A GUIDE TO NAS-SYS

0.0 Introduction

Imagine the Nascom with no NAS-SYS rom. When switched on, it will start executing whatever instruction it finds in location 0000 and, without a program beginning at this address, the hardware will achieve nothing. Nor can a program be typed in because pressing the keyboard keys will mean nothing to the Nascom since the keyboard itself has to be driven by a program.

To make the hardware perform a useful function, an "operating system" is required and NAS-SYS is an example of just such a small operating system or "monitor". Following power up, NAS-SYS is in control of the hardware. After some initial chores, it scans the keyboard and displays keyed characters on the screen. It then attempts to sort out what the user wishes to do. Does he wish to display or modify the contents of selected ram locations, for example ? Using the various commands, which are described in Chapter 2, the user is given control over the hardware. The combination of the Nascom hardware and the NAS-SYS operating system results in a useful computer system.

One of the major activities of the monitor is to drive the peripherals, which are described from a programmers point of view in Chapter 1. Thus, NAS-SYS includes routines for reading the keyboard, writing to the video display, and transferring data via the serial input and output ports. All these routines and many others are required for the operation of the monitor itself. However, they are also available for use in the users own programs. For example, if the user wishes to read the keyboard during his program, he does not have to write out all the code necessary to do this; instead he simply calls the routine within the monitor. Chapter 3 describes the many useful routines within NAS-SYS and Chapter 4 gives examples of how they might be used.

NAS-SYS is one example of a system program; the BASIC interpreter is another, and several others are available including an assembler, dis-assembler, debugger and compilers or interpreters for various high level languages such as PASCAL and FORTH. All these programs themselves make use of the routines within NAS-SYS. For example, if a BASIC program is requesting an input from the keyboard, it is the keyboard routine within NAS-SYS that is actually being used.

The serious Nascom programmer can greatly extend his comprehension and use of his computer by making the effort to understand NAS-SYS; it is hoped that this book will make that task easier and enjoyable.

1.0 INPUT AND OUTPUT DEVICES

NAS-SYS contains routines for driving the hardware associated with the keyboard, the display screen, and the serial input-output port; it also allows an exchange of data with any other port.

1.1 THE KEYBOARD

The keyboard hardware is physically input/output port 0. These ports are driven by the keyboard driver routine, KBD, within NAS-SYS, see Chapter 3 for a description. (Not all the connections to the ports are used - bit 7 of input port 0 is available to the user via TP3 on Nascom2, and bits 2 and 5 of output port 0 are available via PL3/8 and PL3/6 respectively. Bit 4 is used to drive the TAPE led.)

The keyboard keys are arranged logically in a matrix of 8 rows and 7 columns as shown below:

		Column, CCC						
		6	5	4	3	2	1	0	Mapped location
	1	CH	@	SHFT	CTRL	_	NL	BS	KMAP+8,KMAP
	2	]	R	sp	C	4	V	G	KMAP+7
	3	[	P	1	2	0	/	:	KMAP+6
	4	GRAF	O	Q	3	9	.	;	KMAP+5
Row,	5	--	I	A	W	8	,	L	KMAP+4
RRRR	6	↑	U	S	E	7	M	K	KMAP+3
	7	--	Y	Z	D	6	N	J	KMAP+2
	8	↓	T	X	F	5	B	H	KMAP+1

KMAP=0C01H

The keyboard driver, KBD, scans the rows in turn. When a key is pressed, the input is debounced and the state of the keyboard is entered into the keyboard map, workspace locations KMAP to KMAP+9. Thus, if key P is pressed location KMAP+6 is loaded with 20H, all other map locations being 00.
The position of the pressed key is then encoded as a single byte, structured

SRRRRCCC

where S is the state of the SHFT key,
RRRR is the row number from the above matrix,
CCC is the column number from the above matrix.

Thus, unshifted key P which is at row 3, column 5 produces the byte 0,0011,101 or 1DH.

The keyboard table, KTAB, is then searched to find byte 1DH. It is found 50H bytes down the table which is so ordered that the displacement down the table corresponds to the ASCII code for the key; 50H is the ASCII code for character P.

The keyboard table contains only those characters having ASCII codes 00 to 5FH inclusive. It does not include lower case

characters. If a search fails to find the coded byte, the shift bit,S,is masked off and a second search made through the table. The ASCII code so determined is then modified depending on the state of the Shift, Control, and Graphics keys. The Shift key causes 20H to be added to the ASCII code, so that Shift A becomes 61H, the ASCII code for chracter a. The Control key complements bit 6 of the ASCII code, effectively adding or subtracting 40H, so that Control A produces 01H. Similarly, the Graphics key complements bit 7, effectively adding or subtracting 80H, so that Graphic A produces C1H. By use of the Shift, Control, and Graphics keys in any combination together with the character keys, it is possible to generate all the 256 possible codes from the keyboard.

Note that the effect of the Shift and Graphic keys may be reversed using the K command, see Chapter 2.

If a keypress is detected by the driver routine, KBD, the appropriate ASCII code is returned in register A with the Carry flag set. Otherwise the routine returns with the Carry flag cleared.

The address of the start of the keyboard table, KTAB, is held in workspace location $KTAB (0C6FH) and the length of the table is held in $KTABL (0C6DH). These locations are initialised by NAS-SYS so that the table used is that one within NAS-SYS itself. However, these locations may be changed to point to a user defined table in ram so effectively allowing the codes generated by the keyboard to be determined by the user.

1.2 THE VIDEO DISPLAY

The video display is "memory mapped", ie. it appears to the programmer as a block of read/write memory. It occupies memory locations 0800 to 0BFF, the "video ram".
The screen is divided into sixteen lines, each displaying 48 characters. Each of the 768 (=16 x 48) character positions corresponds to a location in video ram. A map of the screen showing the video ram addresses is shown below.

The character actually displayed at any screen position is determined by the contents of the corresponding video ram location and the character generator roms within the Nascom. The standard Nascom character generator rom is such that to display a particular character its ASCII code should be loaded into the corresponding video ram location. For example, to display 'A' at the beginning of the top line, location 0BCA must be loaded with 41H.

Map of the Video Ram Addresses

```
OBCA(3018) OBCB OBCC ...                ..OBF8 OBF9(3065)
080A(2058) 080B 080C ...                ..0838 0839(2105)
084A(2122) 084B 084C ...                ..0878 0879(2169)
088A(2186) 088B 088C ...                ..08B8 08B9(2233)
08CA(2250) 08CB 08CC ...                ..08F8 08F9(2297)
090A(2314)                                     0939(2361)
094A(2378)                                     0979(2425)
098A(2442)                                     09B9(2489)
09CA(2506)                                     09F9(2553)
0A0A(2570)                                     0939(2617)
0A4A(2634)                                     0A79(2681)
0A8A(2698)                                     0AB9(2745)
0ACA(2762)                                     0AF9(2809)
0B0A(2826)                                     0B39(2873)
0B4A(2890)                                     0B79(2937)
0B8A(2954)                                     0BB9(3001)
```

Note: The top line (OBCA to OBF9) is not scrolled.
Those locations not shown are used as 'margins'.

1.3 THE SERIAL INPUT & OUTPUT PORT

Serial input and output is via a universal asynchronous receiver-transmitter or UART. The transmitter section is output port 1, the receiver section is input port 1, and the control of the UART is via input port 2. The receiver and transmitter are independent of each other except for the number of stop bits which has to be the same for both transmitter and receiver. Link Switch LSW1/5 up produces 1 stop bit while LSW1/5 down produces 2 stop bits. A 110 baud device usually requires 2 stop bits while devices operating at other speeds usually require 1 stop bit.

OUTPUT

The serial output port is driven by driver routine SRLX (Chapter 3).

The serial data is transmitted simultaneously to the cassette as a series of audio tones, to a 20mA loop device as a sequence of 20mA and 0mA, and to an RS232 device as a series of +12 V and -12 V. The speed of transmission may be set to 110, 300, or 1200 baud by means of link switches LSW2/1, LSW2/2, and LSW2/3. Additionally the transmission rate may be set to 2400 baud by adding a link as described below.

Output speed	LSW2/1	LSW2/2	LSW2/3
110 baud	x	up	down
300 baud	down	down	x
1200 baud	up	down	x
External	x	up	up

where x signifies 'don't care'.

The External clock frequency should be 16 times the desired baud rate. For 2400 baud, an 'external' clock of 38.4kHz (=16 x 2400) is available at TP20. Thus, connect TP20 to TP4 (external transmit clock) and set both LSW2/2 and LSW2/3 up.

INPUT

The serial input port is driven by driver routine SRLIN (Chapter 3).

Input data may be accepted from the cassette, from a 20mA loop device, or an RS232 device. Link Switch LSW2/7 determines the source of the data:

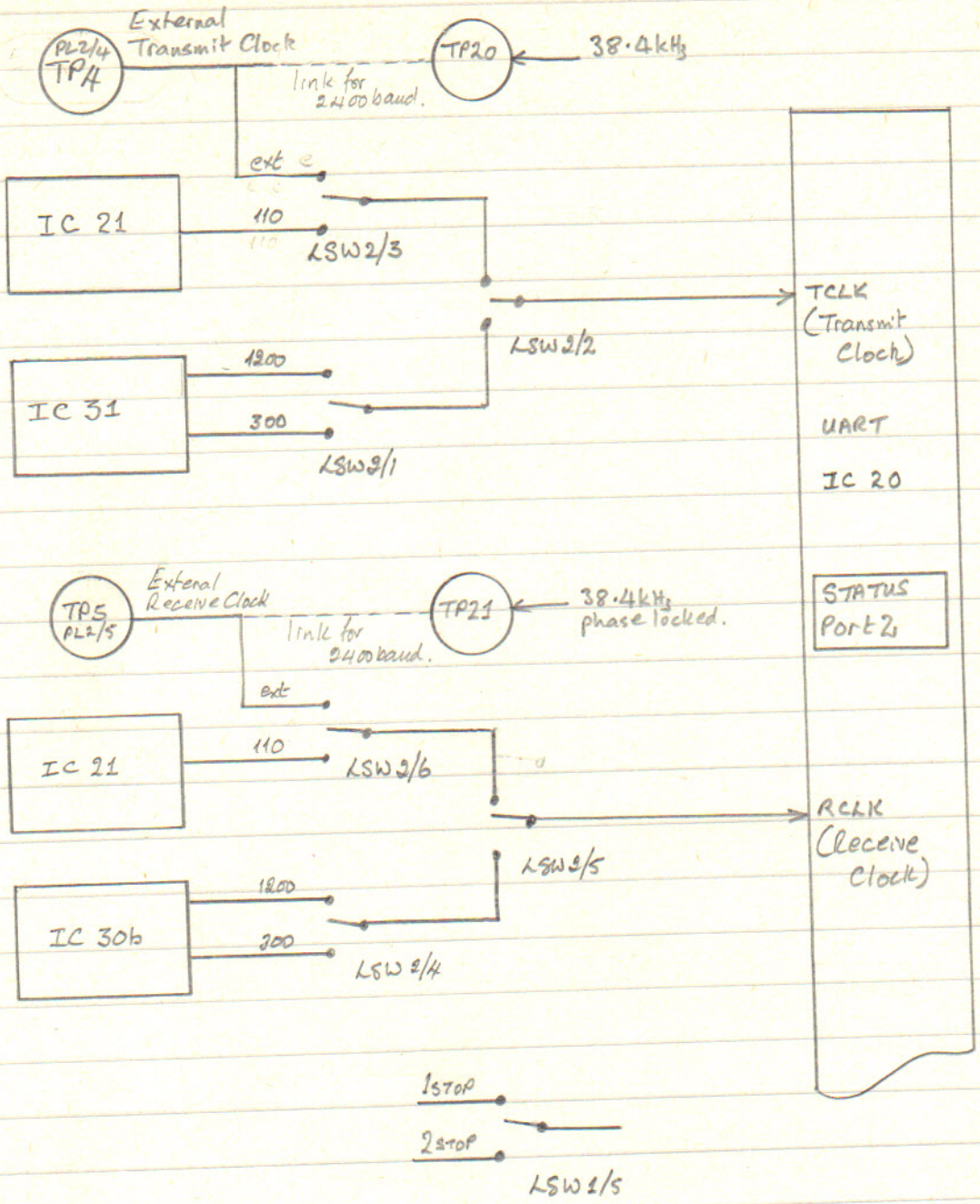
Input from	LSW2/7
Cassette recorder	down
RS232 device or loop device	up

Input speed	LSW2/4	LSW2/5	LSW2/6
110 baud	x	up	down
300 baud	down	down	x
1200 baud	up	down	x
External	x	up	up

where x signifies 'don't care'

The External clock frequency should be 16 times the desired baud rate. For 2400 baud input, an 'external' clock of 38.4kHz (=16 x 2400) which is phase locked to the incoming signal is available at TP21. Thus, connect TP21 to TP5 (external receive clock) and set both LSW2/5 and LSW2/6 up.

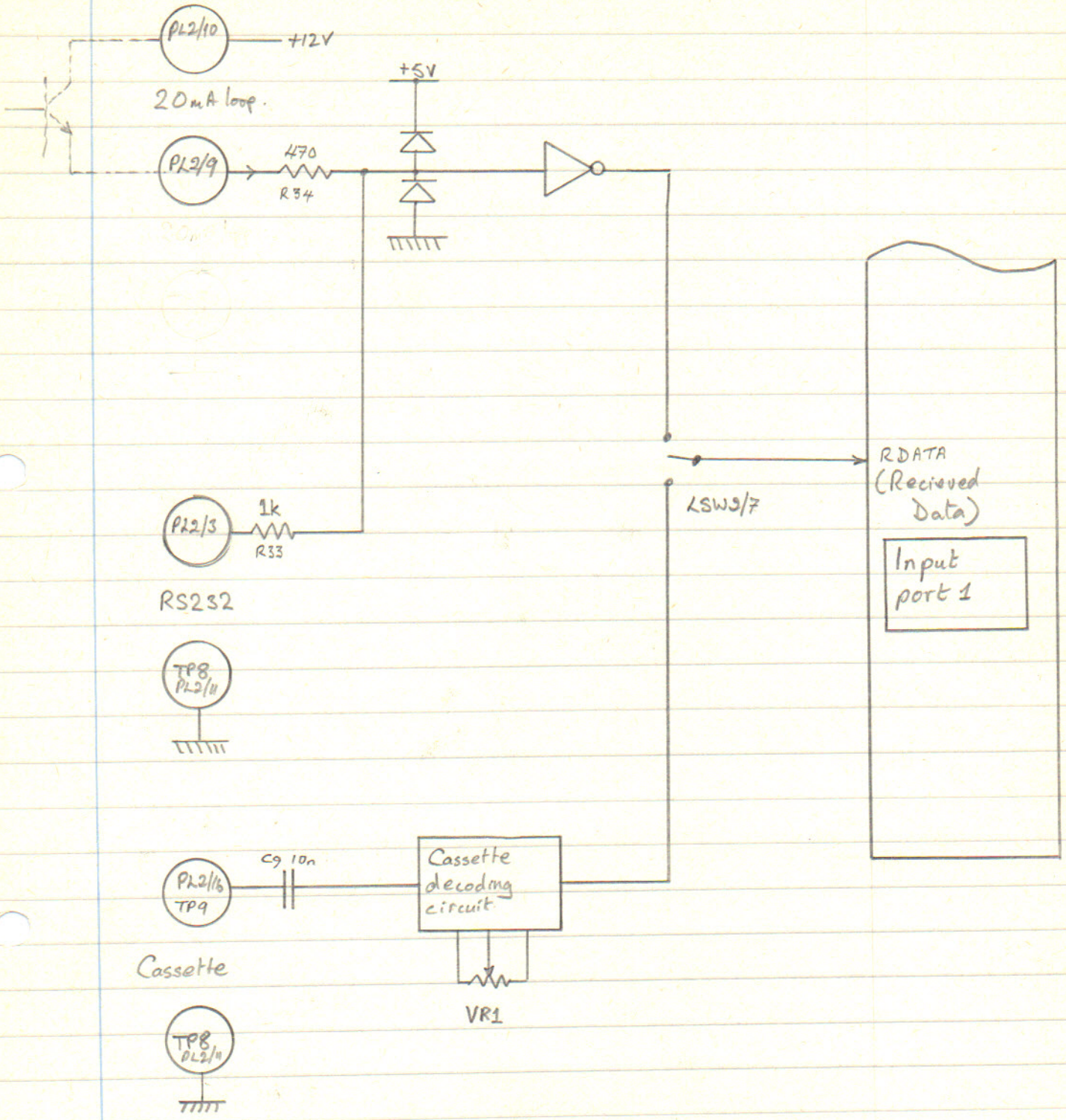
Setting the receive and transmit speeds and the number of stop bits.



The up position of the switches in the diagram corresponds to the up position of the switch itself.

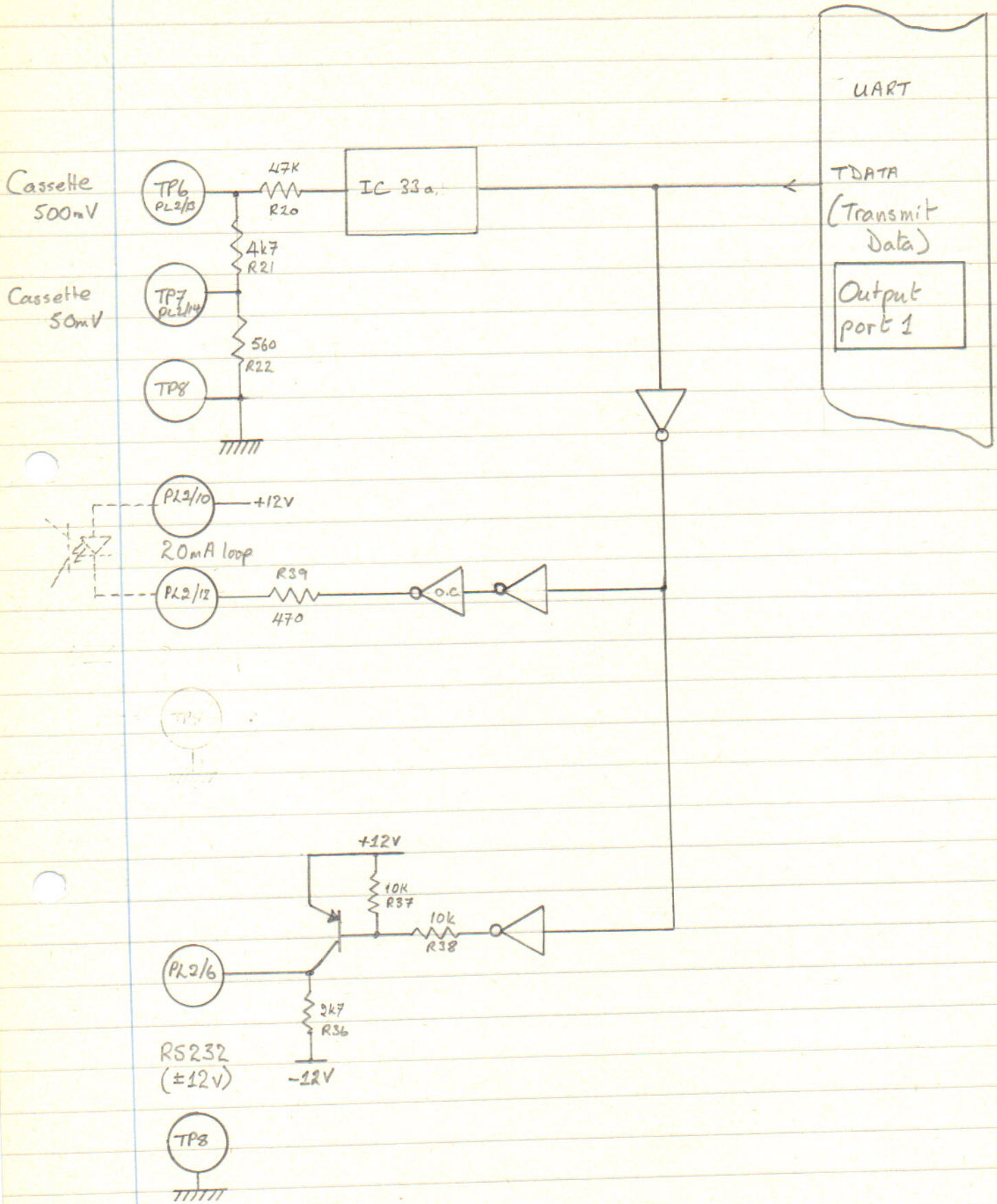
Setting the receive and transmit speeds and the number of stop bits of the serial i/o Port.

