ROM BIOS

The design of the MSX ROM is of importance if machine code programs are to be developed efficiently and operate reliably. Almost every program, including the BASIC Interpreter itself, will require a certain set of primitive functions to operate. These include screen and printer drivers, a keyboard decoder and other hardware related functions. By separating these routines from the BASIC Interpreter they can be made available to any application program. The section of ROM from 0000H to 268BH is largely devoted to such routines and is called the ROM BIOS (Basic Input Output System). This chapter gives a functional description of every recognizably separate routine in the ROM BIOS. Special attention is given to the "standard" routines. These are documented by Microsoft and guaranteed to remain consistent through possible hardware and software changes. The first few undred bytes of the ROM consists of Z80 JP instructions which provide fixed position entry points to these routines. For maximum compatibility with future software an application program should restrict its dependence on the ROM to these locations only. The description of the ROM begins with this list of entry points to the standard routines. A brief comment is placed with each entry point, the full description is given with the routine itself.

4.1 Data Areas

It is expected that most users will wish to disassemble the ROM to some extent (the full listing runs to nearly four hundred pages). In order to ease this process the data areas, which do not contain executable Z80 code, are shown below:

```
0004H-0007H   185DH-1863H 4B3AH-4B4CH   73E4H-73E4H
002BH-002FH   1B97H-1BAAH 4C2FH-4C3FH   752EH-7585H
0508H-050DH   1BBFH-23BEH 555AH-5569H   7754H-7757H
092FH-097FH   2439H-2459H 5D83H-5DB0H   7BA3H-7BCAH
0DA5H-0EC4H   2CF1H-2E70H 6F76H-6F8EH   7ED8H-7F26H
1033H-105AH   3030H-3039H 70FFH-710CH   7F41H-7FB6H
1061H-10C1H   3710H-3719H 7182H-7195H   7FBEH-7FFFH
1233H-1252H   392EH-3FE1H 71A2H-71B5H
13A9H-1448H   43B5H-43C3H 71C7H-71DAH
160BH-1612H   46E6H-46E7H 72A6H-72B9H
```

Note that these data areas are for the UK ROM, there are slight differences in the Japanese ROM relating to the keyboard decoder and the video character set. Disparities between the ROMs are restricted to these regions with the bulk of the codebeing identical in both cases.

4.2 Terminology

Reference is frequently made in this chapter to the standard routines and to Workspace Area variables. Whenever this is done the Microsoft recommended name is used in upper case letters, for example "the FILVRM standard routine" and "SCRMOD is set". Subroutines which are not named are referred to by a parenthesized address, "the screen is cleared (0777H)" for example. When reference is made to the Z80 status flags assembly language conventions are used, for example "Flag C" would mean that the carry flag is set while "Flag NZ" means that the zero flag is reset. The terms "EI" and "DI" mean enabled interrupts and disabled interrupts respectively.

ADDR	NAME	TO	FUNCTION
0000H	CHKRAM	02D7H	Power-up, check RAM
0004H	.....		Two bytes, address of ROM character set
0006H	.....		One byte, VDP Data Port number
0007H	.....		One byte, VDP Data Port number
0008H	SYNCHR	2683H	Check BASIC program character
000BH	.....		NOP
000CH	RDSLT	01B6H	Read RAM in any slot
000FH	.....		NOP
0010H	CHRGTR	2686H	Get next BASIC program character
0013H	.....		NOP
0014H	WRSLT	01D1H	Write to RAM in any slot
0017H	.....		NOP
0018H	OUTDO	1B45H	Output to current device
001BH	.....		NOP
001CH	CALSLT	0217H	Call routine in any slot
001FH	.....		NOP
0020H	DCOMPR	146AH	Compare register pairs HL and DE
0023H	.....		NOP
0024H	ENASLT	025EH	Enable any slot permanently
0027H	.....		NOP
0028H	GETYPR	2689H	Get BASIC operand type
002BH	.....		Five bytes Version Number
0030H	CALLF	0205H	Call routine in any slot
0033H	.....		FiveNOPs
0038H	KEYI NT	0C3CH	Interru pt handler, keyboard scan
003BH	INITIO	049DH	Initialize I/O devices
003EH	INIFNK	139DH	Initialize function key strings
0041H	DISSCR	0577H	Disable screen
0044H	ENA SCR	0570H	Enable screen
0047H	WRTVDP	057FH	Write to any VDP register

