known as assembley listing.

```
0C80  EF                 RST PRS
0C81  0C 00              Clear the screen.
0C83  EF           LOOP  RST PRS
0C84  57 68 61 74 20 69  Message.
0C8A  73 20 74 68 65 20
0C90  6E 75 6D 62 65 72
0C96  20 3F 20 00
0C9A  DF 7B              RST SCAL, DEFB BLINK
0C9C  F7                 RST ROUT  Print it.
0C9D  D6 30              SUB 30H  Convert.
0C9F  47                 LD B, A  Save in B.
```

```
0CA0  DF 7B              RST SCAL, DEFB BLINK
0CA2  F7                 RST ROUT  Print it.
0CA3  D6 30              SUB 30H  Convert.
0CA5  4F                 LD C, A  Save in C.
0CA6  AF                 XOR A  Clear A.
0CA7  CB 20              SLA B  Shift left.
0CA9  80                 ADD A, B  Add to A.
0CAA  CB 20              SLA B  Shift left.
0CAC  CB 20              SLA B  Shift left.
0CAE  80                 ADD A, B  Add to A.
0CAF  81                 ADD A, C  Add to A.
0CB0  F5                 PUSH AF  Save AF.
0CB1  EF                 RST PRS
0CB2  0D 49 73 20 00     Message.
0CB7  F1                 POP AF  Restore AF.
0CB8  DF 68              RST SCAL, DEFB B2HEX
0CBA  EF                 RST PRS
0CBB  20 69 6E 20 48 45  Message.
0CC1  58 2E 0D 0D 00
0CC6  18 BB              JR LOOP
```

Program 10
==========

This program is given as another easy example of arithmetic routines. In this case the two numbers are input to the B and C registers in much the same way as program 9. Hence little explanation is given.

A is then cleared and by manipulation of the bits in the B and C registers, a crude form of binary multiplication is carried out, the answer being accumulated in A. Further manipulation is carried out on the result to convert it back into a decimal number. The result is then printed using 'B2HEX' and the program loops back to the beginning.

```
0C80  EF                 RST PRS
0C82  0C 00              Clear the screen.
0C83  EF           LOOP  RST PRS
0C84  31 73 74 20 6E 75  Message.
0C8A  6D 62 65 72 20 00
0C90  DF 7B              RST SCAL, DEFB BLINK
0C92  F7                 RST ROUT
0C93  D6 30              SUB #30
0C95  47                 LD B, A
0C96  EF                 RST PRS
0C97  20 74 69 6D 65 73  Message.
0C9D  20 32 6E 64 20 6E
0CA3  75 6D 62 65 72 20
0CA9  00
0CAA  DF 7B              RST SCAL, DEFB BLINK
0CAC  F7                 RST ROUT
0CAD  D6 30              SUB #30
0CAF  4F                 LD C, A
```

The numbers are now saved in B and C. A is then cleared, and bit 0 in C tested. If the reslt is a 0, the proram jumps forward and shifts B one to the left. If the result was not zero, B is put into A before the left shift. The shifting to the left is equivalent to adding 0's to the partial products in ordinary decimal long multiplication.

```
0CB0  AF                 XOR A  To clear it.
0CB1  CB 41              BIT 0, C  Test bit.
0CB3  28 01              JR Z L1  If 0, jump.
0CB5  78                 LD A, B
0CB6  CB 20        L1    SLA B
```

Then the next bit in C is tested, and depending on the result, the partial product is added to A, the process is repeated until all the bits in C (4 bits) are accounted for.

```
0CB8  CB 49              BIT 1, C  Test bit.
0CBA  28 01              JR Z L2  Jump if 0.
0CBC  80                 ADD A, B  Add to A.
0CBD  CB 20        L2    SLA B  Shift left.
0CBF  CB 51              BIT 2, C  Test bit.
0CC1  28 01              JR Z L3  Jump if 0.
0CC3  80                 ADD A, B  Add to A.
0CC4  CB 20        L3    SLA B  Shift left.
0CC6  CB 59              BIT 3, C  Test bit
0CC8  28 01              JR Z L4  Jump if 0.
0CCA  80                 ADD A, B  Add to A.
```

The A register now contains the binary result of the product of the B and C registers. This now has to be converted back into decimal. This is done by successively subtracting 10 (in decimal) from the number and counting the number of 10's until a carry is found. This partial result, which represents the number of '10's' in the number, then has 10 added back to it (as one too many 10's were subtracted in producing a negative result) and is then shifted four places to the left, and the remainder added to it, giving the decimal representation in A. 'B2HEX' is then used to print it.

```
0CCB  06 00        L4    LD B, 0  To clear it.
0CCD  04           L5    INC B  Increase count.
0CCE  DE 0A              SBC #0A  Subtract 10.
0CD0  30 FB              JR C L5  If no C, jump.
0CD2  05                 DEC B  Adjust count.
0CD3  CE 0A              ADC #0A  Add 10.
0CD5  CB 20              SLA B  Shift left.
0CD7  CB 20              SLA B  Shift left.
0CD9  CB 20              SLA B  Shift left.
0CDB  CB 20              SLA B  Shift left.
0CDD  80                 ADD A, B  Add to A.
0CDE  3D                 DEC A  Adjust count.
```

The A register now contains the number in decimal, which is output to the screen by using 'B2HEX'.

```
0CDF  F5                 PUSH AF  Save AF.
0CE0  EF                 RST PRS
0CE1  20 69 73 20 00     Message.
0CE6  F1                 POP AF  Restore AF.
0CE7  DF 68              RST SCAL, DEFB B2HEX
0CE9  EF                 RST PRS
0CEA  0D 0D 00           Message.
0CED  C3 83 0C           JP LOOP  Jump to
                         start.
```

Remember, care must be taken, as some of the routines modify the registers, and if these are still required, the registers should be saved.

It is hoped that the above examples give some assistance in the use of NAS-SYS internal routines to simplify machine code (or assembly) programming. Almost all the routines accessible from the table may be treated as modules, and the descriptions given elsewhere should be adequate for an understanding of the use of each module.

DRH 790921 iss. 2.

DAVE HUNT