Serial input circuits, simplified.



Serial Input Circuits, simplified.

Serial output circuits, simplified



Serial Output Circuits, simplified.

1.4 INPUT AND OUTPUT TABLES

Each input and output device on the Nascom has a device driver, ie. a routine which drives the hardware so as to achieve the required data transfer. Thus, for example, driver SRLIN inputs a byte from the serial input port, input port 1 (the UART receiver), while device driver CRT outputs to the video display. Refer to Chapter 3 for a description of the drivers.

Often data transfers are required to be made to or from more than one device. For example, it is usually convenient for a routine such as BLINK to accept input from either the keyboard or the serial input port. BLINK might have been written so that it always requests input from either of these devices. However, the more flexible method used is for the input and output routines to refer to a table which lists the device drivers to be used by the routine. These tables may be changed by the user at any time so directing input and output to selected devices only.

Following power-up or reset, NAS-SYS initialises workspace location $IN (0C75/6) to 077C (NAS-SYS 3), 0782 (NAS-SYS 1). The following tables show that the input routines will thus read the keyboard and the serial input port. Workspace location $OUT (0C73/4) is similarly initialised to 0779 (NAS-SYS 3), 077F (NAS-SYS 1), thus directing the output routines to write to the video display only.
A subsequent U command changes $IN to 077B (NS3), 0781 (NS1) so adding device driver UIN to the list of input devices, while $OUT is changed to 0778 (NS3), 077E (NS1) so adding device driver UOUT to the list of output devices.
The X command changes $IN to 077F(NS3), 0785 (NS1) so including device driver XKBD, while $OUT is changed to 0777 (NS3), 077D (NS1) so including XOUT, UOUT, and CRT.

OUTPUT TABLES
Address of start of table held in $OUT (OC73/4)

NS1 loc.	NS1 name	NS3 loc	NS3 name	Driver number	Driver			
077A	OUTT2	0774	OUTT2	65	CRT			G
077B	OUTT3	0775	..	6F	SRLX			
077C	..	0776	..	00	End of table.			
077D	OUTTX	0777	OUTTX	6E	XOUT			Xn
077E	OUTTU	0778	OUTTU	75	UOUT		U	
077F	OUTT1	0779	OUTT1	65	CRT	N		
0780	..	077A	..	00	End of table.			

INPUT TABLES
Address of start of table held in $IN (OC75/6)

NS1 loc.	NS1 name	NS3 loc.	NS3 name	Driver number	Driver			
0781	INTU	077B	INTU	76	UIN			U
0782	INT1	077C	INT1	61/7D	KBD/RKBD		N	
0783	INT3	077D	..	70	SRLIN			
0784	..	077E	..	00	End of table.			
0785	INTX	077F	INTX	74	XKBD		X	
0786	INT4	0780		61/7D	KBD/RKBD			
0787	..			00	End of table.			

$UOUT (OC77/8/9) is C3 - - ,jump to user written output driver.
$UIN (OC7A/B/C) is C3 - - ,jump to user written input driver.

1.5 Use of NMI by NAS-SYS

The very useful facility of being able to step through a user program one instruction at a time is achieved by driving the single-step circuit from NAS-SYS. Essentially, the single-step command arms the single-step circuit of the NASCOM so that, after execution of one user program instruction, an NMI (non-maskable interrupt) is generated.
The NMI service routine displays the contents of the cpu registers as they were at the end of the instruction and then waits for a further keyboard command.

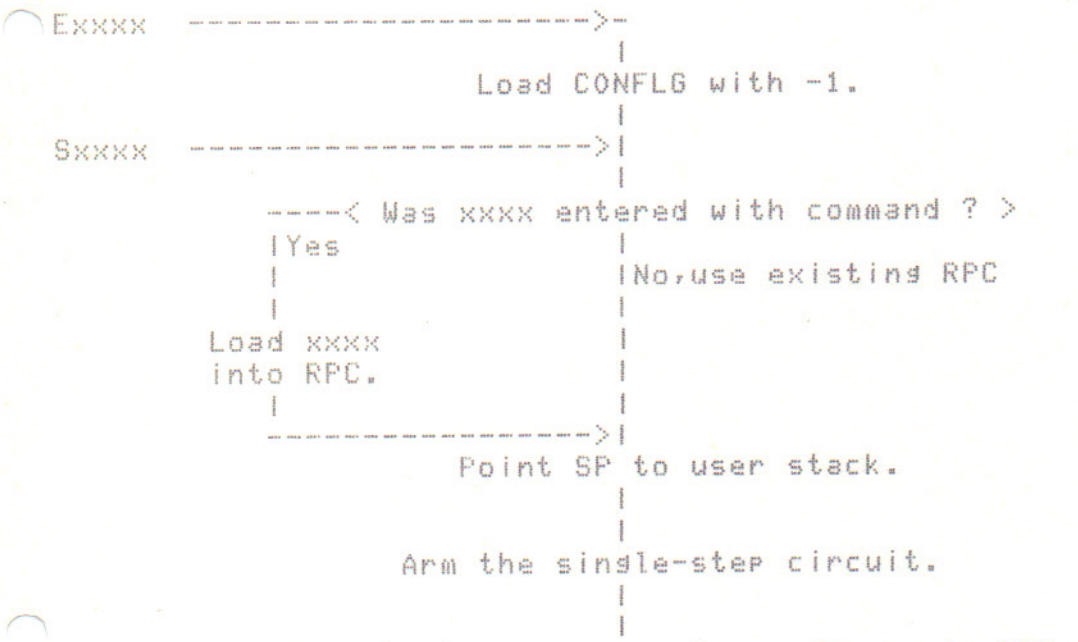
In response to an NMI, the Z80 pushes the program counter onto the stack and then executes the code in location 0066H, which is a JP $NMI instruction. At workspace location $NMI (OC7DH) is the instruction JP TRAP where TRAP is the start of the NMI service routine.

An NMI may also be generated by a push-button which, when pressed, brings connector PL3/4 to zero volts. This button will

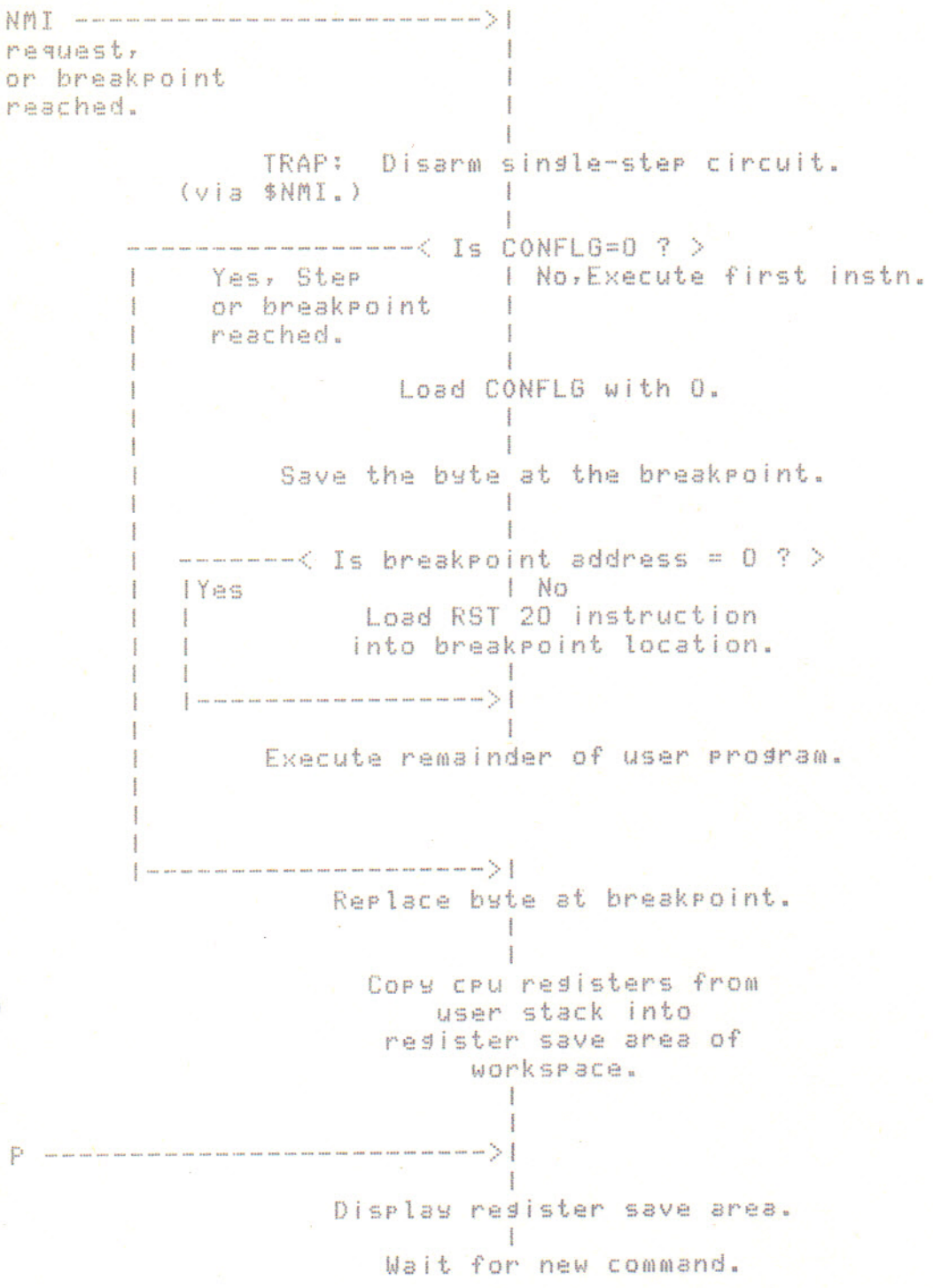
then function as a 'break' key, stopping program execution and displaying the register contents.

A third method of generating an NMI is by driving line 21 of NASBUS from an external device. Such a device will usually require a different response to that of the break key described above. This may be achieved by loading workspace locations $NMI+1/2 with the address of the start of the service routine, which must terminate with a RETN instruction.

The following diagram shows the relationship of the NMI service routine to NAS-SYS 1 commands E and S, which are described in Chapter 3. The NAS-SYS 1 coding differs slightly but performs the same function.



(First instruction of user program always generates an NMI.)


NMI request, or breakpoint reached.
TRAP: Disarm single-step circuit.
(via $NMI.)
Is CONFLG=0 ?
Yes, Step or breakpoint reached.
No,Execute first instn.
Load CONFLG with 0.
Save the byte at the breakpoint.
Is breakpoint address = 0 ?
Yes
No
Load RST 20 instruction into breakpoint location.
Execute remainder of user program.
Replace byte at breakpoint.
Copy cpu registers from user stack into register save area of workspace.
P
Display register save area.
Wait for new command.

1.6 Memory Maps

2.1 DESCRIPTION OF COMMANDS

In this chapter each command is described in alphabetical order. These commands are normally entered from the keyboard but they may be incorporated into user programs as described in Section 2.2.2. Each description begins with the following information:

Command line from keyboard	eg A xxxx yyyy
Command in ZEAP-type assemblers	eg SCAL "A
Command in other assemblers	eg RST 18H:DEFB 41H
Command in machine code	eg DF 41

ARITHMETIC COMMAND

```
A xxxx yyyy
SCAL "A
RST 18H:DEFB 41H
DF 41
```

This performs simple arithmetic in hexadecimal, then returns to NAS-SYS. Three results are displayed, viz:

SSSS DDDD JJ where SSSS = yyyy + xxxx
DDDD = yyyy - xxxx
JJ = the displacement required in a Jump Relative instruction which starts at location xxxx and is to jump to yyyy. If the displacement required is too large, ?? is displayed.

e.g. A23 35
0058 0012 10

BREAKPOINT COMMAND

```
B xxxx
SCAL "B
RST 18H:DEFB 42H
DF 42
```

Argument xxxx is stored in workspace location BRKADR (0C23/4), and then the command returns to NAS-SYS.

Following this command, when the user program reaches address xxxx the program execution is stopped and the current contents of the cpu registers displayed. The user program may be restarted from the breakpoint address by giving the command 'E', ie. the Execute command without an argument.

See also the S command.

An Execute command following a B command causes the byte in location xxxx to be replaced by E7, the RST 20 instruction. The original contents of location xxxx is stored in location BRKVAL (0C25). If the breakpoint is reached during execution of the user

program, a jump is made to location $NMI (0C7D/E/F). These locations are initialised by NAS-SYS 3 to contain the instruction JP TRAP. The TRAP routine restores the byte at the breakpoint, saves the cpu registers in the register save area of the NAS-SYS workspace and displays the contents of the registers in the same format as the single step command (refer to S command). Breakpoints may be set only at the first byte of an instruction since the RST 20 can only replace an existing operation code.

COPY COMMAND

```
C xxxx yyyy zzzz
SCAL "C
RST 18H:DEFB 43H
DF 43
```

The Copy command copies memory locations in the block xxxx to xxxx+zzzz-1 to the block yyyy to yyyy+zzzz-1. The transfer begins by writing location xxxx to location yyyy and increments through the block. Note that it is possible to lose data if the two areas overlap.(The intelligent copy command,I,avoids this possibility.)

To fill memory locations xxxx to xxxx+zzzz-1 with the same byte, put the required byte into location xxxx using the M command, then use the C command C xxxx xxxx+1 zzzz. eg. to fill 0E00 to 0EFF with 33, deposit 33 into location 0E00 using M0E00 then C E00 E01 FF.

JUMP TO D000H (NAS-SYS 3 only)

```
D
```

The D command simply starts executing the code beginning at location D000. This will normally be the start of a proprietary program in rom such as ZEAP.

EXECUTE COMMAND

```
E xxxx
SCAL "E
RST 18H:DEFB 45H
DF 45
```

Execute the code beginning at location xxxx. If xxxx is omitted the address used is that stored in the register save area of the NAS-SYS workspace at location RPC (0C69/A).

Arguments may be entered into a program using this command: see section 4.4, Program 4.

Note that the command does **not** simply load the program counter with xxxx and allow program execution from there. The first instruction of a user program will generate an NMI as though that instruction were being single-stepped. However, the contents of the cpu registers are not displayed as they would be using the S, single-step command because workspace location CONFLG (0C26) is set to a non-zero value by the E command, so signalling to NAS-SYS that the E command was called rather than the S command. User program instructions other than the first are executed normally until a breakpoint (if set) is reached.

Since an interrupt will move the cpu past a HALT instruction and the first instruction of the user program when Executed always generates an interrupt, it follows that if the first instruction is HALT, the NASCOM will not in fact halt. To get round this, program a NOP as the first instruction.

F

This command letter is not used in NAS-SYS; a call is made to the routine which displays Error.

GENERATE COMMAND

```
G   xxxx  yyyy zzzz
SCAL "G
RST 18H:DEFB 47H
DF 47
```

Generate writes a header comprising some commands to NAS-SYS followed by program bytes from memory locations xxxx to yyyy-1 on to a cassette tape. When the tape is subsequently read, memory locations from xxxx to yyyy-1 are loaded from the tape and program execution automatically started at location zzzz.

The output to the tape is:

(cr)	Newline
E0 (cr)	Restart NAS-SYS
R (cr)	Read a tape
data	Program code as in W command
E zzzz (cr)	Execute from location zzzz

A program stored on tape using this command can be loaded and run without any commands from the keyboard since the necessary commands are input from the cassette itself.

HALF DUPLEX COMMAND

```
H
SCAL "H
RST 18H:DEFB 48H
DF 48
```

The H command causes the Nascom simply to input characters from the devices in the input table and output them to the devices in

the output table.
With the normal tables in operation, the Nascom simply inputs characters from either the keyboard or the serial input port and outputs them to the video display. By using the command X 30 before the H command, input is accepted both from the keyboard and from an external serial device (this latter using device handler XKBD, chapter 3) and outputs to an external serial device (using device handler XOUT, chapter 3) as well as to the video display. Half duplex communication may thus be established with another serial input-output device.

Since the routine executes INLIN (chapter 3) repeatedly, the only way to exit from this command is to reset the Nascom.

INTELLIGENT COPY COMMAND

```
I xxxx yyyy zzzz
SCAL "I
RST 18H:DEFB 49H
DF 49
```

The I command copies memory locations in the block xxxx to xxxx+zzzz-1 to the block yyyy to yyyy+zzzz-1. Unlike the Copy command, if the two blocks overlap, data is not lost.

JUMP TO FFFA

```
J
SCAL "J
RST 18H:DEFB 4AH
DF 4A
```

The J command simply starts executing the code at location FFFA. This is the cold start entry point for 8K ROM BASIC. See also the Z command.

KEYBOARD COMMAND

```
K x
SCAL "K
RST 18H:DEFB 4BH
DF 4B
```

The Keyboard command changes the characteristics of the keyboard.

x	Effect
0	Shift key produces lower case characters.
1	Shift key produces upper case characters.
4	Effect of Graphics key is reversed: keys produce graphics characters unless Graphics key is pressed when alphanumeric characters are produced.
5	Combines the effect of 1 and 4.

See routine KBD, chapter 3.

LOAD COMMAND
(NAS-SYS 1)

```
L
SCAL "L
RST 18H:DEFB 4C
DF 4C
```

This command is used to read data stored on paper tape using the NAS-SYS 1 Tabulate command. The checksum which is output by the Tabulate command is used to check the integrity of the input data. Where the checksum indicates an error, that line is scrolled up the screen so that those locations may be corrected by the M command.

MODIFY COMMAND

```
M xxxx
SCAL "M
RST 18H:DEFB 4D
DF 4D
```

The memory modify command allows memory locations to be displayed and, if the memory is read/write, allows the contents of the location to be changed.

The display shows xxxx followed by the current contents of location xxxx.

To change the contents of the location, simply type in the new hexadecimal value, followed by Enter. The location contents are changed following the Enter.
Where several consecutive bytes are to be modified, as when entering a program in machine code, the bytes may be typed on one line with a space between them. Do not however type beyond the end of a display line before typing Enter. A leading zero does not need to be typed, eg 0C 04 may be typed as C 4.

To exit from the M command, type a full-stop followed by Enter.

To go back to the previous address, type a colon (:) followed by Enter.

To examine a non-consecutive location, yyyy, type /yyyy followed by Enter.

To enter ASCII codes, as when programming a message, type a comma before each character of the message, eg ,A,*,b loads consecutive locations with 41H, 2AH, 62H which are the corresponding ASCII codes.

Invalid characters are indicated by the display 'Error'.

NORMAL I/O COMMAND

```
N
SCAL "N
SCAL 4EH
RST 18H:DEFB 4EH
```

The N command sets the input and output tables to 'normal'. The output table is simply CRT so that output is directed only to the video display. The input table is RKBD (KBD in NAS-SYS1) and SRLIN so that input is taken either from the keyboard or the serial input port.

See also commands U and X which change the input/output tables.

OUTPUT COMMAND

```
O xx yy
```

The output command sends data to an output port. Output port xx is loaded with data yy.
eg. O 4 6F sends byte 6fH to port 04H.

Note that if the port is physically a PIO, the control register of the PIO must first be loaded with the appropriate code to program it as an output port.
See also the Q command.

PROGRAM REGISTER DISPLAY
(NAS-SYS 3 only)

```
P
SCAL "P
SCAL 50H
RST 18H:DEFB 50H
DF 50
```

The Program registers display command displays the contents of the Z80 cpu registers as stored in the register save area of the workspace.
The Z80 registers are automatically saved in the register save area whenever a breakpoint is reached, or after each instruction step when using the S command (which, incidentally,invokes this command to produce the display), or when RST 20H (E7H) is executed.
For the format of the display refer to the S command.