004AH	RDVRM	07D7H	Read byte from VRAM
004DH	WRTVRM	07CDH	Write byte to VRAM
0050H	SETRD	07ECH	Set up VDP for read
0053H	SETWR T	07DFH	Set up VDP for write
0056H	FILVRM	0815H	Fill block of VRAM with data byte
0059H	LDIRMV	070FH	Copy block to memory from VRAM
005CH	LDIRVM	0744H	Copy block to VRAM, from memory
005FH	CHGMOD	084FH	Change VDP mode
0062H	CHGCLR	07F7H	Change VDP colours
0065H			NOP
0066H	NMI	1398H	Non Maskable Interrupt handler
0069H	CLRSPR	06A8H	Clear all sprites
006CH	INITXT	050EH	Initialize VDP to 40x24 Text Mode
006FH	INIT32	0538H	Initialize VDP to 32x24 Text Mode
0072H	INIGRP	05D2H	Initialize VDP to Graphics Mode
0075H	INIMLT	061FH	Initialize VDP to Multicolour Mode
0078H	SETTXT	0594H	Set VDP to 40x24 Text Mode
007BH	SETT32	05B4H	Set VDP to 32x24 Text Mode
007EH	SETGRP	0602H	Set VDP to Graphics Mode
0081H	SETMLT	0659H	Set VDP to Multicolour Mode
0084H	CALPAT	06E4H	Calculate address of sprite pattern
0087H	CALATR	06F9H	Calculate address of sprite attribute
008AH	GSPSIZ	0704H	Get sprite size
008DH	GRPPRT	1510H	Print character on graphic screen
0090H	GICINI	04BDH	Initialize PSG (GI Chip)
0093H	WRTPSG	1102H	Write to any PSG register
0096H	RDPSG	110EH	Read from any PSG register
0099H	STRTMS	11C4H	Start music dequeueing
009CH	CHSNS	0D6AH	Sense keyboard buffer for character
009FH	CHGET	10CBH	Get character from keyboard buffer (wait)
00A2H	CHPUT	08BCH	Screen character output
00A5H	LPTOUT	085DH	Line printer character output
00A8H	LPTSTT	0884H	Line printer status test
00ABH	CNVCHR	089DH	Convert character with graphic header
00AEH	PINLIN	23BFH	Get line from console (editor)
00B1H	INLIN	23D5H	Get line from console (editor)
00B4H	QINLIN	23CCH	Display "?", get line from console (editor)
00B7H	BREAKX	046FH	Check CTRL-STOP key directly
00BAH	ISCNTC	03FBH	Check CRTL-STOP key
00BDH	CKCNTC	10F9H	Check CTRL-STOP key
00C0H	BEEP	1113H	Go beep
00C3H	CLS	0848H	Clear screen
00C6H	POSIT	088EH	Set cursor position
00C9H	FNKSB	0B26H	Check if function key display on
00CCH	ERAFNK	0B15H	Erase function key display
00CFH	DSPFNK	0B2BH	Display function keys
00D2H	TOTEXT	083BH	Return VDP to text mode
00D5H	GTSTCK	11EEH	Get joystick status
00D8H	GTTRIG	1253H	Get trigger status
00DBH	GTPAD	12ACH	Get touch pad status
00DEH	GTPDL	1273H	Get paddle status
00E1H	TAPION	1A63H	Tape input ON
00E4H	TAPIN	1ABCH	Tape input
00E7H	TAPIOF	19E9H	Tape input OFF
00EAH	TAPOON	19F1H	Tape output ON
00EDH	TAPOUT	1A19H	Tape output
00F0H	TAPOOF	19DDH	Tape output OFF
00F3H	STMOTR	1384H	Turn motor ON/OFF
00F6H	LFTQ	14EBH	Space left in music queue
00F9H	PUTQ	1492H	Put byte in music queue
00FCH	RIGHTC	16C5H	Move current pixel physical address right
00FFH	LEFTC	16EEH	Move current pixel physical address left
0102H	UPC	175DH	Move current pixel physical address up
0105H	TUPC	173CH	Test then UPC if legal
0108H	DOWNC	172AH	Move current pixel physical address down
010BH	TDOWNC	170AH	Test then DOWNC if legal
010EH	SCALXY	1599H	Scale graphics coordinates
0111H	MAPXYC	15DFH	Map graphic coordinates to physical address
0114H	FETCHC	1639H	Fetch current pixel physical address
0117H	STOREC	1640H	Store current pixel physical address
011AH	SETATR	1676H	Set attribute byte
011DH	READC	1647H	Read attribute of current pixel
0120H	SETC	167EH	Set attribute of current pixel
0123H	NSETCX	1809H	Set attribute of number of pixels
0126H	GTASPC	18C7H	Get aspect ratio
0129H	PNTINI	18CFH	Paint initialize
012CH	SCANR	18E4H	Scan pixels to right
012FH	SCANL	197AH	Scan pixels to left
0132H	CHGCAP	0F3DH	Change Caps Lock LED
0135H	CHGSND	0F7AH	Change Key Click sound output
0138H	RSLREG	144CH	Read Primary Slot Register
013BH	WSLREG	144FH	Write to Primary Slot Register
013EH	RDVDP	1449H	Read VDP Status Register
0141H	SNSMAT	1452H	Read row of keyboard matrix
0144H	PHYDIO	148AH	Disk, no action
0147H	FORMAT	148EH	Disk, no action
014AH	ISFLIO	145FH	Check for file I/O
014DH	OUTDLP	1B63H	Formatted output to line printer
0150H	GETVCP	1470H	Get music voice pointer
0153H	GETVC2	1474H	Get music voice pointer
0156H	KILBUF	0468H	Clear keyboard buffer
0159H	CALBAS	01FFH	Call to BASIC from any slot
015CH			NOPs to
01B5H			for expansion

```
Address... 01B6H
Name...... RDSLT
Entry..... A=Slot ID, HL=Address
Exit...... A=Byte read
Modifies.. AF, BC, DE, DI
```

Standard routine to read a single byte from memory in any slot. The Slot Identifier is composed of a Primary Slot number, a Secondary Slot number and a flag:

7	6	5	4	3	2	1	0
Flag	0	0	0	Secondary Slot#		Primary Slot#	

Figure 34: Slot ID

The **Flag** is normally 0 but must be 1 if a Secondary Slot number is included in the Slot ID. The memory address and Slot ID are first processed (027EH) to yield a set of bit masks to apply to the relevant slot register. If a **Secondary Slot Number** is specified then the Secondary Slot Register is first modified to select the relevant page from that Secondary Slot (02A3H). The Primary Slot is then switched in to the Z80 address space, the byte read and the Primary Slot restored to its original setting via the RDPRIM routine in the Workspace Area. Finally, if a **Secondary Slot Number** is included in the Slot ID, the original Secondary Slot Register setting is restored (01ECH). Note that, unless it is the slot containing the Workspace Area, any attempt to access page 3 (C000H to FFFFH) will cause the system to crash as RDPRIM will switch itself out. Note also that interrupts are left disabled by all the memory switching routines.

```
Address... 01D1H
Name...... WRSLT
Entry..... A=Slot ID, HL=Address, E=Byte to write
Exit...... None
Modifies.. AF, BC, D, DI
```