QUERY COMMAND Q xx

The Query command inputs data from an input port and displays it in hexadecimal.
eg. Q 4 inputs data from port 04H and displays it.
Note that if the port is physically a PIO, the control register of the PIO must first be loaded with the appropriate code to make it an input port.
See also the O command.

READ CASSETTE TAPE COMMAND

```
R xxxx
SCAL "R
SCAL 52H
RST 18H:DEFB 52H
DF 52H
```

The Read cassette tape command reads a tape that was written using the W (Write cassette) command. Data is normally loaded into the same memory locations from which it was written. However, NAS-SYS 3 allows a displacement, xxxx, to be added to the addresses of the data so that if memory locations 1000H to 1400H were written to the tape, command R 3000 loadds the data into memory locations 4000H to 4400H. The tape may also be read into lower addresses, eg if data were written from locations 4000H onwards, command R F000 will load the data into locations 3000H onwards since 4000H + F000H = (1)3000H.

At the start of the command execution, the tape led is switched on. Input from the serial input port or the keyboard is read and discarded until four start of block characters (FFH) are read (or four ESCape characters-see below).
The four start of block characters indicate the start of the next block of data on the tape. If the checksum for the header data is incorrect, a question mark is displayed and the data following is ignored. Otherwise, as the block is read the screen displays:

```
SSSS  BBLL
```

where SSSS is the start address of the block
BB is the block number,decrementing.
LL is the number of bytes in the block, normally 00 (=256).

If the data checksum is correct, a full stop is displayed; otherwise, a question mark is displayed indicating that data in the block just loaded is corrupt. The tape may be rewound for a length of tape greater than one block and then replayed.
At the endo of the command the tape led is turned off.

Because both the cassette input and the keyboard provide input data, no keyboard keys should normally be pressed during a tape read: doing so will produce errors in the loaded data which will be indicated by a question mark.

If the ESCape key is pressed four times after the end of one block and before the start of the next, the Read command will be aborted.

SINGLE STEP COMMAND S xxxx

The Single Step command allows a program to be stepped through one program instruction at a time.

S xxxx executes the instruction beginning at memory location xxxx.At the end of the instruction (which may be one to four bytes long) the register contents are stored in the register save area of the workspace and displayed. The display format is given below.
Pressing Enter will cause the next program instruction to be executed and the registers again displayed. If a repeating keyboard is used (as in NAS-SYS 3 or a patched NAS-SYS 1) the Enter key may be held down to single step at high speed.

If the S command is entered without xxxx being specified, the command uses the saved program counter stored in the register save area of the workspace. This allows the user to single step following a breakpoint simply by typing S.

Program Debugging
The S command together with the B command provide a most useful program debugging facility. Often a breakpoint will be set in the program being debugged and the program executed from the start, using the E command. At the breakpoint the registers are examined. Any of these registers may be changed by using the M command to modify the register save area of the workspace, and then the next program instruction executed by typing S. If no registers are changed the next instruction may be executed simply by pressing Enter. Alternatively, the remainder of the program may be executed by typing E.

The workspace locations which store the saved registers are:

0C68	Register A
0C67	Register F
0C62	Register B
0C61	Register C
0C64	Register D
0C63	Register E
0C66	Register H
0C65	Register L
0C6A	Program Counter,high byte
0C69	Program Counter,low byte
0C6C	Stack Pointer,high byte
0C6B	Stack pointer,low byte

Display Format.NAS-SYS 3

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

The two bytes displayed after the first six registers or register pairs are the contents of the location pointed to by that register pair.
eg. A display beginning :-

```
OFFD 21A7  22B6  C8F5  0380  2790 .....
```

shows that the SP is OFFD,
and the contents of location OFFD is A7H
and the contents of location OFFE is 21H,

the PC is 22B6,
and the contents of location 22B6 is F5H
and the contents of location 22B7 is C8H,

the A register is 03H,
the F register is 80H,
and the contents of location 0380 is 90H
and the contents of location 0381 is 27H.

The Flags part of the display is a parsed version of the F register; in the example shown, Flags will show S, indicating that the Sign flag is set (F register is 80H).

Display Format: NAS-SYS 1

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

Note that in both displays, -PC- shows the address of the next instruction to be executed.

TABULATE COMMAND
(NAS-SYS 3)

```
T xxxx yyyy zzzz vv hhaa
SCAL "T
SCAL 54H
RST 18H:DEFB 54H
DF 54
```

The Tabulate command displays the contents of a block of memory locations.

xxxx — is the first location displayed.

yyyy — is one more than the last location displayed

zzzz — determines the number of lines displayed at a time; press any key to display the next group of lines; press ESC to abort the command. If zzzz is zero, all the locations will be tabulated without pause.

vv — determines the number of bytes on each line. 8+vv bytes will be displayed. eg. if vv=04H there will be twelve bytes per line (04H + 08H = 0CH); if vv=FCH there will be four bytes per line (FCH + 08H = 04H)

hh — If hh is non-zero, the hexadecimal tabulation is suppressed.

aa — If aa is non-zero, the ASCII tabulation is supressed.

Each display line begins with a memory address followed by the contents of that location (in hexadecimal) followed by the contents of several succeeding locations. At the right-hand part of the display line, the same data is displayed by showing the ASCII characters corresponding to the data bytes. Hexadecimal values 00-1F, 7F-9F, and FF are shown as a full stop.

The Tabulate command is the only one using more than three parameters.

TABULATE COMMAND
(NAS-SYS 1)

```
T xxxx yyyy zzzz
SCAL "T
SCAL 54H
RST 18H: DEFB 54H
DF 54
```

This is similar to the Tabulate command in NAS-SYS 3; however, only the first three arguments of that command are utilised. The display does not include the ASCII characters corresponding to the data bytes. Instead, at the end of each line of eight data bytes, a checksum is output. This is backspaced over on the display so appears only momentarily. By activating the X command, the contents of a block of memory may be dumped onto a serial output device, such as a paper tape punch, and then loaded back into the computer using the L (Load) command.

USER I/O COMMAND

```
U
SCAL "U
SCAL 55H
RST 18H:DEFB 55H
```

The U command brings user-written input and output routines into the input/output tables. See Chapter 1.4.

The output table is set to include UOUT and CRT so that output is directed to the users output routine as well as to the video display.

The input table is set to include UIN, RKBD (or KBD in NAS-SYS 1), and SRLIN, so that input is taken from the users input routine, the keyboard, or the serial input port.

The start address of the users output routine must be placed in

workspace location $U0UT+1 (0C78/9). The start address of the users input routine must be placed in $UIN+1 (0C7B/C). These locations are initialised by NAS-SYS to point to a RET instruction, so that if these locations are not changed, the U command has no effect.

The U command is automatically suspended during the execution of the G,R,V,and W commands.

The N command (which see) changes the input/output tables so to exclude user-written routines.

For an example of a user-written output routine see Chapter 4.13.

VERIFY TAPE COMMAND

```
V
SCAL "V
SCAL 56H
RST 18H:DEFB 56H
DF 56
```

The Verify Tape command simply reads a cassette tape in the same manner as the R command (which see) but does not enter the data into memory.
The command is used to check that a tape can be read without error.

WRITE COMMAND

```
W xxxx yyyy
SCAL "W
SCAL 57H
RST 18H:DEFB 57H
DF 57
```

The Write command is used to write data to a cassette tape. Data from memory locations xxxx to yyyy-1 inclusive are transmitted from the serial port. Note that, because all the serial outputs are physically connected, the data is also transmitted to the RS232 output and the 20mA output at the same time.

The data is transmitted in blocks of 256 bytes, except for the last block which may have fewer bytes. The format of each block is:

00	Nul
FF FF FF FF	Four start-of-block characters
SL SH	Start address, low byte first
LL	Number of data bytes in this block,(00=256)
BB	Block number: decrements to zero for each suceeding block
CC	Checksum for the header data
data bytes	Normally 256 bytes
EE	Checksum for the data bytes

00 (x10) Ten nul characters

As each block is transmitted, the following data is displayed:
SSSS BBLL
where SSSS = the memory address of the first byte of the block being transmitted
BB = the block number; the last block is numbered 00
LL = the number of bytes in the data block:00=256

At the beginning of the Write command, the tape led is illuminated, there is a brief delay, and 256 nul characters (00) are transmitted. The overall format of the tape is thus:

00 (x 256) 256 nuls
Block n
Block n-1
...
...
...
Block 1
Block 0

EXTERNAL SERIAL DEVICE COMMAND

```
X xx
SCAL "X
SCAL 58H
RST 18H:DEFB 58H
DF 58
```

The External serial device command places xx into workspace location $XOPT and sets the start of the input table to INTX (NAS-SYS 3) and the start of the output table to OUTTX. The input table is thus XKBD and RKBD while the output table is XOUT, UOUT, and CRT. See Chapter 1.4.
The device handlers XOUT and XKBD (see chapter 3) allow both the incoming and the outgoing data via the serial input and output ports to be modified slightly so as to meet the requirements of various external serial devices such as a VDU, printer, or a mainframe computer. Thus, for example, the parity of the output data may be set to either odd or even, and a line feed character may be automatically inserted after a carriage-return.

The characteristics of the serial link are determined by the individual bits in xx, as follows:-

Bit 0 0 Output characters have even parity.
1 Output characters have odd parity.

Bit 1 NAS-SYS 3 only. 0 If input character is CR, bit 4 determines whether or not LF is output. Also, any character, other than NUL and ESC, will not be output by the next call to the output routine, XOUT.

	1	If input character is CR, LF is not output, and the next call to the output routine, XOUT, will produce an output.
Bit 4	0	If CR is output then output LF. If bit 1=0 and bit 4=0 and input character is CR, then output LF.
	1	If CR is input or output do not output LF.
Bit 5	0	All input characters are echoed with parity determined by bit 0.
	1	Input characters are not echoed.
Bit 7		This bit is used by XKBD to determine whether or not a subsequent call to XOUT will actually output a character.

Some examples:

XX	Characteristics
22	Bits 5 and 1 set. Input characters are not echoed, bit 5 =1. An output CR is automatically followed by an output LF, bit 4=0. Output parity is even, bit 0 =0.
23	Bits 5, 4, and 0 set. As for 22 but odd parity,bit 0=1.
32	Bits 5, 4, and 1 set. As for 22 but an output CR is not followed by a LF, bit 4=1.
33	Bits 5,4,1,and 0 set. As for 32 but odd parity, bit 0=1.
20	Bit 5 set. Input characters are not echoed, bit 5=1. Follow a CR with a LF, bit 4=0. Suppress output via XOUT of all characters other than NUL and ESC, bit 1=0. Output parity is even, bit 0=0.
21	Bits 5 and 0 set. As for 20 but odd parity, bit 0=1.
30	Bits 5 and 4 set. As for 20 but LF not output after CR, bit 4=1.
31	Bits 5,4, and 0 set. As for 30, but odd parity, bit 0=1.

For a detailed description of XKBD and XOUT, see chapter 3.

JUMP TO B000H
(NAS-SYS 3 only)

```
Y
SCAL "Y
SCAL 59H
RST 18H:DEFB 59H
DF 59
```

The Y command simply starts executing the code at location B000H.

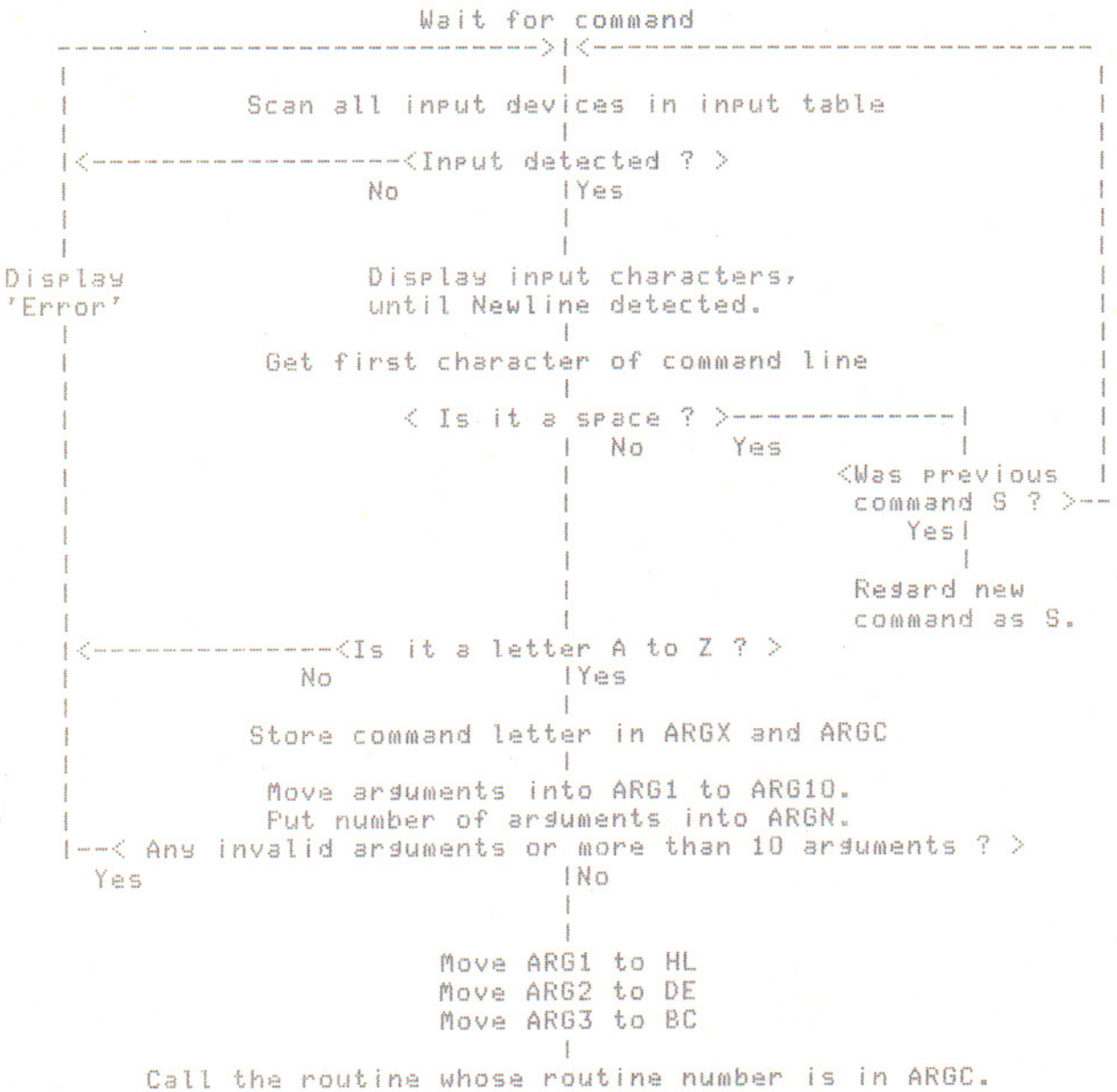
JUMP TO FFFDH

```
Z
SCAL "Z
SCAL 5AH
RST 18H:DEFB 5AH
DF 5A
```

The Z command simply starts executing the code at location FFFDH. This is the warm start entry point for 8K ROM BASIC. See also the J command.

2.2 Waiting for commands

When NAS-SYS is blinking the cursor and waiting for a command, it is executing the procedure shown below:



2.3 Commands within a user program

Commands may be called within a user program by coding the instructions shown at the beginning of the description of the command in Section 2.1. If the command has arguments these must be entered into appropriate registers before calling the command. The first argument must be placed in register pair HL, the second argument in register pair DE, and the third argument in register pair BC. Some commands also require that the arguments be stored

in workspace locations ARG1, ARG2,etc.
The following examples show the method of using commands within a program.

eg1. Programming an A command.

```
                        ;Do a A 1234 0234 command
0D00  21 34 12          LD HL,1234H       ;First argument
0D03  01 34 02          LD BC,0234H       ;Second argument
0D06  DF 41             SCAL  "A          ;Call A command
                        ;
                        ;Do a A 0011 0022 command
0D08  21 11 00          LD HL,0011H       ;First argument
0D0B  01 22 00          LD BC,0022H       ;Second argument
0D0E  DF 41             SCAL "A           ;Call A command
                        ;
                        ;Return to NAS-SYS
0D10  DF 5B             SCAL 5BH
```

eg2. Programming the R,T, and W commands.

This example, which is listed below, begins by writing consecutive hexadecimal numbers into locations 0E00 to 0EFF. This portion of memory is then tabulated using the T command. A prompt is given to start the cassette recorder and the data is written to tape. After waiting for the tape to be rewound, the memory locations are cleared to zero and a second tabulation done to show that the memory is in fact clear. The tape is then read and a third tabulation shows that the tape data has been transferred into memory.

The first call to the T command, line 230, illustrates the standard method of calling a command. The arguments are loaded into workspace locations ARG1 to ARG5. The first three arguments are then transferred to HL,DE, and BC respectively at line 380 using the subroutine ARGS (refer to Chapter 3).

The call to the W command, line 480, is similarly preceded by loading the arguments into ARG1 and ARG2. (There is no copy of these arguments in registers HL and BC since the W command does this transfer within itself.)

The second and third calls to the T command, lines 750 and 940, are abbreviated versions of the first. This is possible because the third, fourth, and fifth arguments remain unchanged in workspace locations ARG3, ARG4, and ARG5; although ARG1 and ARG2 have been changed by the call to the W command this is of no consequence since the T routine takes the first two arguments from HL and DE which are loaded immediately before the call.

The call to the R command, line 850, is a little different to the others. This is because both the R and V commands call up the same routine, the only functional difference being whether or not the input data is transferred to ram. The common routine checks

to see if the command was R or V to determine whether or not to make the transfer. The ASCII code for the command must therefore be placed in workspace location ARGX and, if no argument is used with the R command, zero must be placed in workspace location ARGN.

Listing of example 2.

```
2D00                    0010        ORG  2D00H
                        0020 ;NAS-SYS ROUTINE NUMBERS:
2D00 0060               0030 ARGS   EQU  60H
2D00 0061               0040 KBD    EQU  61H
2D00 005B               0050 MRET   EQU  5BH
2D00 0028               0060 PRS    EQU  28H
                        0070 ;NAS-SYS WORKSPACE:
2D00 0C0C               0080 ARG1   EQU  0C0CH
2D00 0C0E               0090 ARG2   EQU  0C0EH
2D00 0C10               0100 ARG3   EQU  0C10H
2D00 0C12               0110 ARG4   EQU  0C12H
2D00 0C0B               0120 ARGN   EQU  0C0BH
2D00 0C2B               0130 ARGX   EQU  0C2BH
2D00 0C14               0140 ARG5   EQU  0C14H
                        0150 ;FILL LOCATIONS 0E00 TO 0EFF WITH
                        0160 ; BINARY COUNT.
2D00 0600               0170        LD   B,0
2D02 21000E             0180        LD   HL,0E00H
2D05 75                 0190 FILL   LD   (HL),L
2D06 23                 0200        INC  HL
2D07 10FC               0210        DJNZ FILL
                        0220 ;
                        0230 ;DO A T 0E00 0F00 0000 00 0000 COMMAND.
                        0240 ;PUT FIRST ARGUMENT INTO ARG1.
2D09 21000E             0250        LD   HL,0E00H
2D0C 220C0C             0260        LD   (ARG1),HL
                        0270 ;PUT SECOND ARGUMENT INTO ARG2.
2D0F 21000F             0280        LD   HL,0F00H
2D12 220E0C             0290        LD   (ARG2),HL
                        0300 ;PUT THIRD ARGUMENT INTO ARG3.
2D15 210000             0310        LD   HL,0       ;ARG3,ARG4,ARG5
2D18 22100C             0320        LD   (ARG3),HL
                        0330 ;PUT FOURTH ARGUMENT INTO ARG4.
2D1B 22120C             0340        LD   (ARG4),HL
                        0350 ;PUT FIFTH ARGUMENT INTO ARG5.
2D1E 22140C             0360        LD   (ARG5),HL
                        0370 ;MOVE ARGUMENTS INTO REGISTERS.
2D21 DF60               0380        SCAL ARGS
                        0390 ;NOW CALL THE TABULATE COMMAND.
2D23 DF54               0400        SCAL "T
                        0410 ;
                        0420 ;TELL USER TO START RECORDER.
2D25 EF                 0430        RST  PRS
2D26 53746172           0440        DEFM /Start recorder now./
     74207265
     636F7264
     6572206E
     6F772E
```

Listing of example 2, continued.

```
2D39 0D00      0450          DEFB 0DH,0
               0460 ;
               0470 ;
               0480 ;DO A W 0E00 0F00 COMMAND.
               0490 ;PUT FIRST ARGUMENT INTO ARG1.
2D3B 21000E    0500          LD   HL,0E00H
2D3E 220C0C    0510          LD   (ARG1),HL
               0520 ;PUT SECOND ARGUMENT INTO ARG2.
2D41 21000F    0530          LD   HL,0F00H
2D44 220E0C    0540          LD   (ARG2),HL
               0550 ; NOW CALL THE W COMMAND.
2D47 DF57      0560          SCAL "W
               0570 ;
2D49 EF        0580          RST  PRS
2D4A 57726974  0590          DEFM /Write done./
     6520646F
     6E652E
2D55 0D        0600          DEFB 0DH        ;NEW LINE ON DISPLAY
2D56 52657769  0610          DEFM /Rewind tape then press <space>./
     6E642074
     61706520
     7468656E
     20707265
     7373203C
     73706163
     653E2E
2D75 00        0620          DEFB 0
               0630 ;WAIT FOR AN INPUT.
2D76 DF61      0640 WAIT     SCAL KBD    ;READ KEYBOARD
2D78 30FC      0650          JR   NC,WAIT
               0660 ;
               0670 ;CLEAR 0E00 TO 0EFF.
2D7A AF        0680          XOR  A
2D7B 47        0690          LD   B,A
2D7C 21000E    0700          LD   HL,0E00H
2D7F 77        0710 CLEAR    LD   (HL),A
2D80 23        0720          INC  HL
2D81 10FC      0730          DJNZ CLEAR
               0740 ;
               0750 ;DO A T 0E00 0F00 0000 00 0000 COMMAND.
2D83 21000E    0760          LD   HL,0E00H    ;FIRST ARGUMENT
2D86 11000F    0770          LD   DE,0F00H    ;SECOND ARGUMENT
               0780 ;Other arguments have not changed.
2D89 DF54      0790          SCAL "T
               0800 ;
2D8B EF        0810          RST  PRS
2D8C 53746172  0820          DEFM /Start tape now./
     74207461
     7065206E
     6F772E
2D9B 0D00      0830          DEFB 0DH,0
               0840 ;
```

Listing of example 2, continued

```
                0850 ;DO AN R COMMAND
                0860 ;PUT ASCII CODE FOR R INTO ARGX.
2D9D 3E52       0870        LD   A,52H
2D9F 322B0C     0880        LD   (ARGX),A
                0890 ;PUT ZERO INTO ARGN; NO ARGUMENTS FOR R.
2DA2 AF         0900        XOR  A
2DA3 320B0C     0910        LD   (ARGN),A
                0920 ;CALL THE R COMMAND
2DA6 DF52       0930        SCAL "R
                0940 ;DO A T 0E00 0F00 0000 00 0000 COMMAND.
2DA8 21000E     0950        LD   HL,0E00H
2DAB 11000F     0960        LD   DE,0F00H
2DAE DF54       0970        SCAL "T
                0980 ;
                0990 ;RETURN TO NAS-SYS.
2DB0 DF5B       1000        SCAL MRET
```

2.4 Changing the Commands.

There is no difference in the way commands and subroutines are processed by NAS-SYS. Both are accessed from a subroutine table which lists the start addresses of both commands and routines.

2.4.1 How NAS-SYS accesses the routines

Each command and routine has a number which indicates its position in the subroutine address table which begins at location STABA (0782,NS3; 0788,NS1). The table starts with the 'A' command (routine number 41H) and the first two locations of the table give the address at which the 'A' command routine begins. Since the first routine is numbered 41H (this being convenient since it is the ASCII code for A), the conceptual start of the table (ie. the table base address, STAB) is 82H less than STABA, ie. 0782 - 82 = 0700 in NS3 or 0788-82=0706 in NS1.
The program instruction SCAL "A (= SCAL 41H = RST 18H:DEFB 41H = DF 41) causes NAS-SYS to read the byte 41H, multiply it by two and add it to the table base address. The result is the location of the 'A' command in the table, which gives the address of the start of routine number 41H. NAS-SYS then jumps to that address so executing the required routine.

As another example, suppose SCAL MRET (=SCAL 5BH) is programmed. NAS-SYS 3 calculates STAB + 2 x 5B,
ie. 0700 + 2 x 5B = 07B6.
The two bytes at location 07B6/7 are the address of the start of the MRET routine; NAS-SYS then jumps to this address and so executes MRET.

The address of the start of the table of routines, ie. STAB is held in workspace location $STAB (0C71/2). The memory space is thus:

Location	Contents
$STAB (0C71)	Address of the conceptual start of table,
(0C72)	normally STAB = STABA-82 = 0700(NS3),0706(NS1)
	points to
STAB normally	Conceptual start of table.
	.
	.
	.
STABA normally	Address of A command, routine number 41H.
	Address of B command, routine number 42H.
	.
	.
	.
	Address of CPOS, routine number 7CH.

2.4.2 Changing the subroutine table.

Because the address of the start of the table is held in ram, it is possible for a user to change it so that it points to another table. This alternative table can be defined by the user. As an example, suppose an 'F' command is to be added to the list of commands, the other commands remaining unchanged. (An 'F' command in both current versions of NAS-SYS simply calls the ERRM routine which writes 'Error' on the screen.)
The new F command is to simply write "F routine" on the video display, and is assembled from location FCMD. The new table begins at location NEWTAB. The program begins by copying the existing table to location NEWTAB. The address of the new F command in the new table is then changed to make it FCMD. Finally, $STAB is changed to point to the conceptual start of the table, ie. 82H bytes before NEWTAB.

Listing of program to add an 'F' command.

```
ZEAP Z80 Assembler - Source Listing

                    0010 ;ADDING AN F COMMAND
                    0020 ;
0D00                0030        ORG  0D00H
                    0040 ;
                    0050 ;NAS-SYS routine numbers
0D00 005B           0060 MRET   EQU  5BH
0D00 0028           0070 PRS    EQU  28H
                    0080 ;
                    0090 ;NAS-SYS workspace
0D00 0C71           0100 $STAB  EQU  0C71H
0D00 0782           0110 STABA  EQU  0782H   ;0788 FOR NS1.
0D00 0C80           0120 NEWTAB EQU  0C80H
                    0130 ;
                    0140 ;Copy existing table to NEWTAB.
0D00 218207         0150        LD   HL,STABA
0D03 11800C         0160        LD   DE,NEWTAB
0D06 017E00         0170        LD   BC,7EH
0D09 EDB0           0180        LDIR
                    0190 ;Change address of F command in new table.
0D0B 21190D         0200        LD   HL,FCMD
0D0E 228A0C         0210        LD   (NEWTAB-82H+"F+"F),HL
                    0220 ;Make $STAB point to NEWTAB.
0D11 21FE0B         0230        LD   HL,NEWTAB-82H
0D14 22710C         0240        LD   ($STAB),HL
                    0250 ;Return to NAS-SYS.
0D17 DF5B           0260        SCAL MRET
                    0270 ;
                    0280 ;
                    0290 ;New F command
0D19 EF             0300 FCMD   RST  PRS
0D1A 4620726F       0310        DEFM /F routine/
     7574696E
     65
0D23 0D00           0320        DEFB 0DH,0
0D25 C9             0330        RET
```

0D00

Assemble the above code into ram and execute from location ~~0C00.~~ Then check that an 'F' command produces the expected response. Check also that the other commands remain unchanged.

The memory used by the program is shown below. The program may be relocated elsewhere in ram simply by redefining the origin of the program and the location of NEWTAB.

```
                  ----------------
        0D25
                  New F command
                    routine.
        0D19(FCMD)
                  ----------------
        0D18      Routine to change
                     the table.
        0D00      ----------------

                  Space for
                  new table.

        0C80(NEWTAB)----------------
```

An F command which is more useful is listed below. Its purpose is to allow the user to change the register contents when single stepping through a program. The command description is:

FIX REGISTERS Frs hhll
or Fs ll

The Fix command changes register r to hh, and register s to ll, where r and s are:

r/s	Register referenced
0	none
1	I
2	IY
3	IX
4	PC,H
5	PC,L
6	SP,H
7	SP,L
8	H
9	L
A	A
B	B
C	C
D	D
E	E
F	F

Thus, command FA 76 changes register A to 76H,
F45 1234 changes the PC to 1234.

Commands Fx2 hhll and Fx3 hhll, where x represents any other

register reference, are automatically changed to F02 hhll and F03 hhll respectively.
Commands F2x hhll and F3x hhll are illegal and generate an error message.
Following the F command, pressing the Enter key causes a single step from the location indicated by the PC.

The routine begins by placing the ASCII code for S into workspace location ARGX which stores the last command letter. This allows a subsequent Enter key to continue single-stepping.
NAS-SYS loads argument rs into register pair HL and argument hhll into register pair DE (see chapter 2.2). A correctly entered command should make register H zero and this is checked at line 170. Register L is then copied into register H.
At line 240 argument r is extracted from register H and checked to see if it is 2 or 3. If so subroutine ERR displays Error.
At line 350 argument s is extracted from register L and stored in register pair BC. If argument s is 2 or 3, argument r is forced to 0 by subroutine RZER. Subroutine PROC is then called to change the register.
At line 460 argument r is stored in register pair BC and argument hh moved to register E. Subroutine PROC then changes the register.
Finally, the P command is called to display the register contents.

Subroutine PROC is entered with one of the arguments r or s in register pair BC and the new data in register E.If the register number is 0, the subroutine returns without making any changes. Register numbers 1 (I), 2 (IY), and 3 (IX) are changed directly; these registers are not stored in the register save area of the workspace following a breakpoint or single step. Register numbers 4 and up are stored in the workspace (the actual locations are given in chapter 2.1, S command). These locations are accessed and modified via the table TAB which gives only the lower bytes of the addresses in the register save area, the upper byte always being 0CH.

To implement this new F command, replace the trivial code at location FCMD (0D19) in the previous program with the following code; then execute from location 0D00.

Listing of FIX REGISTERS

```
                    0010 ;FIX REGISTERS
                    0020 ;Frs hhll
                    0030 ;Change reg r to hh, reg s to ll.
                    0040 ;
0D19                0050         ORG  0D19H
                    0060 ;
                    0070 ;NAS-SYS routine numbers
0D19 006B           0080 ERRM    EQU  6BH
0D19 0028           0090 PRS     EQU  28H
                    0100 ;
                    0110 ;NAS-SYS workspace
0D19 0C2B           0120 ARGX    EQU  0C2BH
                    0130 ;
                    0140 ;
                    0150 ;For single-step
0D19 3E53           0160 FCA     LD   A,"S
0D1B 322B0C         0170         LD   (ARGX),A
                    0180 ;
                    0190 ;Check that H is zero, else Error.
0D1E 7C             0200         LD   A,H
0D1F B7             0210         OR   A
0D20 2803           0220         JR   Z,F1
0D22 DF6B           0230         SCAL ERRM
0D24 C9             0240         RET
0D25 65             0250 F1      LD   H,L
                    0260 ;Get arg r.
0D26 CB3C           0270         SRL  H
0D28 CB3C           0280         SRL  H
0D2A CB3C           0290         SRL  H
0D2C CB3C           0300         SRL  H
                    0310 ;Check for r = 2 or 3.
0D2E 7C             0320         LD   A,H
0D2F FE02           0330         CP   2
0D31 2824           0340         JR   Z,ERR
0D33 FE03           0350         CP   3
0D35 2820           0360         JR   Z,ERR
                    0370 ;Get argument s.
0D37 7D             0380         LD   A,L
0D38 E60F           0390         AND  0FH
0D3A 4F             0400         LD   C,A
0D3B 0600           0410         LD   B,0
                    0420 ;Check for s=2 or 3.
0D3D FE02           0430         CP   2
0D3F CC540D         0440         CALL Z,RZER
0D42 FE03           0450         CP   3
0D44 CC540D         0460         CALL Z,RZER
0D47 CD5A0D         0470         CALL PROC
                    0480 ;Get argument r.
0D4A 4C             0490         LD   C,H
0D4B 0600           0500         LD   B,0
0D4D 5A             0510         LD   E,D
0D4E CD5A0D         0520         CALL PROC
                    0530 ;Display registers
0D51 DF50           0540         SCAL "P
0D53 C9             0550         RET
                    0560 ;
                    0570 ;Force r to 0 if s=2 or 3.
```

```
0D54 2600      0580 RZER   LD   H,0
0D56 C9        0590        RET
               0600 ;r=2 or 3 is illegal.
0D57 DF6B      0610 ERR    SCAL ERRM
0D59 C9        0620        RET
               0630 ;
               0640 ;On entry BC holds 000r or 000s.
               0650 ;          new byte in E,
               0660 ;          double byte in DE.
0D5A 79        0670 PROC   LD   A,C
0D5B B7        0680        OR   A
0D5C C8        0690        RET  Z
               0700 ;Is reg no.= 1?
0D5D FE01      0710        CP   1
0D5F 2004      0720        JR   NZ,P2
0D61 7B        0730        LD   A,E
0D62 ED47      0740        LD   I,A
0D64 C9        0750        RET
               0760 ;Is reg no.=2?
0D65 FE02      0770 P2     CP   2
0D67 2004      0780        JR   NZ,P3
0D69 D5        0790        PUSH DE
0D6A FDE1      0800        POP  IY
0D6C C9        0810        RET
               0820 ;Is reg no.=3?
0D6D FE03      0830 P3     CP   3
0D6F 2004      0840        JR   NZ,P4
0D71 D5        0850        PUSH DE
0D72 DDE1      0860        POP  IX
0D74 C9        0870        RET
               0880 ;Reg no. is 4 up.
0D75 E5        0890 P4     PUSH HL
0D76 217D0D    0900        LD   HL,TAB-4
0D79 09        0910        ADD  HL,BC
0D7A 7E        0920        LD   A,(HL)
0D7B 6F        0930        LD   L,A
0D7C 260C      0940        LD   H,0CH        ;HL is 0Cxx
0D7E 73        0950        LD   (HL),E
0D7F E1        0960        POP  HL
0D80 C9        0970        RET
               0980 ;
               0990 ;
0D81 6A696C6B  1000 TAB    DEFB 6AH,69H,6CH,6BH
0D85 66656862  1010        DEFB 66H,65H,68H,62H,61H,64H,63H,67H
     61646367
```

3.1 QUICK REFERENCE GUIDE TO SUBROUTINES

a) Routines which input from the keyboard only:

KBD	Keyboard driver
RKBD (NAS-SYS 3 only)	Keyboard driver with repeat key

b) Routines which output to the video display only:

CRT	Video display driver

c) Routines which input from the screen only:

CPOS
RLIN

d) Routines which input from the serial port only:

SRLIN	Serial input driver
XKBD	External keyboard driver

e) Routines which output to the serial port only:

SOUT

SRLX	Serial output driver

XOUT

f) Routines for user-defined input/output:

$UIN
$UOUT

g) Routines which input from devices in the input table:

BLINK
IN
INLIN
RIN

h) Routines which output to devices in the output table:

B1HEX
B2HEX
CRLF
ERRM
PRS
ROUT
SPACE
SP2 (NAS-SYS 3 only)
TBCD2
TBCD3
TX1

i) Routines which call other routines:

ATE
RCAL
SCAL
SCALI (NAS-SYS 3 only)
SCALJ
START

j) Routines which alter the input/output tables:
   NIM
   NOM
   NNIM
   NNOM

k) Routines which generate delays:
   RDEL
   TDEL

l) Routines which alter bits of output port 0:
   FFLP
   MFLP

m) Other routines:
   ARGS
   BRKPT
   NUM
   MRET
   STMON

3.2 DESCRIPTIONS OF SUBROUTINES IN ALPHABETICAL ORDER

```
SCAL ARGS
SCAL 60H
RST 18H:DEFB 60H
DF 60
```

ARGS simply transfers the contents of workspace locations into cpu register pairs:

ARG1 into HL
ARG2 into DE
ARG3 into BC.

ARG1 is locations 0C0C/D,
ARG2 is locations 0C0E/F,
ARG3 is locations 0C10/1.

ADDRESS TABLE EXECUTION

```
SCAL ATE
SCAL 73H
RST 18H:DEFB 73H
DF 73
```

ATE automatically calls a sequence of other routines which are listed in a table. On entry, the start address of the table is in register pair HL. The routine ends when either the Carry flag is set (the input device handlers set the Carry flag when an input is detected) or when the end of the table is reached.
The table pointed to by HL comprises the numbers of the routines to be called, terminated by 00. The routines are normally input or output routines. Thus to automatically execute CRT and KBD (ie. to output to the video display and then scan the keyboard, the table would be 65H(CRT), 61H(KBD),00(terminator).

Since facility exits within NAS-SYS for changing the input and output tables used by the input and output device drivers listed in section 3.1g and h, routine ATE will rarely need to be used explicitly.

See also commands U and X in chapter 2, and NIM, NOM, NNIM, NNOM in this section.

```
SCAL B1HEX
SCAL 7AH
RST 18H:DEFB 7AH
DF 7A
```

B1HEX transmits the ASCII code for the hexadecimal representation of the low order nibble (ie. the right-hand four bits) of register A to the devices in the output table. The screen display shows this as a single hexadecimal character, 0 to F. The serial output port transmits the ASCII code for this chracter, eg. if register A contains 1AH, the byte 41H is transmitted, this being the ASCII code for character A.

SCAL B2HEX

```
SCAL 68H
RST 18H:DEFB 68H
DF 68
```

B2HEX transmits the ASCII codes for the hexadecimal representation of the contents of register A to the devices in the output table. The screen display shows the contents of register A as two hexadecimal digits. The serial output port transmits the ASCII code for the two chracters, eg. if register A contains 4AH, the bytes 34H and 41H are transmitted, these being the ASCII codes for characters 4 and A respectively.

```
SCAL BLINK
SCAL 7BH
RST 18H:DEFB 7BH
DF 7B
```

BLINK waits for an input from the devices in the input table, (normally the keyboard and the serial input port) returning only when an input is received. While waiting for an input, the cursor on the screen is blinked. The routine returns with the ASCII code for the input character in register A.

NAS-SYS 3 allows the blink rate of the cursor to be set by the user. Workspace location KBLINK (0C32/3), which is initialised to 0100H, may be altered.

```
RST BRKPT
RST 20H
E7
```

BRKPT copies the Z80 cpu registers into the register save area of the workspace, displays them, and then returns control to NAS-SYS. The register display format is the same as that described under 'command P', chapter 2.

```
SCAL CPOS
SCAL 7CH
RST 18H:DEFB 7CH
DF 7C
```

On entry register pair HL points to a position on the video display. On exit, HL points to the start of that line on the screen.

```
SCAL CRLF
SCAL 6AH
RST 18H:DEFB 6AH
DF 6A
```

CRLF transmits the byte 0DH (ASCII code for carriage return) to all the devices in the output table. The screen display routine moves the cursor to the beginning of the next line.
The routine simply loads register A with 0DH (the newline code) and then calls ROUT.

```
SCAL CRT
SCAL 65H
RST 18H:DEFB 65H
DF 65
```

CRT is the video display handler. It writes the character whose ASCII code is in register A onto the screen at the current cursor position. The character may be a screen edit command.

```
eg 1        3E 41        LD  A,41H  Write A on screen at
            DF 65        SCAL CRT   current cursor position.

eg 2        3E 07        LD  A,07H  Write BEL character at
            DF 65        SCAL CRT   current cursor position.
```

```
SCAL ERRM
SCAL 6BH
RST 18H:DEFB 6BH
DF 6B
```

ERRM transmits the bytes 45H, 72H, 72H, 6FH, 72H (the ASCII codes for 'Error') to all the devices in the output table; the code for a carriage return/line feed follows the Error message.
The screen display shows the message 'Error' at the current cursor position.
The routine is simply a call to PRS with the ASCII codes for Error as the string, followed by a call to CRLF.

```
SCAL FFLP
SCAL 5EH
RST 18:DEFB 5EH
DF 5E
```

FFLP changes the state of one or more bits of output port 0, then immediately restores them to their original state. The bits that are changed are the same as those bits of register A which contain a 1. For example, if register A contains 24H or 00100100 then bits 2 and 5 will be pulsed.