Standard routine to write a single byte to memory in any slot. Its operation is fundamentally the same as that of the RDSLT standard routine except that the Workspace Area routine WRPRIM is used rather than RDPRIM.

```
Address... 01FFH
Name...... CALBAS
Entry..... IX=Address
Exit...... None
Modifies.. AF', BC', DE', HL', IY, DI
```

Standard routine to call an address in the BASIC Interpreter from any slot. Usually this will be from a machine code program running in an extension ROM in page 1 (4000H to 7FFFH). The high byte of register pair IY is loaded with the MSX ROM Slot ID (00H) and control transfers to the CALSLT standard routine.

```
Address... 0205H
Name...... CALLF
Entry..... None
Exit...... None
Modifies.. AF', BC', DE', HL', IX, IY, DI
```

Standard routine to call an address in any slot. The Slot ID and address are supplied as inline parameters rather than in registers to fit inside a hook (Chapter 6), for example:

```
RST 30H
DEFB Slot ID
DEFW Address
RET
```

The Slot ID is first collected and placed in the high byte of register pair IY. The address is then placed in register pair IX and control drops into the CALSLT standard routine.

```
Address... 0217H
Name...... CALSLT
Entry..... IY(High byte)=Slot ID, IX=Address
Exit...... None
Modifies.. AF', BC', DE', HL', DI
```

Standard routine to call an address in any slot. Its operation is fundamentally the same as that of the RDSLT standard routine except that the Workspace Area routine CLPRIM is used rather than RDPRIM. Note that CALBAS and CALLF are just specialized entry points to this standard routine which offer a reduction in the amount of code required.

```
Address... 025EH
Name...... ENASLT
Entry..... A=Slot ID, HL=Address
Exit...... None
Modifies.. AF, BC, DE, DI
```

Standard routine to switch in a page permanently from any slot. Unlike the RDSLT, WRSLT and CALSLT standard routines the Primary Slot switching is performed directly and not by a Workspace Area routine. Consequently addresses in page 0 (0000H to 3FFFH) will cause an immediate system crash.

```
Address... 027EH
```

This routine is used by the memory switching standard routines to turn an address, in register pair HL, and a Slot ID, in register A, into a set of bit masks. As an example a Slot ID of FxxxSSPP and an address in Page 1 (4000H to 7FFFH) would return the following:

```
Register B= 00 00 PP 00 (OR mask)
Register C= 11 11 00 11 (AND mask)
Register D= PP PP PP PP (Replicated)
Register E= 00 00 11 00 (Page mask)
```

Registers B and C are derived from the Primary Slot number andthe page mask. They are later used to mix the new Primary Slotnumber into the existing contents of the Primary Slot Register. Register D contains the Primary Slot number replicated four times and register E the page mask. This is produced by examining the two most significant bits of the address, to determine the page number, and then shifting the mask along to the relevant position. These registers are later used during Secondary Slot switching. As the routine terminates bit 7 of the Slot ID is tested, to determine whether a Secondary Slot has been specified, and Flag M returned if this is so.

```
Address... 02A3H
```

This routine is used by the memory switching standard routines to modify a Secondary Slot Register. The Slot ID is supplied in register A while registers D and E contain the bit masks shown in the previous routine. Bits 6 and 7 of register D are first copied into the Primary Slot register. This switches in page 3 from the Primary Slot specified by the Slot ID and makes the required Secondary Slot Register available. This is then read from memory location FFFFH and the page mask, inverted, used to clear the required two bits. The Secondary Slot number is shifted to the relevant position and mixed in. Finally the new setting is placed in the Secondary Slot Register and the Primary Slot Register restoredto its original setting.

```
Address... 02D7H
Name...... CHKRAM
Entry..... None
Exit...... None
Modifies.. AF, BC, DE, HL, SP
```

Standard routine to perform memory initialization at power-up. It non-destructively tests for RAM in pages 2 and 3 in all sixteen possible slots then sets the Primary and Secondary Slot registers to switch in the largest area found. The entire Workspace Area (F380H to FFC9H) is zeroed and EXPTBL and SLTTBL filled in to map any expansion interfaces in existence Interrupt Mode 1 is set and control transfers to the remainder of the power-up initialization routine (7C76H).

```
Address... 03FBH
Name...... ISCNTC
Entry..... None
Exit...... None
Modifies.. AF, EI
```

Standard routine to check whether the CTRL-STOP or STOP keys have been pressed. It is used by the BASIC Interpreter at the end of each statement to check for program termination. BASROM is first examined to see if it contains a non-zero value, if so the routine terminates immediately. This is to prevent users breaking into any extension ROM containing a BASIC program. INTFLG is then checked to determine whether the interrupt handler has placed the CTRL-STOP or STOP key codes (03H or 04H) there. If STOP has been detected then the cursor is turned on (09DAH) and INTFLG continually checked until one of the two key codes reappears. The cursor is then turned off (0A27H) and, if the key is STOP, the routine terminates. If CTRL-STOP has been detected then the keyboard buffer is first cleared via the KILBUF standard routine and TRPTBL is checked to see whether an "ON STOP GOSUB" statement is active. If so the relevant entry in TRPTBL is updated (0EF1H) and the routine terminates as the event will be handled by the Interpreter Runloop. Otherwise the ENASLT standard routine is used to switch in page 1 from the MSX ROM, in case an extension ROM is using the routine, and control transfers to the "STOP" statement handler (63E6H).

```
Address... 0468H
Name...... KILBUF
Entry..... None
Exit...... None
Modifies.. HL
```