Most of the bits of output port 0 are used to drive the keyboard; before using this command refer to Charter 1.1.

```
SCAL IN
SCAL 62H
RST 18H:DEFB 62H
DF 62
```

IN scans the devices in the input table once. If an input is detected from any device, the routine returns with the input data in register A and the Carry flag set. Otherwise the Carry flag is reset.

```
SCAL INLIN
SCAL 63H
RST 18H:DEFB 63H
DF 63
```

INLIN blinks the cursor and displays input from any of the devices in the input table. When the Enter code is received (normally from the keyboard Enter key), the routine returns with register pair DE holding the address in video ram of the start of the line in which the cursor was when the Enter code was received.

```
SCAL KBD
SCAL 61H
RST 18H:DEFB 61H
DF 61
```

KBD scans the keyboard once. If a key press is detected, the routine returns with the ASCII code for the key in register A and the Carry flag set. Otherwise the Carry flag is reset.
Using the shift, control, and graphics keys of the NASCOM 2 keyboard all 256 codes may be generated from the keyboard.

The contents of workspace location $KOPT (OC27) control the characteristics of the keyboard as follows:

$KOPT	Keyboard characteristic
0	Shift key produces lower case characters.
1	Shift key produces upper case characters.
4	Effect of Graphics key is reversed: keys produce graphics characters unless Graphics key is pressed when alphanumeric characters are produced.
5	Combines the effect of 1 and 4.

The Kn command may be used to set the contents of $KOPT to n.

KBD should be used in preference to RIN to input from the keyboard whenever a spurious input from the serial port would be undesirable. To wait for a keyboard input use the following:

```
                PUSH  ???       Save active registers.
DF 61  KBDW     SCAL KBD        Scan keyboard
30 FC           JR NC,KBDW      If no keypress, jump.
                POP   ???       Restore registers.
```

See also RKBD.

```
SCAL MFLP
SCAL 5FH
RST 18H:DEFB 5FH
DF 5F
```

MFLP changes the state of bit 4 of output port 0. This port is used primarily to drive the keyboard and the single-step logic; bit 4 is used to drive LED2 which illuminates when the cassette recorder should be switched on during commands G,R,V,and W. It is possible to use this signal, available at TP10 on Nascom2) to switch a cassette, having a remote control plug, on and off.

```
SCAL MRET
SCAL 5BH
RST 18H:DEFB 5BH
DF 5B
```

MRET is the normal return from a user program back to NAS-SYS. It resets the top of the monitor stack and sets the user stack top to 1000H, displays the logo "--NAS-SYS n --", resets the breakpoint byte, and then waits for a keyboard command.

```
SCAL NIM
SCAL 72H
RST 18H:DEFB 72H
DF 72
```

NIM changes the start of the table of input device handlers to the location pointed to by register pair HL.
Thus to use an input table which starts at location INDEV, program:

```
21 xx xx    LD  HL,INDEV
DF 72       SCAL NIM
```

On exit register pair HL holds the address of the start of the input table in use before calling this routine.

See also commands U and X in chapter 2 and routines NOM, NNIM, and NNOM in this section.

```
SCAL NOM
SCAL 71H
RST 18H:DEFB 71H
DF 71
```

NOM changes the start of the table of output device handlers to the location pointed to by register pair HL.
Thus to use an output table which starts in location OUTDEV, program:

```
21 xx xx    LD HL, OUTDEV
DF 71       SCAL NOM
```

On exit register pair HL holds the address of the start of the output table in use before calling this routine.

See also commands U and X in chapter 2 and routines NIM, NNIM, and NNOM in this section.

```
SCAL NNIM
SCAL 78H
RST 18H:DEFB 78H
DF 78
```

NNIM sets the start of the table of input device handlers to location INT1 within NAS-SYS3. This table contains RKBD and SRLIN in NAS-SYS 3; KBD and SRLIN in NAS-SYS 1.

On exit register pair HL holds the address of the start of the input table in use before calling this routine.

See also commands U and X in chapter 2 and routines NIM, NOM, and NNOM in this section.

```
SCAL NNOM
SCAL 77H
RST 18H:DEFB 77H
DF 77
```

NNOM sets the start of the table of output device handlers to location OUTT1 within NAS-SYS 3. This table contains CRT only.

On exit register pair HL holds the address of the start of the output table in use before calling this routine.

See also commands U and X in chapter 2 and routines NIM, NOM, and NNIM in this section.

```
SCAL NUM
SCAL 64H
RST 18H:DEFB 64H
DF 64
```

NUM converts a number, expressed as up to four ASCII codes, into

hexadecimal.

On entry, the start of the number (which may be preceded by spaces) is pointed to by the contents of register pair DE. On exit, workspace location NUMV (0C21/2) holds the hexadecimal form of the number and NUMN (0C20) holds the number of hexadecimal digits in the number.

eg. If the number is 1E9 (stored as 31H,45H,39H), after calling NUM, NUMV/NUMV+1 holds 01E9H and NUMN holds 03, the number of digits.

The Carry flag is set if an invalid character is encountered.

```
                                        RST  PRS
                                        RST  28H
                                        EF
```

PRS outputs a string of bytes to the devices in the output table. The string of bytes must follow immediately after the call to PRS, and the string must be terminated with byte 00H.

```
eg.             EF              RST     PRS         Call PRS
                46 72 65 64     DEFM    /Fred/      String.
                0D   00         DEFB    0DH,0       Newline   and
                                                    terminator.
```

Since the output table normally contains CRT and SRLX, the bytes appear on the display as Fred followed by a newline, and the bytes 46,72,65,64, and 0D are transmitted from the serial output port.
Note that screen editing commands may be incorporated into the string,

```
        eg.     EF          RST PRS
                0C 00       DEFB 0CH,0      clears the screen.

                EF          RST PRS
                1B  00      DEFB 1BH,0      clears the line in which the
                                             cursor lies.
```

```
                                        RCAL disp
                                        RST 10H:DEFB disp
                                        D7 disp
```

RCAL disp produces a call to a subroutine in a similar manner to the Z80 instruction CALL address. However, whereas the CALL instruction requires the absolute address of the CALLed subroutine to be encoded within the instruction, RCAL requires that the relative displacement of the subroutine be encoded in the same manner as the Z80 JR (Jump Relative) instructions. Thus it may be regarded as a Relative CALl.
As in all the instructions using a relative address, the displacement (disp) must be evaluated bearing in mind that the

actual address referred to is the current program count plus the displacement, ie PC + 2, and that the PC holds the address of the next instruction. Thus,

```
0643 D7 15    RCAL 15H
```

produces a call to the subroutine beginning at 065A, since 0643 + 2 + 15 = 065A.
The advantage gained by using RCAL is that the machine code may be relocated anywhere in memory, ie. it is position independent code. The price paid is the extra processing time taken by RCAL to evaluate the required address.

```
RST RDEL
RST 38H
FF
```

RDEL generates a delay proportional to the number stored in register A. (00 in register A is interpreted as 256.) The delay produced by this routine itself, excluding the delay in calling it, is 44 * (contents of reg A - 1) + 17 clock cycles. If the Z80 clock is 2MHz, each cycle takes 500 ns. Thus delays between 8.5micros and 5.62ms may be produced. If the clock frequency is 4MHz, the delay times are halved.

See also TDEL.

```
RST RIN
RST 08H
CF
```

RIN repeatedly reads in turn all the input devices in the input table until an input is detected from one of them. On returning, the input byte is in register A and the Carry flag is set.

The routine itself calls routine IN repeatedly until an input is detected.

NAS-SYS 3 only.

```
SCAL RKBD
SCAL 7DH
RST 18H:DEFB 7DH
DF 7D
```

RKBD is the keyboard driver which supports repeating keys, ie. a key held down for longer than a certain period will be regarded as having been pressed again and again. (The @ key does not repeat since it functions as a CTRL key.)
Two periods are involved: the initial period for which a key must be held down before it is regarded as being pressed again, and the period which determines the repeat speed after the initial period.
The initial period is determined by workspace location KLONG (0C2E/F), and is set by NAS-SYS to 0280H.
The repeat speed is determined by workspace location KSHORT (0C30/1), and is set by NAS-SYS to 0050H. Either or both of KLONG and KSHORT may be changed by the user.

```
SCAL RLIN
SCAL 79H
RST 18H:DEFB 79H
DF 79
```

RLIN examines a line on the video display and extracts the arguments from the line.
On entry, register pair DE points to the memory location which is the start of the line.
On exit, The first argument is held in workspace location ARG1 (0C0C/D), the second argument in ARG2 (0C0E/F), etc. The number of arguments is held in ARGN (0C0B).

If more than ten arguments or a non-hexadecimal argument is encountered, the Carry flag is set.

```
ie.       X aaaa bbbb cccc .......jjjj        n arguments
          |   |    |    |           |         |
        (DE) ARG1 ARG2 ARG3     ARG10         ARGN
```

```
RST ROUT
RST 30H
F7
```

ROUT writes the character whose ASCII code is in register A to the output devices in the output tble.

The character may be a screen edit command, eg 11H (CUL,cursor left), 16H (CSR, move rest of line to the right so creating a space).

```
SCAL xx
RST 18H:DEFB xx
DF xx
```

SCAL (Subroutine CALl) is the routine by which all the other routines (with the exception of the RSTs) are called. The number of the called routine is xx. Thus, SCAL RLIN = SCAL 79H calls the routine RLIN.
The byte following the SCAL (DFH) is a displacement down a table which contains the actual addresses of the NAS-SYS routines. By using this technique other versions of NAS-SYS may be produced while still retaining compatability with the previous versions, ie. programs using a certain routine in one version will still function correctly when using the same routine in another version.

NAS-SYS 3 only

```
SCAL SCALI
SCAL 7FH
RST 18H:DEFB 7FH
DF 7F
```

SCALI is almost the same as SCALJ. The only difference is that

the number of the routine to be called must be in register E.

```
SCAL SCALJ
SCAL 5CH
RST 18H:DEFB 5CH
DF 5C
```

SCALJ calls the routine whose number is in workspace location ARGC (0C0A).
eg. If ARGC contains 54H, the Tabulate command is called since the Tabulate routine is numbered 54H,(the ASCII code for T).

The arguments for the command must be stored in workspace locations ARG1, ARG2, etc and the first three arguments must be in HL, DE, and BC.

See also SCALI.

```
SCAL SOUT
SCAL 6DH
RST 18H:DEFB 6DH
DF 6D
```

SOUT transmits a series of bytes to the serial output port.

The start address of the string must be loaded into register pair HL and the length of the string in register B. At the end of transmission, register C holds the eight bit checksum of all the bytes transmitted.

NAS-SYS 3 only

```
SCAL SP2
SCAL 7EH
RST 18H:DEFB 7EH
DF 7E
```

SP2 is the equivalent of two SPACEs, described in the following paragraph.

```
SCAL SPACE
SCAL 69H
DF 69
```

SPACE transmits the byte 20H (ASCII code for a space) to all the devices in the output table.

This routine simply loads register A with 20H and then calls ROUT.
See also SP2.

```
SCAL SRLIN
SCAL 70H
RST 18H:DEFB 70H
DF 70
```

SRLIN scans the serial input port once. If data is ready, the routine returns with the data in register A and the Carry flag set. Otherwise, the Carry flag is cleared.

The serial port may be connected to a variety of different devices. Refer to Chapter 1 for the physical connections and speed switch settings.

```
SCAL SRLX
SCAL 6FH
RST 18H:DEFB 6FH
DF 6F
```

SRLX transfers the contents of register A to the serial output port and waits for transmission to be completed.
See Chapter 1 for setting up the serial port.

```
SCAL TBCD2
SCAL 67H
RST 18H:DEFB 67H
DF 67
```

TBCD2 transmits the ASCII codes for the hexadecimal representation of the contents of register A to the devices in the output table. The byte in register A is added into register C, so accumulating an eight bit checksum.

The video display shows the contents of register A as two hexadecimal digits. The serial output port transmits the ASCII codes for the two digits.

```
SCAL TBCD3
SCAL 66H
RST 18H:DEFB 66H
DF 66
```

TBCD3 transmits the ASCII codes for the headecimal representation of the contents of register pair HL to the devices in the output table. These bytes are followed by the ASCII code for a space (20H). The contents of registers H and L are added into register C which thus accumulates an eight bit checksum.

```
SCAL TDEL
SCAL 5DH
RST 18H:DEFB 5DH
DF 5D
```

TDEL generates a delay of 2.9s if the Z80 clock rate is 2MHz. At 4MHz the delay is halved.

```
SCAL TX1
SCAL 6CH
RST 18H:DEFB 6CH
DF 6C
```

TX1 transmits (to the devices in the output table) the ASCII codes for the hexadecimal representation of the contents of register pair HL, followed by the ASCII code for a space (20H), followed by the ASCII codes for the hexadecimal representation of the contents of register pair DE, followed by the ASCII code for a space.

The contents of registers H,L,D, and E are added into register C which thus accumulates an eight bit checksum.

```
SCAL UIN
SCAL 76H
RST 18H:DEFB 76H
DF 76
```

UIN allows a user-written input device driver to be added to the input table.
UIN causes a jump to workspace location $UIN (0C7A). This location contains C3H, the op code for the Z80 JP instruction. The two following bytes are initialised by NAS-SYS to an address within the ROM which contains the byte C9H, the op code for the Z80 RET instruction.
A user-written input device handler may be called by placing the handler start address in the two btes followin $UIN, ie.0C7B/C. Exit from the handler should be via one of the RET instructions with the Carry flag set only if an input was received.
See UOUT.

```
SCAL UOUT
SCAL 75H
RST 18H:DEFB 75H
DF 75
```

UOUT allows a user-written output device handler to be accessed via the output table. The start address of the handler should be loaded into workspace locations 0C78/9. These locations are initialised by NAS-SYS to the address of a location within the ROM containing the op code for the Z80 RET instruction. Thus UOUT has no effect unless the address is changed.
See UIN.

```
SCAL XOUT
SCAL 6EH
RST 18H:DEFB 6EH
DF 6E
```

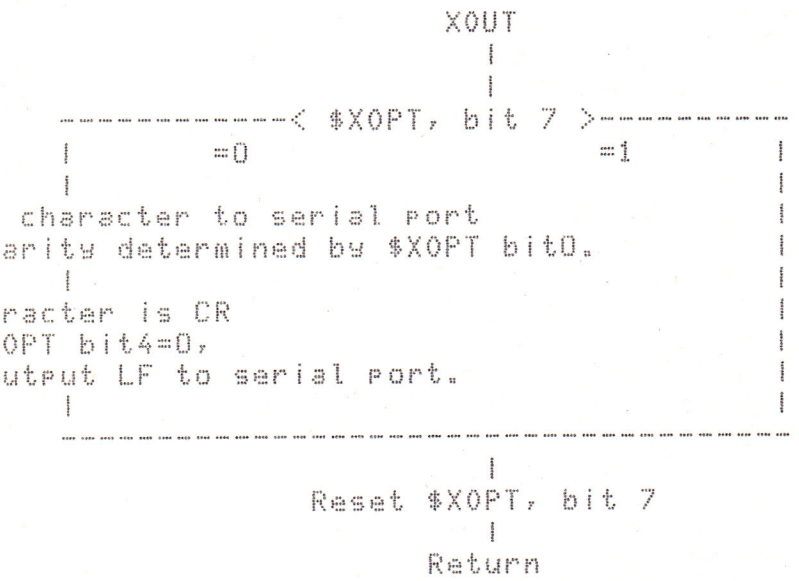
XOUT is the handler for output to an external serial device. Individual bits of workspace location $XOPT (0C28) determine the characteristics of the handler, as follows:-

Bit 0=0 transmit even parity.
     1 transmit odd parity.

Bit 4=0 if CR is transmitted, also transmit LF.
     1 no LF appended after CR.

Bit 7=0 output characters are transmitted.
     1 output characters are not transmitted.

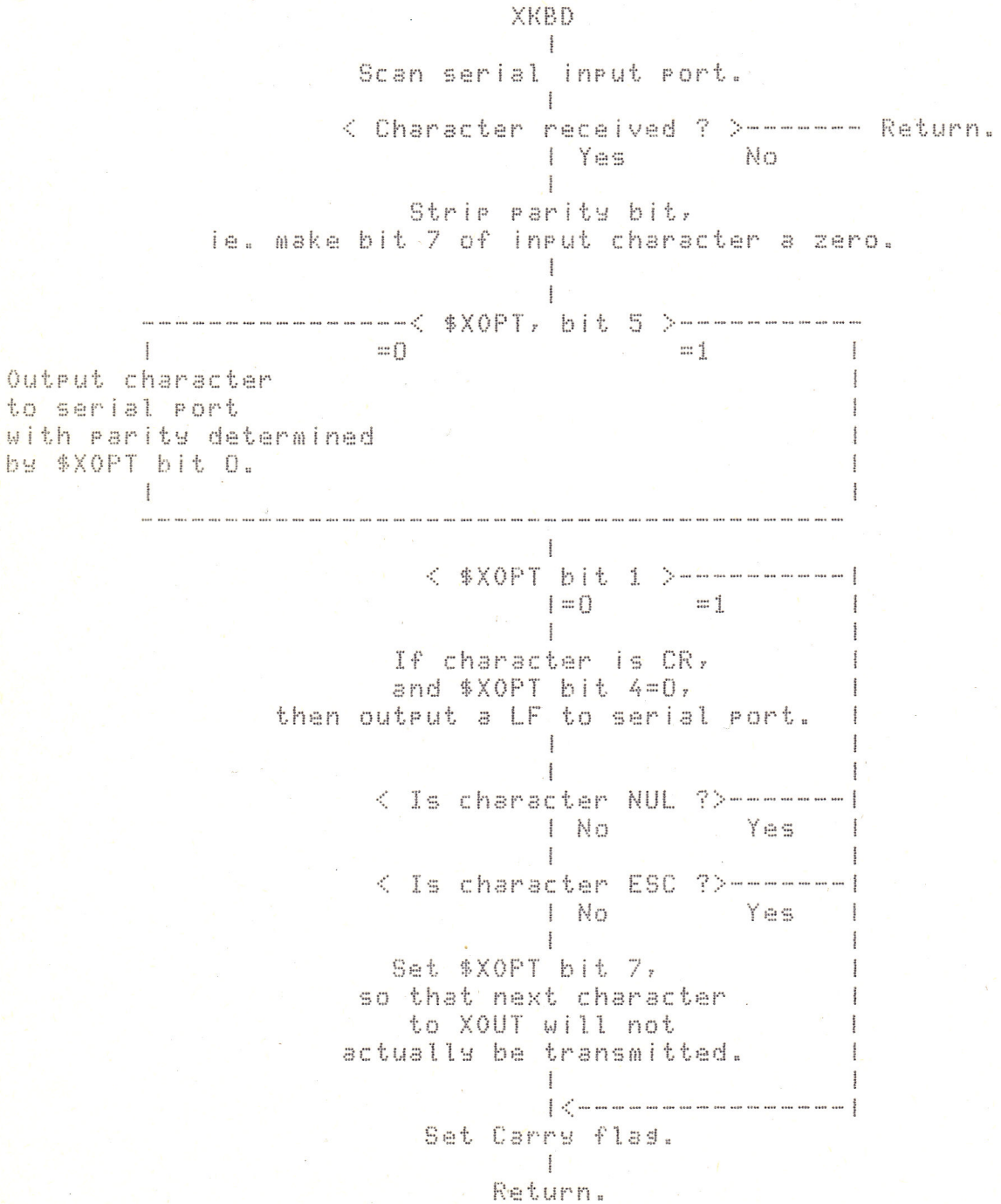
Flowchart:



See also the X command, charter 2.

```
                                        SCAL XKBD
                                        SCAL 74H
                                        RST 18H:DEFB 74H
                                        DF 74
```

XKBD is the handler for input from an external serial device. Individual bits of workspace location $XOPT (0C28) determine the characteristics of the handler: these are shown on the following flowchart.



4.0 Guide to the Examples

Number	Name	Used in Example number:
5B	MRET	2 3 8 9 10 13
5C		8
5D		1 4 6 12
5E		6
5F		6 12
60		4 5
61		11
62	IN	.
63		5 7 12
64		7
65		9 10 11
66		3 5 7 10
67		3
68		3 4 5
69		7
6A		1 4 5
6B		1
6C		3 5
6D		12
6E	XOUT	.
6F		11
70		.
71		10
72		9 10
73		.
74		.
75		.
76		.
77		.
78		.
79		5
7A		3
7B		1 9 10
7C		2
7D		.
7E		.
7F		.
RST 0	START	.
RST 8	RIN	.
RST 10	RCAL	14
RST 18	SCAL	all
RST 20	BRKPT	.
RST 28	PRS	1 2 3 5 7 10
RST 30	ROUT	1 10
RST 38	RDEL	10

4.1 PROGRAM 1

Shows the use of routines BLINK,CRLF,ERRM,PRS,RIN,ROUT, and TDEL.

The program simply prompts the user to press one of the keys 0,1 or 2. The character corresponding to the pressed key is displayed together with the message 'Error' if it was not a valid input. After a delay the program restarts.

The program begins, line 130, by using PRS to prompt the user to press 0,1, or 2. Note that the display control characters, 0CH (clear screen) and 0DH (newline) are incorporated into the string displayed by PRS.

At line 170 the program waits for a character from one of the devices in the input table (normally the keyboard and serial input). The input character in register A is saved by tranferring it to register C because the following PRS routine at line 190 will change the contents of register A. This PRS routine is used to display 'You entered ' without a following new line so that the entered character, now back in register A is output to the devices in the output table using ROUT, line 230. A new line is generated by CRLF at line 240.

Lines ~~230-290~~ 250 -310 check the validity of the entered character and direct program control either to line ~~310~~ 330 or ~~320~~ 340. The error message is displayed using ERRM, a new line generated by CRLF, and a delay produced by TDEL.

Note that the call to BLINK, line 170, may be replaced by RST RIN. The cursor will then not be blinked while waiting for an input.

Listing of Program 1

```
                0010 ;PROGRAM 1
                0020 ;
2D00            0030        ORG  2D00H
                0040 ;
                0050 ;Table of NAS-SYS subroutine numbers.
2D00 007B       0060 BLINK  EQU  7BH
2D00 006A       0070 CRLF   EQU  6AH
2D00 006B       0080 ERRM   EQU  6BH
2D00 0028       0090 PRS    EQU  28H
2D00 0030       0100 ROUT   EQU  30H
2D00 005D       0110 TDEL   EQU  5DH
                0120 ;
2D00 EF         0130 START  RST  PRS
2D01 0C         0140        DEFB 0CH                ;Clear screen.
2D02 50726573   0150        DEFM /Press 0,1,or 2.../
     7320302C
     312C6F72
     20322E2E
     2E
2D13 0D00       0160        DEFB 0DH,0              ;New line,End
2D15 DF7B       0170        SCAL BLINK              ;Wait for input.
2D17 4F         0180        LD   C,A                ;Save character.
2D18 EF         0190        RST  PRS
2D19 596F7520   0200        DEFM /You entered /
     656E7465
     72656420
2D25 00         0210        DEFB 0
2D26 79         0220        LD   A,C                ;Get saved char.
2D27 F7         0230        RST  ROUT               ;Output character
2D28 DF6A       0240        SCAL CRLF               ;Carriage Return
                0250 ;Check for valid input.
2D2A 79         0260        LD   A,C                ;Get saved char.
2D2B FE33       0270        CP   33H                ;If > 33H,
2D2D 3006       0280        JR   NC,ERROR           ;then error.
2D2F E6F0       0290        AND  0F0H               ;If 3xH,
2D31 FE30       0300        CP   30H                ;then
2D33 2802       0310        JR   Z,CONT             ;continue,else
                0320 ;
2D35 DF6B       0330 ERROR  SCAL ERRM               ;error.
2D37 DF6A       0340 CONT   SCAL CRLF
2D39 DF5D       0350        SCAL TDEL               ;Long delay.
2D3B 18C3       0360        JR   START
```

4.2 PROGRAM 2

Program 2 illustrates the use of CPOS together with display control commands within a PRS string.

The program simply draws a rectangle on the display, positions the cursor in the middle of the rectangle, and then returns to NAS-SYS.

The lines are drawn using PRS with strings which include cursor control characters together with the characters to be used to form the line. The position of the cursor is saved on the stack at line 290, workspace location CURSOR holding the current position. The position saved is that of the top left-hand corner of the rectangle.

The top horizontal line is drawn at line 320 and the right-hand vertical at line 330. The cursor position is now at the bottom right-hand corner of the rectangle. In order to draw the bottom horizontal, the cursor must be positioned to the left-hand end of the line; this is done in lines 340-390. The current cursor position is loaded into register pair HL and a call made to CPOS. This routine returns with HL containing the address of the start of the line. This value is incremented (to bring the start of the line away from the edge of the display) and the value loaded into location CURSOR. The cursor position is now the bottom left-hand corner of the rectangle and the line may be drawn.

At line 430, the position of the top left-hand corner of the rectangle, which was saved in lines 290-300, is popped off the stack and loaded into CURSOR. The left-hand vertical line is then drawn.

Finally, at line 490, the cursor is positioned towards the middle of the screen so that when the return to NAS-SYS is made in line 510, the logo appears in the middle of the rectangle.

Listing of Program 2

```
                0010 ;PROGRAM 2
                0020 ;
2D00            0030        ORG  2D00H
                0040 ;
2D00 000C       0050 CLS    EQU  0CH        ;Clear screen code.
2D00 0014       0060 CUD    EQU  14H        ;Cursor down code.
2D00 0011       0070 CUL    EQU  11H        ;Cursor left code.
2D00 0012       0080 CUR    EQU  12H        ;Cursor right code.
                0090 ;
                0100 ;NAS-SYS subroutine numbers.
2D00 007C       0110 CPOS   EQU  7CH
2D00 0028       0120 PRS    EQU  28H
2D00 005B       0130 MRET   EQU  5BH
                0140 ;
                0150 ;NAS-SYS memory location.
2D00 0C29       0160 CURSOR EQU  0C29H
                0170 ;
                0180 ;
                0190 ;
```

Listing of Program 2, continued.

```
                    0200 ;
                    0210 ;
                    0220 ;
                    0230 ;
                    0240 ;Clear screen,move cursor down & right.
2D00 EF             0250        RST  PRS
2D01 0C141414       0260        DEFB CLS,CUD,CUD,CUD,CUR,0
     1200
                    0270 ;
                    0280 ;Save current cursor position on stack.
2D07 2A290C         0290        LD   HL,(CURSOR)
2D0A E5             0300        PUSH HL
                    0310 ;
2D0B CD392D         0320        CALL HLINE      ;Top line.
2D0E CD2C2D         0330        CALL VLINE      ;Right vertical.
                    0340 ;Get the start of current line,
                    0350 ;  increment it, & move cursor there.
2D11 2A290C         0360        LD   HL,(CURSOR)
2D14 DF7C           0370        SCAL CPOS
2D16 23             0380        INC  HL
2D17 22290C         0390        LD   (CURSOR),HL
                    0400 ;
2D1A CD392D         0410        CALL HLINE      ;Bottom line.
                    0420 ;Get saved cursor position from stack.
2D1D E1             0430        POP  HL
2D1E 22290C         0440        LD   (CURSOR),HL
                    0450 ;
2D21 CD2C2D         0460        CALL VLINE      ;Left vertical.
                    0470 ;Move cursor to middle of screen,
                    0480 ;  so that logo will appear there.
2D24 21180A         0490        LD   HL,0A18H
2D27 22290C         0500        LD   (CURSOR),HL
2D2A DF5B           0510        SCAL MRET
                    0520 ;
                    0530 ;
                    0540 ;
                    0550 ;Draw a vertical line from cursor position
2D2C 0605           0560 VLINE  LD   B,5
2D2E EF             0570 VL1    RST  PRS
2D2F 7F11142A       0580        DEFB 7FH,CUL,CUD,2AH,CUL,CUD,0
     111400
2D36 10F6           0590        DJNZ VL1
2D38 C9             0600        RET
                    0610 ;
                    0620 ;Draw a horizontal line from cursor posn.
2D39 0617           0630 HLINE  LD   B,17H
2D3B EF             0640 H1     RST  PRS
2D3C 7F2A00         0650        DEFB 7FH,2AH,0
2D3F 10FA           0660        DJNZ H1
2D41 C9             0670        RET
```

4.3 PROGRAM 3

Program 3 illustrates the use of those routines which output register contents, ie.B1HEX, B2HEX, TBCD2, TBCD3, and TX1. See Chapter 3 for a full description of each routine.

The output generated by each routine is displayed followed by a dot and newline produced by this program. Routines TBCD2, TBCD3, and TX1 generate checksums in register C and these are also displayed using B2HEX.

TBCD2 outputs 01 and adds this to register C which thus contains 01H. TBCD3 outputs 23H,45H and adds these to register C which therefore contains 01 + 23 + 45 = 69H. TX1 outputs 23H,45H,67H,89H so the checksum becomes 69 + 23 + 45 + 67 + 89 =1C1H, the overflow 1 being lost.

When run the program outputs:

```
1.
01.
01.
01.
2345 .
69.
2345 6789 .
C1.
** NAS-SYS 3 **
```

Listing of Program 3

```
                    0010 ;PROGRAM 3
                    0020 ;
2D00                0030        ORG  2D00H
                    0040 ;NAS-SYS subroutine numbers.
                    0050 ;
2D00 007A           0060 B1HEX  EQU  7AH
2D00 0068           0070 B2HEX  EQU  68H
2D00 0028           0080 PRS    EQU  28H
2D00 0067           0090 TBCD2  EQU  67H
2D00 0066           0100 TBCD3  EQU  66H
2D00 006C           0110 TX1    EQU  6CH
2D00 005B           0120 MRET   EQU  5BH
                    0130 ;
                    0140 ;
2D00 EF             0150        RST  PRS
2D01 0C00           0160        DEFB 0CH,0     ;Clear screen.
                    0170 ;
2D03 3E01           0180        LD   A,1
2D05 DF7A           0190        SCAL B1HEX     ;Display low nibble of A.
2D07 CD382D         0200        CALL DOT
                    0210 ;
2D0A 3E01           0220        LD   A,1
2D0C DF68           0230        SCAL B2HEX     ;Display A.
2D0E CD382D         0240        CALL DOT
                    0250 ;
2D11 3E01           0260        LD   A,1
2D13 0E00           0270        LD   C,0
2D15 DF67           0280        SCAL TBCD2     ;Display A,checksum in C.
2D17 CD382D         0290        CALL DOT
2D1A 79             0300        LD   A,C
2D1B DF68           0310        SCAL B2HEX     ;Display checksum from C.
2D1D CD382D         0320        CALL DOT
                    0330 ;
2D20 214523         0340        LD   HL,2345H
2D23 DF66           0350        SCAL TBCD3     ;Display HL,checksum in C.
2D25 79             0360        LD   A,C
2D26 DF68           0370        SCAL B2HEX     ;Display checksum from C.
2D28 CD382D         0380        CALL DOT
                    0390 ;
2D2B 118967         0400        LD   DE,6789H
2D2E DF6C           0410        SCAL TX1       ;Display HL DE,checksum inC.
2D30 79             0420        LD   A,C
2D31 DF68           0430        SCAL B2HEX     ;Display checksum from C.
2D33 CD382D         0440        CALL DOT
                    0450 ;
2D36 DF5B           0460        SCAL MRET
                    0470 ;
                    0480 ;Subroutine to display a dot,
                    0490 ;   followed by a new line.
2D38 EF             0500 DOT    RST  PRS
2D39 2E0D00         0510        DEFB 2EH,0DH,0
2D3C C9             0520        RET
```

4.4 PROGRAM 4

This program shows how arguments may be entered when executing a program.
The program displays a count on the screen; the count may be up or down as determined by the second argument in the command line (0=down count, non-zero = up count), and the step size is determined by the third argument.

The program begins by moving the arguments into register pairs HL, DE, and BC. The up/down argument in DE is examined to see if it is zero or non-zero so determining the direction of the count. The step size in BC (only C is relevant) is negated if a down count is required.
The remainder of the program produces the required count using B2HEX to output the count, TDEL to produce a delay between counts, and CRLF to output each count on a new line.

Sample run:

```
E2D00 1 4
00
04
08
0C
10
14
18
1C
20
```

Listing of Program 4

```
                0010 ;PROGRAM 4
                0020 ;Up/down counter.
                0030 ;Execute as E2D00 <up/down> <step>
                0040 ;
2D00            0050        ORG  2D00H
                0060 ;
                0070 ;NAS-SYS subroutine numbers.
2D00 0060       0080 ARGS   EQU  60H
2D00 006A       0090 CRLF   EQU  6AH
2D00 0068       0100 B2HEX  EQU  68H
2D00 005D       0110 TDEL   EQU  5DH
                0120 ;
                0130 ;
                0140 ;
                0150 ;
2D00 DF60       0160        SCAL ARGS     ;Move arguments into regs.
2D02 7B         0170        LD   A,E      ;If HL
2D03 B2         0180        OR   D        ;is not zero,
2D04 2004       0190        JR   NZ,GO    ;then upcount,so jump,
2D06 79         0200        LD   A,C      ;else
2D07 ED44       0210        NEG  ;negate
2D09 4F         0220        LD   C,A      ;the step size.
                0230 ;
2D0A AF         0240 GO     XOR  A        ;Start count at zero.
2D0B 47         0250        LD   B,A      ;B holds current count.
2D0C D9         0260 COUNT  EXX  ;Save active regs.
2D0D DF68       0270        SCAL B2HEX     ;Display count from A.
2D0F DF5D       0280        SCAL TDEL      ;Delay.
2D11 DF6A       0290        SCAL CRLF      ;Newline.
2D13 D9         0300        EXX  ;Restore active regs.
2D14 78         0310        LD   A,B      ;Add
2D15 81         0320        ADD  A,C      ;stepsize from C
2D16 47         0330        LD   B,A      ;and put back in B.
2D17 18F3       0340        JR   COUNT
```

4.5 PROGRAM 5

This example shows the use of INLIN and RLIN to extract arguments from an input line. B2HEX, TBCD3, and TX1 are used to display the data extracted from the input line.

The program begins in line 240 by calling INLIN to blink the cursor and echo the keyboard input to the video display until Enter is keyed. RLIN then extracts the data from the input line and transfers it to workspace locations ARG1, ARG2, etc, the number of arguments being in ARGN. Lines 260 to 590 display these locations. At line 610, ARGS is used to transfer the first three arguments into register pairs HL, DE, and BC. These are also displayed.

Some sample runs:

```
E2D00
1234 5678 9ABC DEF0              ( User input.)
(ARGN)=04
(ARG1)=1234
(ARG2)=5678
(ARG3)=9ABC
(ARG4)=DEF0
1234 5678 9ABC

1 2 3 4 5 6 7 8 9 0              ( User input.)
(ARGN)=0A
(ARG1)=0001
(ARG2)=0002
(ARG3)=0003
(ARG4)=0004
00001 0002 0003


                                  ( User input is blank line.)
(ARGN)=00
(ARG1)=0001
(ARG2)=0002
(ARG3)=0003
(ARG4)=0004
0001 0002 0003
```

Listing of Program 5, ~~continued~~

```
                    0010 ;PROGRAM 5
                    0020 ;
2D00                0030        ORG  2D00H
                    0040 ;
                    0050 ;NAS-SYS subroutine numbers.
2D00 0060           0060 ARGS   EQU  60H
2D00 0068           0070 B2HEX  EQU  68H
2D00 006A           0080 CRLF   EQU  6AH
2D00 0063           0090 INLIN  EQU  63H
2D00 0028           0100 PRS    EQU  28H
2D00 0079           0110 RLIN   EQU  79H
2D00 0066           0120 TBCD3  EQU  66H
2D00 006C           0130 TX1    EQU  6CH
                    0140 ;
                    0150 ;NAS-SYS memory locations.
2D00 0C0B           0160 ARGN   EQU  0C0BH
2D00 0C0C           0170 ARG1   EQU  0C0CH
2D00 0C0E           0180 ARG2   EQU  0C0EH
2D00 0C10           0190 ARG3   EQU  0C10H
2D00 0C12           0200 ARG4   EQU  0C12H
2D00 0C14           0210 ARG5   EQU  0C14H
                    0220 ;
                    0230 ;
2D00 DF63           0240 START  SCAL INLIN     ;Get an input line.
2D02 DF79           0250        SCAL RLIN      ;Get arguments from line.
2D04 EF             0260        RST  PRS
2D05 28415247       0270        DEFM /(ARGN)=/
     4E293D
2D0C 00             0280        DEFB 0
2D0D 3A0B0C         0290        LD   A,(ARGN)    ;Number of arguments in
2D10 DF68           0300        SCAL B2HEX     ;line is displayed.
2D12 DF6A           0310        SCAL CRLF
                    0320 ;
2D14 EF             0330        RST  PRS
2D15 28415247       0340        DEFM /(ARG1)=/
     31293D
2D1C 00             0350        DEFB 0
2D1D 2A0C0C         0360        LD   HL,(ARG1) ;Content of memory
2D20 DF66           0370        SCAL TBCD3    ;location ARG1 displayed.
2D22 DF6A           0380        SCAL CRLF
                    0390 ;
2D24 EF             0400        RST  PRS
2D25 28415247       0410        DEFM /(ARG2)=/
     32293D
2D2C 00             0420        DEFB 0
2D2D 2A0E0C         0430        LD   HL,(ARG2) ;Content of memory location
2D30 DF66           0440        SCAL TBCD3    ;ARG2 is displayed.
2D32 DF6A           0450        SCAL CRLF
                    0460 ;
```

Listing of Prosram 5, continued

```
2D34 EF          0470          RST  PRS
2D35 28415247    0480          DEFM /(ARG3)=/
     33293D
2D3C 00          0490          DEFB 0
2D3D 2A100C      0500          LD   HL,(ARG3)  ;Content of memory
2D40 DF66        0510          SCAL TBCD3     ;location ARG3 displayed.
2D42 DF6A        0520          SCAL CRLF
                 0530 ;
2D44 EF          0540          RST  PRS
2D45 28415247    0550          DEFM /(ARG4)=/
     34293D
2D4C 00          0560          DEFB 0
2D4D 2A120C      0570          LD   HL,(ARG4)  ;Content of memory location
2D50 DF66        0580          SCAL TBCD3     ;ARG4 is displayed.
2D52 DF6A        0590          SCAL CRLF
                 0600 ;
2D54 DF60        0610          SCAL ARGS      ;First 3 args to HL,DE,BC.
2D56 C5          0620          PUSH BC
2D57 DF6C        0630          SCAL TX1       ;Display HL DE.
2D59 E1          0640          POP  HL
2D5A DF66        0650          SCAL TBCD3     ;Display content of BC.
                 0660 ;
2D5C DF6A        0670          SCAL CRLF
2D5E 18A0        0680          JR   START
```

4.6 PROGRAM 6

This example is actually two short programs. The first, which is lines 110 to 150, and which is executed from location 2D00, shows the use of FFLP. Register A is loaded with the bit pattern 00100100 and consequently bits 2 and 5 of output port 0 are pulsed. These are the only bits of the port which are not dedicated to driving the keyboard hardware.)
The signals produced at PL3 pins 6 and 8 with the Nascom clock at 4 MHz are both:

The second program, which is lines 230 to 250, simply turns led2 on and off using MFLP. TDEL is used to reduce the frequency of the on/off cycle so that it can be visually observed. It is executed from location 2D10.