Standard Routine to clear the forty character type-ahead keyboard buffer KEYBUF. There are two pointers into this buffer, PUTPNT where the interrupt handler places characters, and GETPNT where application programs fetch them from. As the number of characters in the buffer is indicated by the difference between these two pointers KEYBUF is emptied simply by making them both equal.

```
Address... 046FH
Name...... BREAKX
Entry..... None
Exit...... Flag C if CTRL-STOP key pressed
Modifies.. AF
```

Standard routine which directly tests rows 6 and 7 of the keyboard to determine whether the CTRL and STOP keys are both pressed. If they are then KEYBUF is cleared and row 7 of OLDKEY modified to prevent the interrupt handler picking the keys up as well. This routine may often be more suitable for use by an application program, in preference to ISCNTC, as it will work when interrupts are disabled, during cassette I/O for example, and does not exit to the Interpreter.

```
Address... 049DH
Name...... INITIO
Entry..... None
Exit...... None
Modifies.. AF, E, EI
```

Standard routine to initialize the PSG and the Centronics Status Port. PSG Register 7 is first set to 80H making PSG Port B=Output and PSG Port A=Input. PSG Register 15 is set to CFH to initialize the Joystick connector control hardware. PSG Register 14 is then read and the Keyboard Mode bit placed in KANAMD, this has no relevance for UK machines. Finally a value of FFH is output to the Centronics Status Port (I/O port 90H) to set the STROBE signal high. Control then drops into the GICINI standard routine to complete initialization.

```
Address... 04BDH
Name...... GICINI
Entry..... None
Exit...... None
Modifies.. EI
```

Standard routine to initialize the PSG and the Workspace Area variables associated with the "PLAY" statement. QUETAB, VCBA, VCBB and VCBC are first initialized with the values shown in Chapter 6. PSG Registers 8, 9 and 10

are then set to zero amplitude and PSG Register 7 to B8H. This enables the Tone Generator and disables the Noise Generator on each channel.

```
Address... 0508H
```

This six byte table contains the "PLAY" statement parameters initially placed in VCBA, VCBB and VCBC by the GICINI standard routine: Octave=4, Length=4, Tempo=120, Volume=88H, Envelope=00FFH.

```
Address... 050EH
Name...... INITXT
Entry..... None
Exit...... None
Modifies.. AF, BC, DE, HL, EI
```

Standard routine to initialize the VDP to 40x24 Text Mode. The screen is temporarily disabled via the DISSCR standard routine and SCRMOD and OLDSCR set to 00H. The parameters required by the CHPUT standard routine are set up by copying LINI.40 to LINLEN, TXTNAM to NAMBAS and TXTCGP to CGPBAS. The VDP colours are then set by the CHGCLR standard routine and the screen is cleared (077EH). The current character set is copied into the VRAM Character Pattern Table (071EH). Finally the VDP mode and base addresses are set via the SETTXT standard routine and the screen is enabled.

```
Address... 0538H
Name...... INIT32
Entry..... None
Exit...... None
Modifies.. AF, BC, DE, HL, EI
```

Standard routine to initialize the VDP to 32x24 Text Mode. The screen is temporarily disabled via the DISSCR standard routine and SCRMOD and OLDSCR set to 01H. The parameters required by the CHPUT standard routine are set up by copying LINL32 to LINLEN, T32NAM to NAMBAS, T32CGP to CGPBAS, T32PAT to PATBAS and T32ATR to ATRBAS. The VDP colours are then set via the CHGCLR standard routine and the screen is cleared (077EH). The current character set is copied into the VRAM Character Pattern Table (071EH) and all sprites cleared (06BBH). Finally the VDP mode and base addresses are set via the SETT32 standard routine and the screen is enabled.

```
Address... 0570H
Name...... ENASCR
Entry..... None
Exit...... None
Modifies.. AF, BC, EI
```

Standard routine to enable the screen. This simply involves setting bit 6 of VDP Mode Register 1.

```
Address... 0577H
Name...... DISSCR
Entry..... None
Exit...... None
Modifies.. AF, BC, EI
```

Standard routine to disable the screen. This simply involves resetting bit 6 of VDP Mode Register 1.

```
Address... 057FH
Name...... WRTVDP
Entry..... B=Data byte, C=VDP Mode Register number
Exit...... None
Modifies.. AF, B, EI
```

Standard routine to write a data byte to any VDP Mode Register. The register selection byte is first written to the VDP Command Port, followed by the data byte. This is then copied to the relevant register image, RGOSAV to RG7SAV, in the Workspace Area

```
Address... 0594H
Name...... SETTXT
Entry..... None
Exit...... None
Modifies.. AF, BC, DE, HL, EI
```

Standard routine to partially set the VDP to 40x24 Text Mode. The mode bits M1, M2 and M3 are first set in VDP Mode Registers 0 and 1. The five VRAM table base addresses, beginning with TXTNAM, are then copied from the Workspace Area into VDP Mode Registers 2, 3, 4, 5 and 6 (0677H).

```
Address... 05B4H
Name...... SETT32
Entry..... None
Exit...... None
Modifies.. AF, BC, DE, HL, EI
```

Standard routine to partially set the VDP to 32x24 Text Mode. The mode bits M1, M2 and M3 are first set in VDP Mode Registers 0 and 1. The five VRAM table base addresses, beginning with T32NAM, are then copied from the Workspace Area into VDP Mode Registers 2, 3, 4, 5 and 6 (0677H).

```
Address... 05D2H
Name...... INIGRP
Entry..... None
Exit...... None
Modifies.. AF, BC, DE, HL, EI
```