Listing of Program 6

```
                0010 ;PROGRAM 6
                0020 ;
2D00            0030        ORG   2D00H
                0040 ;
                0050 ;
                0060 ;NAS-SYS routine numbers
2D00 005E       0070 FFLP   EQU   5EH
2D00 005F       0080 MFLP   EQU   5FH
2D00 005D       0090 TDEL   EQU   5DH
                0100 ;
                0110 ;Pulse bits 2 and 5 of PORT 0.
2D00 0624       0120        LD    B,24H       ;Bits 2 and 5 to pulse.
2D02 78         0130 LOOP1  LD    A,B
2D03 DF5E       0140        SCAL  FFLP
2D05 18FB       0150        JR    LOOP1
                0160 ;
                0170 ;
                0180 ;
2D10            0190        ORG   2D10H
                0200 ;
                0210 ;Toggle the cassette led.
                0220 ;
2D10 DF5F       0230 LOOP2  SCAL  MFLP
2D12 DF5D       0240        SCAL  TDEL
2D14 18FA       0250        JR    LOOP2
```

4.7 PROGRAM 7

This program prompts the user to enter a hexadecimal number of upto four digits, and then uses INLIN and NUM to transfer the number to workspace location NUMV (0C21/2) and the number of digits in the number to location NUMN (0C20).
The input number is read at line 240 using INLIN. The number, which is stored as ASCII codes in video ram locations pointed to by register pair DE, is converted to hexadecimal in workspace location NUMV using NUM, the number of digits being stored in NUMN. The remainder of the program simply displays the contents of NUMV and NUMN.

Listing of Program 7

```
                0010 ;PROGRAM 7
2D00            0020        ORG  2D00H
                0030 ;
                0040 ;
                0050 ;NAS-SYS routine numbers.
2D00 0068       0060 B2HEX  EQU  68H
2D00 0063       0070 INLIN  EQU  63H
2D00 0064       0080 NUM    EQU  64H
2D00 0066       0090 TBCD3  EQU  66H
2D00 0028       0100 PRS    EQU  28H
2D00 0069       0110 SPACE  EQU  69H
                0120 ;
                0130 ;NAS-SYS workspace locations.
2D00 0C21       0140 NUMV   EQU  0C21H
2D00 0C20       0150 NUMN   EQU  0C20H
                0160 ;
                0170 ;
2D00 EF         0180 START  RST  PRS
2D01 456E7465   0190        DEFM /Enter a hex number,up to 4 digits/
     72206120
     68657820
     6E756D62
     65722C75
     7020746F
     20342064
     69676974
     73
2D22 0D00       0200        DEFB 0DH,0
                0210 ;
                0220 ;
                0230 ;Get an input line.
2D24 DF63       0240        SCAL INLIN
                0250 ;
                0260 ;Convert input from ASCII to hex,
                0270 ; result in NUMV.
2D26 DF64       0280        SCAL NUM
                0290 ;
```

Listing of Program 7, continued

```
2D28 EF          0300          RST  PRS
2D29 456E7465    0310          DEFM /Entered number is stored at /
     72656420
     6E756D62
     65722069
     73207374
     6F726564
     20617420
2D45 0D          0320          DEFB 0DH
2D46 6C6F6361    0330          DEFM /location NUMV as /
     74696F6E
     204E554D
     56206173
     20
2D57 00          0340          DEFB 0
                 0350 ;
                 0360 ;Now display.
2D58 2A210C      0370          LD   HL,(NUMV)     ;Move result to HL,
2D5B DF66        0380          SCAL TBCD3         ; and display it.
2D5D DF69        0390          SCAL SPACE         ;One space.
2D5F 3A200C      0400          LD   A,(NUMN)      ;Number of digits to A.
2D62 DF68        0410          SCAL B2HEX         ; and disply.
                 0420 ;
2D64 EF          0430          RST  PRS
2D65 0D0D0D00    0440          DEFB 0DH,0DH,0DH,0
2D69 1895        0450          JR   START
```

4.8 PROGRAM 8

This program shows how SCALJ is used to call another routine. The A command is to be called, so the required two arguments are loaded into register pairs HL and DE. The number of the A command, 41H, is transferred to ARGC at lines 150 and 160 and then the A command is called using SCALJ at line 170.
The called routine may be any of the commands or routines having numbers from 41H to 7FH.

When executed, the program outputs:

```
0008 FFFC FA
* NAS-SYS 3 *
```

Listing of Program 8

```
                0010 ;PROGRAM 8
                0020 ;
2D00            0030        ORG  2D00H
                0040 ;
                0050 ;NAS-SYS routine numbers.
2D00 005B       0060 MRET   EQU  5BH
2D00 005C       0070 SCALJ  EQU  5CH
                0080 ;
                0090 ;NAS-SYS workspace
2D00 0C0A       0100 ARGC   EQU  0C0AH
                0110 ;
                0120 ;
2D00 210600     0130        LD   HL,6          ;Arguments
2D03 110200     0140        LD   DE,2          ; for
2D06 3E41       0150        LD   A,"A          ;  Arithmetic
2D08 320A0C     0160        LD   (ARGC),A      ;   command,
2D0B DF5C       0170        SCAL SCALJ       ;called via SCALJ.
                0180 ;
2D0D DF5B       0190        SCAL MRET
```

4.2.9 PROGRAM 9

This program allows a screenful of text to be created and edited on the screen and then saved on cassette tape.
The program begins by clearing the screen, lines 150-160, using CRT to display the 'clear screen' character,0CH. (A better alternative here is to program RST PRS:DEFB 0CH,0. This requires only three bytes instead of four.) Next, at lines 180-190, the input table is set so that input is from the keyboard only. This avoids any spurious inputs from the cassette recorder when rewinding, etc.

At TEXT,line 210, the cursor is blinked until a keyboard character is input. Either alphanumeric characters or cursor control keys may be entered tp produce the required display. If the input character is a NUL (CNTRL/SHFT/@), the screen editing is terminated and the program jumps to TAPE. Otherwise, the input character is displayed and the process repeated.

Saving the screen text on cassette tape, TAPE at line 280, begins by transferring the video ram to memory locations A000 to A400. It is these locations which will actually be written to the tape. This is necessary because during the write to tape information indicating the progress of the Write command is displayed and this information corrupts the desired screenful of text.

At line 330, a W A000 A400 command is programmed. Workspace location ARG1 is loaded with A000, location ARG2 is loaded with A400 and then the W command is called as SCAL "W. At this time the cassette recorder should be set to record. The program ends by returning control to NAS-SYS.

The tape so saved may, at some later time, be read back using the R command and the required text again displayed by copying from A000 to 0800; the command string is thus R

C A000 0800 0400

Note that if the cassette input of a second Nascom doing an R command were to be connected to the Nascom running this program, the second Nascom could receive the screenful of text. Since the cassette output is a sequence of audio tones, any audio link, such as the telephone system or radio, may be used. This program could therefore be used as a simple method of communicating between Nascoms. Of course, the legal restrictions applying to the communication medium will have to be observed.

Listing of Program 9

```
                0010 ;PROGRAM 9
                0020 ;
2D00            0030        ORG  2D00H
                0040 ;
                0050 ;NAS-SYS routine numbers.
2D00 007B       0060 BLINK  EQU  7BH
2D00 0065       0070 CRT    EQU  65H
2D00 0072       0080 NIM    EQU  72H
2D00 005B       0090 MRET   EQU  5BH
                0100 ;NAS-SYS workspace
2D00 0C0C       0110 ARG1   EQU  0C0CH
2D00 0C0E       0120 ARG2   EQU  0C0EH
2D00 0780       0130 INTAB  EQU  0780H ;For NAS-SYS 1,0786H.
                0140 ;
2D00 3E0C       0150        LD   A,0CH     ;Clear the
2D02 DF65       0160        SCAL CRT     ;  screen.
                0170 ;
2D04 218007     0180        LD   HL,INTAB   ;Input table is
2D07 DF72       0190        SCAL NIM      ;  keyboard only.
                0200 ;Enter and edit text.
2D09 DF7B       0210 TEXT   SCAL BLINK    ;Get a key
2D0B A7         0220        AND  A              ;Is it CTRL/SHFT/@
2D0C 2804       0230        JR   Z,TAPE          ;If so,jump,else..
2D0E DF65       0240        SCAL CRT           ;  display character.
2D10 18F7       0250        JR   TEXT
                0260 ;Transfer to tape.
                0270 ;First copy video ram to A000-A400.
2D12 210008     0280 TAPE   LD   HL,0800H
2D15 1100A0     0290        LD   DE,0A000H
2D18 010004     0300        LD   BC,0400H
2D1B EDB0       0310        LDIR
                0320 ;Do a W A000 A400
2D1D 2100A0     0330        LD   HL,0A000H
2D20 220C0C     0340        LD   (ARG1),HL
2D23 2100A4     0350        LD   HL,0A400H
2D26 220E0C     0360        LD   (ARG2),HL
2D29 DF57       0370        SCAL "W
                0380 ;
2D2B DF5B       0390        SCAL MRET
```

4.10 PROGRAM 10

This program is a development of program 9. The tape of screen data when replayed will be automatically displayed on the screen. This is achieved by writing onto the tape itself the commands to read the tape into the screen buffer and copy from the screen buffer to the video ram. (This is similar to the technique used by the Generate command.)

The first part of the program, lines 200 to 300, is the text editing process, as in Program 9. When the data is to be saved, the video ram contents are copied into the buffer area which starts at location BUF. At line 390, the output table is defined to avoid output to any devices other than the screen and tape. Next, the sequence (CR)

E0 (CR)

R (CR)

is written to the tape by lines 430 to 540. A delay gives time for the Nascom to respond following receipt of these commands from the tape. The text data is then written using a call to the W command, lines 570 to 610. Following another short delay the command string for a Copy command is written to the tape. The original output table is restored in lines 760 to 770.

Listing of Program 10

```
                    0010 ;PROGRAM 10
                    0020 ;
2D00                0030         ORG  2D00H
                    0040 ;NAS-SYS routine numbers.
2D00 007B           0050 BLINK   EQU  7BH
2D00 0065           0060 CRT     EQU  65H
2D00 0072           0070 NIM     EQU  72H
2D00 0071           0080 NOM     EQU  71H
2D00 005B           0090 MRET    EQU  5BH
2D00 0066           0100 TBCD3   EQU  66H
2D00 0028           0110 PRS     EQU  28H
2D00 0038           0120 RDEL    EQU  38H
2D00 0030           0130 ROUT    EQU  30H
                    0140 ;NAS-SYS workspace
2D00 0C0C           0150 ARG1    EQU  0C0CH
2D00 0C0E           0160 ARG2    EQU  0C0EH
2D00 0780           0170 INTAB   EQU  0780H ;For NAS-SYS 1,0786H.
2D00 0774           0180 OUTT2   EQU  0774H
                    0190 ;
2D00 3E0C           0200 START   LD   A,0CH ;Clear the
2D02 DF65           0210         SCAL CRT     ;  screen.
                    0220 ;
2D04 218007         0230         LD   HL,INTAB   ;Input table is
2D07 DF72           0240         SCAL NIM      ;  keyboard only.
                    0250 ;Enter and edit text.
2D09 DF7B           0260 TEXT    SCAL BLINK    ;Get a key
2D0B A7             0270         AND  A             ;Is it CTRL/SHFT/@
2D0C 2804           0280         JR   Z,TAPE         ;If so,jump,else..
2D0E DF65           0290         SCAL CRT          ;  display character.
2D10 18F7           0300         JR   TEXT
```

Listing of Program #10, continued

```
                    0310 ;Transfer to tape.
                    0320 ;First copy video ram BUF.
2D12 210008         0330 TAPE   LD   HL,0800H
2D15 11642D         0340        LD   DE,BUF
2D18 010004         0350        LD   BC,0400H
2D1B EDB0           0360        LDIR
                    0370 ;
                    0380 ;Set output table to CRT & SRLX.
2D1D 217407         0390        LD   HL,OUTT2
2D20 DF71           0400        SCAL NOM
2D22 E5             0410        PUSH HL      ;Save previous table.
                    0420 ;Output the command header.
2D23 215E2D         0430        LD   HL,ECS
2D26 0606           0440        LD   B,6
2D28 7E             0450 OC     LD   A,(HL)
2D29 F7             0460        RST  ROUT
                    0470 ;Wait
2D2A 0E20           0480        LD   C,20H
2D2C AF             0490        XOR  A
2D2D FF             0500 W1     RST  RDEL
2D2E 0D             0510        DEC  C
2D2F 20FC           0520        JR   NZ,W1
2D31 23             0530        INC  HL
2D32 10F4           0540        DJNZ OC
                    0550 ;Output text data.
                    0560 ;Do a W buf buf+400
2D34 21642D         0570        LD   HL,BUF
2D37 220C0C         0580        LD   (ARG1),HL
2D3A 216431         0590        LD   HL,BUF+400H
2D3D 220E0C         0600        LD   (ARG2),HL
2D40 DF57           0610        SCAL "W
                    0620 ;
                    0630 ;Wait
2D42 AF             0640        XOR  A
2D43 FF             0650        RST  RDEL
                    0660 ;
                    0670 ;Output C buffer address 0800 0400.
2D44 3E43           0680        LD   A,"C
2D46 F7             0690        RST  ROUT
2D47 21642D         0700        LD   HL,BUF
2D4A DF66           0710        SCAL TBCD3
2D4C EF             0720        RST  PRS
2D4D 30383030       0730        DEFM /0800 0400/
     20303430
     30
2D56 0D00           0740        DEFB 0DH,0
                    0750 ;
2D58 E1             0760        POP  HL
2D59 DF71           0770        SCAL NOM
                    0780 ;
2D5B C3002D         0790        JP   START
2D5E 0D45300D       0800 ECS    DEFB 0DH,"E,"0,0DH,"R,0DH
     520D
2D64 0400           0810 BUF    DEFS 400H
```

4.11 PROGRAM 11

This program simply inputs characters from the keyboard and outputs them to the video display and the serial output port. The repeat keyboard facility is not used.
Lines 100 and 110 form a loop which is terminated only when an input character from the keyboard is detected. This character, in register A, is displayed and transmitted from the serial port by SRLX and the process repeats.

When the tape is replayed after resetting the Nascom, it will have the same effect as a phantom pressing the keyboard.

Listing of Program 11

```
                0010 ;PROGRAM 11
                0020 ;
2D00            0030        ORG  2D00H
                0040 ;
                0050 ;NAS-SYS routine numbers:
2D00 0065       0060 CRT    EQU  65H
2D00 0061       0070 KBD    EQU  61H
2D00 006F       0080 SRLX   EQU  6FH
                0090 ;
2D00 DF61       0100 LOOP   SCAL KBD       ;Wait for a
2D02 30FC       0110        JR   NC,LOOP       ; keyboard input.
2D04 DF6F       0120        SCAL SRLX      ;Write to serial port.
2D06 18F8       0130        JR   LOOP      ;Do forever.
```

4.12 PROGRAM 12

Program 12 waits for a line of text to entered from any of the devices in the input table, normally the keyboard. Following an Enter, the tape led is turned on and ,after a delay, the content of the line on the screen is transmitted to the serial port. The tape led is then turned off and another input line awaited.

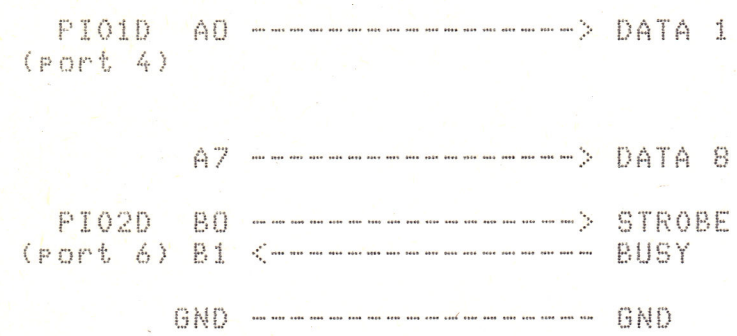
Listing of Program 12

```
                0010 ;PROGRAM 12
                0020 ;
2D00            0030        ORG  2D00H
                0040 ;
                0050 ;NAS-SYS routine numbers:
2D00 0063       0060 INLIN  EQU  63H
2D00 005F       0070 MFLP   EQU  5FH
2D00 006D       0080 SOUT   EQU  6DH
2D00 005D       0090 TDEL   EQU  5DH
                0100 ;
                0110 ;Get an input line:
2D00 DF63       0120 START  SCAL INLIN
2D02 D5         0130        PUSH DE      ;Save address of line start.
                0140 ;Turn tape led on:
2D03 DF5F       0150        SCAL MFLP
                0160 ;Wait:
2D05 DF5D       0170        SCAL TDEL
                0180 ;Output the line to the serial port:
2D07 E1         0190        POP  HL      ;Start of data.
2D08 0630       0200        LD   B,30H      ;Line length is 48 chars.
2D0A DF6D       0210        SCAL SOUT
                0220 ;Turn tape led off.
2D0C DF5F       0230        SCAL MFLP
                0240 ;Do forever:
2D0E 18F0       0250        JR   START
```

4.13 PROGRAM 13

This program is a user-written output routine to drive a parallel printer. Such printers usually have seven or eight data lines which carry the character code to the printer, and two 'handshake' lines. One of these lines, STROBE, is pulsed low by the computer to transfer the character code into the printer register. While the printer is busy printing, it generates a level on its BUSY line to indicate to the computer that a further character code should not yet be transmitted.

The connections between the PIO and the printer are assumed to be:



The program begins by initialising the two PIO ports to make PIO1 an 8 bit output port to carry the character code, and PIO2 a mode 3 port with bit 0 output and bits 1 to 7 input.
The STROBE signal is then set high, and the address of the PRINT subroutine loaded into workspace locations $U0UT+1/2.The initialisation section then returns to NAS-SYS.

The PRINT subroutine begins by waiting for the printer's BUSY signal to indicate that it is not busy. (It is assumed here that the printer indicates that it is busy by a logic 1 on its BUSY line; if the printer busy level is actually a logic 0, then the JR NZ,WAIT instruction must be replaced with a JR Z,WAIT instruction.)
If the output character is a carriage return it is replaced by the code for a linefeed. (If the printer automatically line feeds on receipt of a carriage return, lines 430 -450 inclusive must be removed.) The STROBE line is then brought low and then high again to strobe the character code into the printer.

To activate this parallel printer routine, execute it from 0C80, ie command EC80; then, a U command will bring the printer routine into the table of output device. An N command removes the printer routine from the table. See U command, Chapter 2.

Listing of Program 13

```
                    0010 ;PROGRAM 13
                    0020 ;
                    0030 ;User routine to drive a parallel printer.
                    0040 ;
0C80                0050        ORG  0C80H
                    0060 ;
                    0070 ;NAS-SYS routine numbers:
0C80 005B           0080 MRET   EQU  5BH
                    0090 ;
                    0100 ;NAS-SYS workspace:
0C80 0C77           0110 $UOUT  EQU  0C77H
                    0120 ;
                    0130 ;PIO definitions:
0C80 0004           0140 PAD    EQU  4          ;Data, 8 bits.
0C80 0006           0150 PAC    EQU  PAD+2      ;Control register.
0C80 0005           0160 PBD    EQU  5          ;Printer status.
0C80 0007           0170 PBC    EQU  PBD+2      ;Control register.
                    0180 ;Initialise PIOs:
0C80 3E0F           0190        LD   A,0FH       ;PA,
0C82 D306           0200        OUT  (PAC),A    ;  output mode.
0C84 3ECF           0210        LD   A,0CFH      ;PB,
0C86 D307           0220        OUT  (PBC),A    ;  mode 3...
0C88 3EFE           0230        LD   A,0FEH      ;Bit 0 out,
0C8A D307           0240        OUT  (PBC),A    ;  others in.
                    0250 ;
                    0260 ;Set STROBE high:
0C8C 3E01           0270        LD   A,1
0C8E D305           0280        OUT  (PBD),A
                    0290 ;Initialise $UOUT:
0C90 21980C         0300        LD   HL,PRINT
0C93 22780C         0310        LD   ($UOUT+1),HL
                    0320 ;
0C96 DF5B           0330        SCAL MRET
                    0340 ;
                    0350 ;
0C98 F5             0360 PRINT  PUSH AF     ;Save A.
0C99 DB05           0370 WAIT   IN   A,(PBD)  ;Wait for printer
0C9B CB4F           0380        BIT  1,A        ;  not
0C9D 20FA           0390        JR   NZ,WAIT     ;   busy.
0C9F F1             0400        POP  AF         ;Restore A.
0CA0 F5             0410        PUSH AF
0CA1 FE0D           0420        CP   0DH         ;Is char a CR?
0CA3 2002           0430        JR   NZ,CHAR     ;If not, then jump,
0CA5 3E0A           0440        LD   A,0AH       ;  else make it LF.
0CA7 D304           0450 CHAR   OUT  (PAD),A  ;Output data.
0CA9 AF             0460        XOR  A          ;Pull STROBE
0CAA D305           0470        OUT  (PBD),A    ;  low.
0CAC 00             0480        NOP
0CAD 3C             0490        INC  A          ;STROBE high.
0CAE D305           0500        OUT  (PBD),A
0CB0 F1             0510        POP  AF
0CB1 C9             0520        RET
```

4.14 PROGRAM 14

This program illustrates the use of the Relative CALL routine, RCAL. It simply prompts the user to input a single decimal digit and then displays that digit multiplied by three.
Observe that line 240, RCAL X3, produces machine code D7 04 which is the code for RST 10H : DEFB 04 where 04 is the displacement from the RCAL instruction to the required subroutine, ie 4025 - 401F -2 = 04.