Standard routine to initialize the VDP to Graphics Mode. The screen is temporarily disabled via the DISSCR standard routine and SCRMOD set to 02H. The parameters required by the GRPPRT standard routine are set up by copying GRPPAT to PATBAS and GRPATR to ATRBAS. The character code driver pattern is then copied into the VDP Name Table, the screen cleared (07A1H) and all sprites cleared (06BBH). Finally the VDP mode and base addresses are set via the SETGRP standard routine and the screen is enabled.

```
Address... 0602H
Name...... SETGRP
Entry..... None
Exit...... None
Modifies.. AF, BC, DE, HL, EI
```

Standard routine to partially set the VDP to Graphics Mode. The mode bits M1, M2 and M3 are first set in VDP Mode Registers 0 and 1. The five VRAM table base addresses, beginning with GRPNAM, are then copied from the Workspace Area into VDP Mode Registers 2, 3, 4, 5 and 6 (0677H).

```
Address... 061FH
Name...... INIMLT
Entry..... None
Exit...... None
Modifies.. AF, BC, DE, HL, EI
```

Standard routine to initialize the VDP to Multicolour Mode. The screen is temporarily disabled via the DISSCR standard routine and SCRMOD set to 03H. The parameters required by the GRPPRT standard routine are set up by copying MLTPAT to PATBAS and MLTATR to ATRBAS. The character code driver pattern is then copied into the VDP Name Table, the screen cleared (07B9H) and all sprites cleared (06BBH). Finally the VDP mode and base addresses are set via the SETMLT standard routine and the screen is enabled.

```
Address... 0659H
Name...... SETMLT
Entry..... None
Exit...... None
Modifies.. AF, BC, DE, HL, EI
```

Standard routine to partially set the VDP to Multicolour Mode. The mode bits M1, M2 and M3 are first set in VDP Mode Registers 0 and 1. The five VRAM table base addresses, beginning with MLTNAM, are then copied from the Workspace Area to VDP Mode Registers 2, 3, 4, 5 and 6.

```
Address... 0677H
```

This routine is used by the SETTXT, SETT32, SETGRP and SETMLT standard routines to copy a block of five table base addresses from the Workspace Area into VDP Mode Registers 2, 3, 4, 5 and 6. On entry register pair HL points to the relevant group of addresses. Each base address is collected in turn shifted the required number of places and then written to the relevant Mode Register via the WRTVDP standard routine.

```
Address... 06A8H
Name...... CLRSPR
Entry..... None
Exit...... None
Modifies.. AF, BC, DE, HL, EI
```

Standard routine to clear all sprites. The entire 2 KB Sprite Pattern Table is first filled with zeros via the FILVRM standard routine. The vertical coordinate of each of the thirty-two sprite attribute blocks is then set to -47 (D1H) to place the sprite above the top of the screen, the horizontal coordinate is left unchanged. The pattern numbers in the Sprite Attribute Table are initialized with the series 0, 1, 2, 3, 4,... 31 for 8x8 sprites or the series 0, 4, 8, 12, 16,... 124 for 16x16 sprites. The series to be generated is determined by the Size bit in VDP Mode Register 1. Finally the colour byte of each sprite attribute block is filled in with the colour code contained in FORCLR, this is initially white. Note that the Size and Mag bits in VDP Mode Register 1 are not affected by this routine. Note also that the INIT32, INIGRP and INIMLT standard routines use this routine with an entry point at 06BBH, leaving the Sprite Pattern Table undisturbed.

```
Address... 06E4H
Name...... CALPAT
Entry..... A=Sprite pattern number
Exit...... HL=Sprite pattern address
Modifies.. AF, DE, HL
```

Standard routine to calculate the address of a sprite pattern. The pattern number is first multiplied by eight then, if 16x16 sprites are selected, multiplied by a further factor of four. This is then added to the Sprite Pattern Table base address, taken from PATBAS, to produce the final address. This numbering system is in line with the BASIC Interpreter's usage of pattern numbers rather than the VDP's when 16x16 sprites are selected. As an example while the Interpreter calls the second pattern number one, it is actually VDP pattern number four. This usage means that the maximum pattern number this routine should allow, when 16x16 sprites are selected, is sixty-three. There is no actual check on this limit so large pattern numbers will produce addresses greater than 3FFFH. Such addresses, when passed to the other VDP routines, will wrap around past zero and corrupt the Character Pattern Table in VRAM.

```
Address... 06F9H
Name...... CALATR
Entry..... A=Sprite number
Exit...... HL=Sprite attribute address
Modifies.. AF, DE, HL
```

Standard routine to calculate the address of a sprite attribute block. The sprite number, from zero to thirty-one, is multiplied by four and added to the Sprite Attribute Table base address taken from ATRBAS.

```
Address... 0704H
Name...... GSPSIZ
Entry..... None
Exit...... A=Bytes in sprite pattern (8 or 32)
Modifies.. AF
```

Standard routine to return the number of bytes occupied by each sprite pattern in the Sprite Pattern Table. The result is determined simply by examining the Size bit in VDP Mode Register 1.

```
Address... 070FH
Name...... LDIRMV
Entry..... BC=Length, DE=RAM address, HL=VRAM address
Exit...... None
Modifies.. AF, BC, DE, EI
```

Standard routine to copy a block into main memory from the VDP VRAM. The VRAM starting address is set via the SETRD standard routine and then sequential bytes read from the VDP Data Port and placed in main memory.

```
Address... 071EH
```

This routine is used to copy a 2 KB character set into the VDP Character Pattern Table in any mode. The base address of the Character Pattern Table in VRAM is taken from CGPBAS. The starting address of the character set is taken from CGPNT. The RDSLT standard routine is used to read the character data so this may be situated in an extension ROM. At power-up CGPNT is initialized with the address contained at ROM location 0004H, which is 1BBFH. CGPNT is easily altered to produce some interesting results, POKE &HF920,&HC7:SCREEN 0 provides a thoroughly confusing example.

```
Address... 0744H
Name...... LDIRVM
Entry..... BC=Length, DE=VRAM address, HL=RAM address
Exit...... None
Modifies.. AF, BC, DE, HL, EI
```

Standard routine to copy a block to VRAM from main memory. The VRAM starting address is set via the SETWRT standard routine and then sequential bytes taken from main memory and written to the VDP Data Port.

```
Address... 0777H
```

This routine will clear the screen in any VDP mode. In 40x24 Text Mode and 32x24 Text Mode the Name Table, whose base address is taken from NAMBAS, is first filled with ASCII spaces. The cursor is then set to the home position (0A7FH) and LINTTB, the line termination table, re-initialized. Finally the function key display is restored, if it is enabled, via the FNKSB standard routine. In Graphics Mode the border colour is first set via VDP Mode Register 7 (0832H). The Colour Table is then filled with the background colour code, taken from BAKCLR, for both 0 and 1 pixels. Finally the Character Pattern Table is filled with zeroes. In Multicolour Mode the border colour is first set via VDP Mode Register 7 (0832H). The Character Pattern Table is then filled with the background colour taken from BAKCLR.

```
Address... 07CDH
Name...... WRTVRM
Entry..... A=Data byte, HL=VRAM address
Exit...... None
Modifies.. EI
```

Standard routine to write a single byte to the VDP VRAM. The VRAM address is first set up via the SETWRT standard routine and then the data byte written to the VDP Data Port. Note that the two seemingly spurious EX(SP),HL instructions in this routine, and several others, are required to meet the VDP's timing constraints.

```
Address... 07D7H
Name...... RDVRM
Entry..... HL=VRAM address
Exit...... A=Byte read
```
Modifies.. AF, EI

Standard routine to read a single byte from the VDP VRAM. The VRAM address is first set up via the SETRD standard routine and then the byte read from the VDP Data Port.

```
Address... 07DFH
Name...... SETWRT
Entry..... HL=VRAM address
Exit...... None
Modifies.. AF, EI
```

Standard routine to set up the VDP for subsequent writes to VRAM via the Data Port. The address contained in register pair HL is written to the VDP Command Port LSB first, MSB second as shown in Figure 7. Addresses greater than 3FFFH will wrap around past zero as the two most significant bits of the address are ignored.

```
Address... 07ECH
Name...... SETRD
Entry..... HL=VRAM address
Exit...... None
Modifies.. AF, EI
```

Standard routine to set up the VDP for subsequent reads from VRAM via the Data Port. The address contained in register pair HL is written to the VDP Command Port LSB first, MSB second as shown in Figure 7. Addresses greater than 3FFFH will wrap around past zero as the two most significant bits of the address are ignored.

```
Address... 07F7H
Name...... CHGCLR
Entry..... None
Exit...... None
Modifies.. AF, BC, HL, EI
```

Standard routine to set the VDP colours. SCRMOD is first examined to determine the appropriate course of action. In 40x24 Text Mode the contents of BAKCLR and FORCLR are written to VDP Mode Register 7 to set the colour of the 0 and 1 pixels, these are initially blue and white. Note that in this mode there is no way of specifying the border colour, this will be the same as the 0 pixel colour. In 32x24 Text Mode, Graphics Mode or Multicolour Mode the contents of BDRCLR are written to VDP Mode Register 7 to set the colour of the border, this is initially blue. Also in 32x24 Text Mode the contents of BAKCLR and FORCLR are copied to the whole of the Colour Table to determine the 0 and 1 pixel colours.

```
Address... 0815H
Name...... FILVRM
Entry..... A=Data byte, BC=Length, HL=VRAM address
Exit...... None
Modifies.. AF, BC, EI
```

Standard routine to fill a block of the VDP VRAM with a single data byte. The VRAM starting address, contained in register pair HL, is first set up via the SETWRT standard routine. The data byte is then repeatedly written to the VDP Data Port to fill successive VRAM locations.

```
Address... 083BH
Name...... TOTEXT
Entry..... None
Exit...... None
Modifies.. AF, BC, DE, HL, EI
```

Standard routine to return the VDP to either 40x24 Text Mode or 32x24 Text Mode if it is currently in Graphics Mode or Multicolour Mode. It is used by the BASIC Interpreter Mainloop and by the "INPUT" statement handler. Whenever the INITXT or INIT32 standard routines are used the mode byte, 00H or 01H, is copied into OLDSCR. If the mode is subsequently changed to Graphics Mode or Multicolour Mode, and then has to be returned to one of the two text modes for keyboard input, this routine ensures that it returns to the same one. SCRMOD is first examined and, if the screen is already in either text mode, the routine simply terminates with no action. Otherwise the previous text mode is taken from OLDSCR and passed to the CHGMOD standard routine.

```
Address... 0848H
Name...... CLS
Entry..... Flag Z
Exit...... None
Modifies.. AF, BC, DE, EI
```

Standard routine to clear the screen in any mode, it does nothing but call the routine at 0777H. This is actually the "CLS" statement handler and, because this indicates that there is illegal text after the statement, it will simply return if entered with Flag NZ.

```
Address... 084FH
Name...... CHGMOD
Entry..... A=Screen mode required (0, 1, 2, 3)
Exit...... None
Modifies.. AF, BC, DE, HL, EI
```

Standard routine to set a new screen mode. Register A, containing the required screen mode, is tested and control transferred to INITXT, INIT32, INIGRP or INIMLT.

```
Address... 085DH
Name...... LPTOUT
Entry..... A=Character to print
Exit...... Flag C if CTRL-STOP termination
Modifies.. AF
```