The assembled code occupies locations 4000H to 402CH and is executed from 4000H using the NAS-SYS command E4000. However, the point of using RCAL is that it allows position independent code to be written when used in conjunction with relative jumps, JR, as at line 270. Thus if the machine code is copied to another area of ram it will still operate. Try copying the code to location D00 onwards using NAS-SYS commands I4000 D00 2D followed by ED00.

Listing of Program 14

```
                    0010 ;PROGRAM 14
                    0020 ;
4000                0030        ORG  4000H
                    0040 ;
                    0050 ;NAS-SYS routine numbers
4000 0068           0060 B2HEX  EQU  68H
4000 007B           0070 BLINK  EQU  7BH
4000 006A           0080 CRLF   EQU  6AH
4000 0028           0090 PRS    EQU  28H
4000 0030           0100 ROUT   EQU  30H
4000 0069           0110 SPACE  EQU  69H
                    0120 ;
                    0130 ;Display prompt
4000 DF6A           0140 BEGIN  SCAL CRLF
4002 EF             0150        RST  PRS
4003 456E7465       0160        DEFM /Enter digit 0....9  /
     72206469
     67697420
     302E2E2E
     2E392020
4017 00             0170        DEFB 0
4018 DF7B           0180        SCAL BLINK          ;GET DIGIT
401A 47             0190        LD   B,A            ;SAVE IT
401B F7             0200        RST  ROUT           ;DISPLAY IT
401C DF69           0210        SCAL SPACE
401E 78             0220        LD   A,B              ;RECOVER DIGIT
                    0230 ;Now call routine to multiply
401F D704           0240        RCAL X3
                    0250 ;Display result
4021 DF68           0260        SCAL B2HEX
4023 18DB           0270        JR   BEGIN
                    0280 ;
```

Listing of Program 14, continued.

```
                0290 ;SUBROUTINE TO X 3.
                0300 ;ON ENTRY CHAR CODE IN A,
                0310 ;ON RETURN NUMBER IN A.
4025 D630       0320 X3     SUB  30H        ;CONVERT TO NUMBER
4027 47         0330        LD   B,A          ;X1
4028 87         0340        ADD  A,A         ;X2
4029 27         0350        DAA  ;CHANGE TO DECIMAL
402A 80         0360        ADD  A,B          ;X3
402B 27         0370        DAA  ;CHANGE TO DECIMAL
402C C9         0380        RET
```

5.1 The X=USR(N) Statement

5.1 Adding simple USR routines

The USR(N) statement in BASIC is effectively a CALL to the routine whose starting address is given in BASIC workspace location USRLOC (1004/5H = 4100/1d). Thus,

```
      DOKE 4100, 3200    :REM Load USRLOC with 3200d = 0C80H
          .
          .
      A=USR(0)           :REM Equivalent to CALL (USRLOC)
```

is effectively CALL 0C80H.

Example 1.
The USR routine is to change the state of the tape drive led. The code to do this is to start at 0C80H (=3200d).
First the assembly language code to toggle the led:

```
                        ORG 0C80H
                MFLP    EQU 5FH
0C80 DF 5F              SCAL MFLP       ;Toggle TAPE LED
0C82 C9                 RET             ;Return to BASIC
```

This code may be entered into ram using the NAS-SYS M command or an assembler.(But see 5.1.4)

Next the BASIC instructions:

```
10   DOKE 4100,3200          :REM Point to USR routine
20   X=USR(0)                :REM Toggle led
30   FOR I=1 TO 999          :REM Delay so we can see the changes
40   NEXT I
50   GOTO 20                 :REM Do forever
```

Enter and RUN this program and observe that the tape led toggles on and off, ie. each time the USR instruction is reached the code at address 0C80 is executed.

5.1.2 Passing a USR result back to BASIC

Example 1 requires no transmission of data between BASIC and the USR routine. Many useful USR routines require that a single result be passed back from the USR routine to a BASIC variable. Thus, X=USR(0) may be required to place the result into BASIC variable X. (The result, X, must be in the range -32768 to +32767). To achieve this, the USR routine must place its result in register pair AB (sic), and then CALL the routine whose address is stored in locations E00D/E. This routine transfers the result in register pair AB to BASIC variable X. The USR routine should then RETurn to BASIC.

The USR routine should therefore have the structure:

```
      Code to place result
      in register pair AB.
      CALL   (E00D)      ;Call  the  routine   whose address
                           is in E00D/E.
      RET                ;Return
```

But a CALL (nn) istruction does not exist in the Z80 instruction set! However, the structure may be:

```
      Code to place result
        in register pair AB.
      LD HL,(0E00DH)
      JP (HL)
```

This causes a jump (rather than a CALL) to the location whose address is in location E00D. Since the code at this location ends with a RET instruction, it is that instruction which returns the USR routine to BASIC. This is illustrated in the following example.

Example 2.
The USR routine is to scan the keyboard once and return either the ASCII code for the key pressed or 00 if no key is pressed. (This is the INKEY statement available in some BASIC dialects.) First the assembly code to scan the keyboard once and place the result in the BASIC variable.

Listing of INKEY

```
                    0010 ;INKEY FOR BASIC
                    0020 ;SCAN KEYBOARD ONCE
                    0030 ;RETURN ASCII CODE OR 00 AS X.
                    0040 ;
                    0050 ;DOKE 4100,3200 TO USE
                    0060 ;THEN X=USR(0).
                    0070 ;
0C80                0080        ORG  0C80H
0C80 0061           0090 KBD    EQU  61H
                    0100 ;
0C80 DF61           0110        SCAL KBD
0C82 3801           0120        JR   C,CHAR
0C84 AF             0130        XOR  A
0C85 47             0140 CHAR   LD   B,A
0C86 AF             0150        XOR  A
0C87 2A0DE0         0160        LD   HL,(0E00DH)
0C8A E9             0170        JP   (HL)
                     180
                     190
```

Scan kbd once.
If key pressed, jump
else 00.
Save character in
reg pair AB.
Jump to routine
to put result into variable X
and return to BASIC.

As before, this code may be entered as hexadecimal code using the NAS-SYS M command, by an assembler, or the program in 5.1.4.

Next the BASIC code:

```
10 REM TEST INKEY
20 DOKE 4100,3200
30 X=USR(0)        :REM INKEY
40 PRINT X;
50 GOTO 30
```

Enter and RUN the program and observe that the PRINTed numbers are the decimal values of the ASCII codes of the keys pressed.

Another useful command in some BASIC dialects is GET, which waits for a single key to be pressed on the keyboard and returns with the ASCII code of that key. This is given below.

Example 3.
The USR routine is to blink the cursor until an input is received from one of the devices in the input table. The routine is to return with the BASIC variable having the ASCII code for the received character.
First the assembly language code to get an input character:

```
                0010 ;GET FOR BASIC
                0020 ;SCAN INPUT DEVICES, BLINK CURSOR
                0030 ;UNTIL INPUT DETECTED, THEN RETURN
                0040 ;WITH ASCII CODE IN BASIC VARIABLE.
                0050 ;
                0060 ;TO USE, DOKE 4100,3200
                0070 ;
0C80            0080        ORG   0C80H
0C80 007B       0090 BLINK  EQU   7BH
                0100 ;
0C80 DF7B       0110        SCAL BLINK
                0120 ;MOVE INPUT INTO BASIC VARIABLE.
0C82 47         0130        LD    B,A
0C83 AF         0140        XOR   A
0C84 2A0DE0     0150        LD    HL,(0E00DH)
0C87 E9         0160        JP    (HL)
```

Next the BASIC instructions:

```
10 REM TEST GET
20 DOKE 4100,3200
30 X=USR(0)    :REM GET
40 PRINT X;
50 GOTO 30
```

Observe that X is given the decimal value of the ASCII code for the received character.

5.1.3 Transmitting an Argument to the USR Routine

The USR(A) instruction may transmit the variable A to the USR routine. (Whatever the value of A, it is truncated by BASIC to an

integer and, if it is not in the range -32768 to +32767, an FC error will be displayed.)
The USR routine extracts the value of the argument A by calling the subroutine whose address is in location E00B/C. This routine transfers the BASIC variable A to register pair DE.

Example 4.
The USR routine is to toggle the TAPE LED a number of times, the number being determined by input to the BASIC program.
First the assembly language code to toggle the led a number of times:

```
0C80            0010        ORG   0C80H
0C80 005F       0020 MFLP   EQU   5FH
0C80 005D       0030 TDEL   EQU   5DH
                0040 ;Get argument into DE
0C80 2A0BE0     0050        LD    HL,(0E00BH)    ;Effectively
0C83 11880C     0060        LD    DE,ARG         ;a CALL (E00B)
0C86 D5         0070        PUSH  DE             ;with return to
0C87 E9         0080        JP    (HL)           ;ARG, argument in
0C88 4B         0090 ARG    LD    C,E            ;reg pair DE.
0C89 DF5F       0100 T1     SCAL  MFLP           ;Toggle led.
0C8B DF5D       0110        SCAL  TDEL           ;Delay.
0C8D 0D         0120        DEC   C              ;Again
0C8E 20F9       0130        JR    NZ,T1
0C90 C9         0140        RET
```

Note that the number of times the led toggles is determined by the lower byte of the double byte argument transmitted from BASIC. The BASIC argument is thus assumed to be in the range 1 to 256.
Now the BASIC code:

```
10 REM Toggle led N times
20 DOKE 4100,3200
30 INPUT "N";N
40 IF (N<1)OR(N>256) THEN 30
50 X=USR(N)
60 GOTO 30
```

5.1.4 Loading USR Code from BASIC

The USR code once in memory may be stored on cassette tape using the NAS-SYS W command and reloaded using the R command before activating BASIC.
However, it is more convenient to load a BASIC program with a USR routine in the same manner as any other program. This can be achieved by making the BASIC program itself load the machine code for the USR routine: the program begins with instructions to load the USR routine using DOKEs from a DATA statement. The DATA statement must give successive pairs of the machine code code as two byte signed decimal integers.

For Example 1, machine code DF,5F,C9 must be loaded at consecutive locations from 0C80:

```
          write as  DF 5F
                    C9 00
```

then reverse each pair:

```
                    5F DF
                    00 C9
```

and convert each pair to decimal:

```
                    5FDF   =  24543
                    00C9   =    201.
```

These are the decimal values to be DOKEd.

The BASIC program thus becomes:

```
 10 FOR I=3200 TO 3202 STEP 2
 20 READ A
 30 DOKE I,A
 40 NEXT
 50 DOKE 4100,3200
 60 X=USR(0)
 70 FOR I=1 TO 999
 80 NEXT
 90 GOTO 60
100 DATA 24543,201
```

For Example 2, the machine code from location 0C80 is:

```
DF 61
38 01
AF 47
AF 2A
0D E0
E9 00
```

Reverse each pair and convert to decimal:

```
61DF   =   25055
0138   =     312
47AF   =   18351
2AAF   =   10927
E00D   =   57357 , -65536 = -8179
00E9   =     233
```

Note that if the decimal value is greater than 32767, then 65536 must be subtracted to bring the number into the range of 16 bit twos complement numbers.

The BASIC program is thus:

```
 10 FOR I=3200 TO 3210 STEP 2
 20 READ A
 30 DOKE I,A
 40 NEXT
```

```
50 DOKE 4100,3200
60 X=USR(0)
70 PRINT X;" ";
80 GOTO 60
90 DATA 25055,312,18351,10927,-8179,233
```

The conversion from hexadecimal to decimal is tedious and error prone. The following BASIC program automatically generates the first lines of a BASIC program which makes use of assembly code routines.

```
80 REM MCLOAD
90 S=3200:F=3230
100 CLS
110 PRINT"10 FOR I=";S;"TO";F;"STEP 2"
120 PRINT"20 READ A"
130 PRINT"30 DOKE I,A"
140 PRINT"40 NEXT"
150 PRINT"50 DOKE 4100,";S
160 L=60
170 B=S
180 PRINT L;" DATA ";
190 FOR A=B TO B+6 STEP 2
200 PRINT DEEK(A);",";
210 NEXT
220 PRINT CHR$(8)
230 IF B+6>=F THEN 300
240 B=B+8
250 L=L+10
260 GOTO 180
300 NEW
310 END
```

To use the program, first enter the machine code for the USR routine into ram. Convert the first and last addresses to decimal:these are S and F respectively in line 90. (F-S must be odd.) Then load and RUN the BASIC program MCLOAD. It will print the BASIC lines required to load the USR routine. Move the cursor to the beginning of the first line and press the Enter key repeatedly to enter each line into the users BASIC program. Finally, LIST the program and enter the remainder of the required users program. When reloaded from tape, the program will automatically load the required USR code.

As an example assume that the multiple USR(N) function of the following section is to be incorporated into a BASIC program and that the machine code for USR(N) has been loaded into locations 0C80H to 0CC1H, ie. 3200 to 3265d.
Enter BASIC and load MCLOAD with line 90 changed to:

```
90 S=3200: F=3265
```

Then RUN MCLOAD and observe that the following is displayed:

```
10 FOR I=3200 TO 3265
20 READ A
30 DOKE I,A
40 NEXT
```

```
50 DATA 2858,4576,3208,-5675
```

followed by other DATA statements.
(In this particular example there are so many DATA statements that line 10 is scrolled off the screen, so it must be replaced.)
Now move the cursor to the beginning of the first line on the screen. Press Enter repeatedly and then LIST. This is the part of the program which loads the multiple USR(N) function.
To test, add the following lines:

```
150 INPUT"I";I
160 X=USR(I)
170 PRINT"USR";I,X
180 PRINT
190 GOTO 150
```

and SAVE the program. When this program is subsequently LOADed, USR(1) is the BASIC GET command, USR(2) is the BASIC INKEY command, and other functions return an Error message.

5.1.5 Multiple Function USR

Often more than one USR function is required in a program, particularly if it is a games program which must manipulate the video display rapidly. The following program shows how up to eight functions may be incorporated.

In the instruction X=USR(N), N is used to select the required routine. At line 110 the argument N is extracted and placed in register pair DE. Lines 160-180 force N to be in the range 0 to 7. It is then multiplied by two and added to the base address of the jump table, JTAB. Line 230 causes a jump into JTAB at the required position.
The jump table contains the jumps to each of the eight possible routines. Only USR(1) and USR(2) are actually used, the other routines simply write 'Error' to the video display. To add a routine, say USR(3), replace the line U3 JR ERROR with the required code and reassemble.

Listing of general USR function.

```
                    0010 ;GENERAL USR(N) FUNCTION
                    0020 ;N IS 0 TO 7
                    0030 ;TO USE DOKE 4100,3200
                    0040 ;THEN X=USR(N).
                    0050 ;
0C80                0060        ORG   0C80H
0C80 007B           0070 BLINK  EQU   7BH
0C80 006B           0080 ERRM   EQU   6BH
0C80 0061           0090 KBD    EQU   61H
                    0100 ;
                    0110 ;GET ARGUMENT INTO DE.
0C80 2A0BE0         0120        LD    HL,(0E00BH)
0C83 11880C         0130        LD    DE,ARG
0C86 D5             0140        PUSH  DE
0C87 E9             0150        JP    (HL)
```

```
0C88 1600      0160 ARG    LD   D,0
0C8A 7B        0170        LD   A,E
0C8B E607      0180        AND  7
0C8D 87        0190        ADD  A,A
0C8E 5F        0200        LD   E,A
0C8F 21940C    0210        LD   HL,JTAB
0C92 19        0220        ADD  HL,DE
0C93 E9        0230        JP   (HL)
0C94 180E      0240 JTAB   JR   U0
0C96 180E      0250        JR   U1
0C98 1814      0260        JR   U2
0C9A 1819      0270        JR   U3
0C9C 1819      0280        JR   U4
0C9E 1819      0290        JR   U5
0CA0 1819      0300        JR   U6
0CA2 1819      0310        JR   U7
               0320 ;
               0330 ;USR(0)
0CA4 1819      0340 U0     JR   ERROR
               0350 ;USR(1) IS GET
0CA6 DF7B      0360 U1     SCAL BLINK
0CA8 47        0370 RTA    LD   B,A
0CA9 AF        0380        XOR  A
0CAA 2A0DE0    0390        LD   HL,(0E00DH)
0CAD E9        0400        JP   (HL)
               0410 ;
               0420 ;USR(2) IS INKEY
0CAE DF61      0430 U2     SCAL KBD
0CB0 38F6                  JR   C,RTA
0CB2 AF                    XOR  A
0CB3 18F3                  JR   RTA
0CB5 1808      0470 U3     JR   ERROR
0CB7 1806      0480 U4     JR   ERROR
0CB9 1804      0490 U5     JR   ERROR
0CBB 1802      0500 U6     JR   ERROR
0CBD 1800      0510 U7     JR   ERROR
               0520 ;
0CBF DF6B      0530 ERROR  SCAL ERRM
0CC1 C9        0540        RET
```

If arguments are required to be transmitted from the BASIC program to the USR routine, they may be POKEd or DOKEd into free ram locations and then picked up by the USR routine. Similarly, if the USR routine is to transmit arguments back to BASIC, the USR routine should store them in free ram from where they can be picked up in BASIC using PEEK or DEEK.

5.2 Keyboard characteristics, K mode

The NAS-SYS Kx command allows the characteristics of the keyboard to be changed. The characteristics are determined by the contents of workspace loaction $KOPT (0C27H=3111d). The effect of a Kx command may be achieved simply by the BASIC statement:

POKE 3111,x where x=0,1,4, or 5.
Refer to the K command in Chapter 2 for the required value of x.

5.3 Activating User-written I/O, U mode

A user-written output routine may be activated by including UOUT in the table of output device. The required BASIC statement is
DOKE 3187,1912 (1918 for NAS-SYS 1)
The address of the start of the routine must of course have been placed in workspace location $UOUT+1/2 (0C78/9H=3192/3d) when the routine was loaded.

Similarly, a user-written input routine may be activated by
DOKE 3189,1915 (1921 in NAS-SYS 1)
Again, the start address of the routine must be in $UIN+1/2 (0C7B/CH=3195/6d).
Refer to the U command, Chapter 2, and the UOUT routine, Chapter 3.

5.4 Activating the External Serial Device, X mode

The external serial output driver requires an argument in workspace location $XOPT (0C28H=3112d); refer to the X command in Chapter 2, and the XOUT routine in Chapter 3.
The required BASIC statements are
DOKE 3187,1911 (1917 for NAS-SYS 1)
POKE 3112, x
where x=0,1,16, or 17.

The external serial input driver also requires an argument in $XOPT; refer to the X command, Chapter 2, and the XKBD routine in Chapter 3.
The required BASIC statements are
DOKE 3189,1919 (1925 for NAS-SYS 1)
POKE 3112,x
where x=0,2,16,18,32,34,48,or 50.

When both the serial output and input are activated together, argument x must be the logical OR of the input and output options.

5.5 Deactivating the U and X modes, N mode

The U and X modes may be deactivated by the NAS-SYS command, N, as described in Chapter 2. In BASIC this becomes
DOKE 3187,1913 (1919 in NAS-SYS 1)
DOKE 3189,1916 (1922 in NAS-SYS 1)