Standard routine to output a character to the line printer via the Centronics Port. The printer status is continually tested, via the LPTSTT standard routine, until the printer becomes free. The character is then written to the Centronics Data Port (I/O port 91H) and the STROBE signal of the Centronics Status Port (I/O port 90H) briefly pulsed low. Note that the BREAKX standard routine is used to test for the CTRL-STOP key if the printer is busy. If CTRL-STOP is detected a CR code is written to the Centronics Data Port, to flush the printer's line buffer, and the routine terminates with Flag C.

```
Address... 0884H
Name...... LPTSTT
Entry..... None
Exit...... A=0 and Flag Z if printer busy
Modifies.. AF
```

Standard routine to test the Centronics Status Port BUSY signal. This just involves reading I/O port 90H and examining the state of bit 1: 0=Ready, 1=Busy.

```
Address... 088EH
Name...... POSIT
Entry..... H=Column, L=Row
Exit...... None
Modifies.. AF, EI
```

Standard routine to set the cursor coordinates. The row and column coordinates are sent to the OUTDO standard routine as the parameters in an ESC,"Y",Row+1FH, Column+1FH sequence. Note that the BIOS home position has coordinates of 1,1 rather than the 0,0 used by the BASIC Interpreter.

```
Address... 089DH
Name...... CNVCHR
Entry..... A=Character
Exit...... Flag Z,NC=Header; Flag NZ,C=Graphic; Flag Z,C=Normal
Modifies.. AF
```

Standard routine to test for, and convert if necessary, characters with graphic headers. Characters less than 20H are normally interpreted by the output device drivers as control characters. A character code in this range can be treated as a displayable character by preceding it with a graphic header control code (01H) and adding 40H to its value. For example to directly display character code 0DH, rather than have it interpreted as a carriage return, it is necessary to output the two bytes 01H,4DH. This routine is used by the output device drivers, such as the CHPUT standard routine,

to check for such sequences. If the character is a graphic header GRPHED is set to 01H and the routine terminates, otherwise GRPHED is zeroed. If the character is outside the range 40H to 5FH it is left unchanged. If it is inside this range, and GRPHED contains 01H indicating a previous graphic header, it is converted by subtracting 40H.

```
Address... 08BCH
Name...... CHPUT
Entry..... A=Character
Exit...... None
Modifies.. EI
```

Standard routine to output a character to the screen in 40x24 Text Mode or 32x24 Text Mode. SCRMOD is first checked and, if the VDP is in either Graphics Mode or Multicolour Mode, the routine terminates with no action. Otherwise the cursor is removed (0A2EH), the character decoded (08DFH) and then the cursor replaced (09E1H). Finally the cursor column position is placed in TTYPOS, for use by the "PRINT" statement, and the routine terminates.

```
Address... 08DFH
```

This routine is used by the CHPUT standard routine to decode a character and take the appropriate action. The CNVCHR standard routine is first used to check for a graphic character, if the character is a header code (01H) then the routine terminates with no action. If the character is a converted graphic one then the control code decoding section is skipped. Otherwise ESCCNT is checked to see if a previous ESC character (1BH) has been received, if so control transfers to the ESC sequence processor (098FH). Otherwise the character is checked to see if it is smaller than 20H, if so control transfers to the control code processor (0914H). The character is then checked to see if it is DEL (7FH), if so control transfers to the delete routine (0AE3H). Assuming the character is displayable the cursor coordinates are taken from CSRY and CSRX and placed in register pair HL, H=Column, L=Row. These are then converted to a physical address in the VDP Name Table and the character placed there (0BE6H). The cursor column position is then incremented (0A44H) and, assuming the rightmost column has not been exceeded, the routine terminates. Otherwise the row's entry in LINTTB, the line termination table, is zeroed to indicate an extended logical line, the column number is set to 01H and a LF is performed.

```
Address... 0908H
```

This routine performs the LF operation for the CHPUT standard routine control code processor. The cursor row is incremented (0A61H) and, assuming the lowest row has not been exceeded, the routine terminates. Otherwise the screen is scrolled upwards and the lowest row erased (0A88H).

```
Address... 0914H
```

This is the control code processor for the CHPUT standard routine. The table at 092FH is searched for a match with the code and control transferred to the associated address.

```
Address... 092FH
```

This table contains the control codes, each with an associated address, recognized by the CHPUT standard routine:

CODE	TO	FUNCTION
07H	1113H	BELL, go beep
08H	0A4CH	BS, cursor left
09H	0A71H	TAB, cursor to next tab position
0AH	0908H	LF, cursor down a row
0BH	0A7FH	HOME, cursor to home
0CH	077EH	FORMFEED, clear screen and home
0DH	0A81H	CR, cursor to leftmost column
1BH	0989H	ESC, enter escape sequence
1CH	0A5BH	RIGHT, cursor right
1DH	0A4CH	LEFT, cursor left
1EH	0A57H	UP, cursor up
1FH	0A61H	DOWN, cursor down.

Address... 0953H

This table contains the ESC control codes, each with an associated address, recognized by the CHPUT standard routine:

CODE	TO	FUNCTION
6AH	077EH	ESC,"j", clear screen and home
45H	077EH	ESC,"E", clear screen and home
4BH	0AEEH	ESC,"K", clear to end of line
4AH	0B05H	ESC,"J", clear to end of screen
6CH	0AECH	ESC,"l", clear line
4CH	0AB4H	ESC,"L", insert line
4DH	0A85H	ESC,"M", delete line
59H	0986H	ESC,"Y", set cursor coordinates
41H	0A57H	ESC,"A", cursor up
42H	0A61H	ESC,"B", cursor down
43H	0A44H	ESC,"C", cursor right
44H	0A55H	ESC,"D", cursor left
48H	0A7FH	ESC,"H", cursor home
78H	0980H	ESC,"x", change cursor
79H	0983H	ESC,"y", change cursor

Address... 0980H

This routine performs the ESC,"x" operation for the CHPUT standard routine control code processor. ESCCNT is set to 01H to indicate that the next character received is a parameter.

Address... 0983H

This routine performs the ESC,"y" operation for the CHPUT standard routine control code decoder. ESCCNT is set to 02H to indicate that the next character received is a parameter.

Address... 0986H

This routine performs the ESC",Y" operation for the CHPUT standard routine control code processor. ESCCNT is set to 04H to indicate that the next character received is a parameter.

Address... 0989H

This routine performs the ESC operation for the CHPUT standard routine control code processor. ESCCNT is set to FFH to indicate that the next character received is the second control character.

Address... 098FH

This is the CHPUT standard routine ESC sequence processor. If ESCCNT contains FFH then the character is the second control character and control transfers to the control code processor (0919H) to search the ESC code table at 0953H. If ESCCNT contains 01H then the character is the single parameter of the ESC,"x" sequence. If the parameter is "4" (34H) then CSTYLE is set to 00H resulting in a block cursor. If the parameter is "5" (35H) then CSRSW is set to 00H making the cursor normally disabled. If ESCCNT contains 02H then the character is the single parameter in the ESC,"y" sequence. If the parameter is "4" (34H) then CSTYLE is set to 01H resulting in an underline cursor. If the parameter is "5" (35H) then CSRSW is set to 01H making the cursor normally enabled. If ESCCNT contains 04H then the character is the first parameter of the ESC,"Y" sequence and is the row coordinate. The parameter has 1FH subtracted and is placed in CSRY, ESCCNT is then decremented to 03H. If ESCCNT contains 03H then the character is the second parameter of the ESC,"Y" sequence and is the column coordinate. The parameter has 1FH subtracted and is placed in CSRX.

Address... 09DAH

This routine is used, by the CHGET standard routine for example, to display the cursor character when it is normally disabled. If CSRSW is non-zero the routine simply terminates with no action, otherwise the cursor is displayed (09E6H).

Address... 09E1H

This routine is used, by the CHPUT standard routine for example, to display the cursor character when it is normally enabled. If CSRSW is zero the routine simply terminates with no action. SCRMOD is checked and, if the screen is in Graphics Mode or Multicolour Mode, the routine terminates with no action. Otherwise the cursor coordinates are converted to a physical address in the VDP Name Table and the character read from that location (0BD8H) and saved in CURSAV. The character's eight byte pixel pattern is read from the VDP Character Pattern Table into the LINWRK buffer (0BA5H). The pixel pattern is then inverted, all eight bytes if CSTYLE indicates a block cursor, only the bottom three if CSTYLE indicates an underline cursor. The pixel pattern is copied back to the position for character code 255 in the VDP Character Pattern Table (0BBEH). The character code 255 is then placed at the current cursor location in the VDP Name Table (0BE6H) and the routine terminates. This method of generating the cursor character, by using character code 255, can produce curious effects under certain conditions. These can be demonstrated by executing the BASIC statement FOR N=1 TO 100: PRINT CHR$(255);:NEXT and then pressing the cursor up key.

```
Address... 0A27H
```

This routine is used, by the CHGET standard routine for example, to remove the cursor character when it is normally disabled. If CSRSW is non-zero the routine simply terminates with no action, otherwise the cursor is removed (0A33H).

```
Address... 0A2EH
```

This routine is used, by the CHPUT standard routine for example, .to remove the cursor character when it is normally enabled. If CSRSW is zero the routine simply terminates with no action. .SCRMOD is checked and, if the screen is in Graphics Mode or Multicolour Mode, the routine terminates with no action. Otherwise the cursor coordinates are converted to a physical address in the VDP Name Table and the character held in CURSAV written to that location (0BE6H).

```
Address... 0A44H
```

This routine performs the ESC,"C" operation for the CHPUT standard routine control code processor. If the cursor column coordinate is already at the rightmost column, determined by LINLEN, then the routine terminates with no action. Otherwise the column coordinate is incremented and CSRX updated. .

```
Address... 0A4CH
```

This routine performs the BS/LEFT operation for the CHPUT standard routine control code processor. The cursor column4. ROM BIOS coordinate is decremented and CSRX updated. If the column coordinate has moved beyond the leftmost position it is set to the rightmost position, from LINLEN, and an UP operation performed.

```
Address... 0A55H
```

This routine performs the ESC,"D" operation for the CHPUT standard routine control code processor. If the cursor column coordinate is already at the leftmost position then the routine terminates with no action. Otherwise the column coordinate is decremented and CSRX updated.

```
Address... 0A57H
```

This routine performs the ESC,"A" (UP) operation for the CHPUT standard routine control code processor. If the cursor row coordinate is already at the topmost position the routine terminates with no action. Otherwise the row coordinate is decremented and CSRY updated.

```
Address... 0A5BH
```

This routine performs the RIGHT operation for the CHPUT standard routine control code processor. The cursor column coordinate is incremented and CSRX updated. If the column coordinate has moved beyond the rightmost position, determined by LINLEN, it is set to the leftmost position (01H) and a DOWN operation performed.

```
Address... 0A61H
```

This routine performs the ESC,"B" (DOWN) operation for the CHPUT standard routine control code processor. If the cursor row coordinate is already at the lowest position, determined by CRTCNT and CNSDFG (0C32H), then the routine terminates with no action. Otherwise the row coordinate is incremented and CSRY updated.

```
Address... 0A71H
```

This routine performs the TAB operation for the CHPUT standard routine control code processor. ASCII spaces are output (08DFH) until CSRX is a multiple of eight plus one (BIOS columns 1, 9, 17, 25, 33).

```
Address... 0A7FH
```

This routine performs the ESC,"H" (HOME) operation for the CHPUT standard routine control code processor, CSRX and CSRY are simply set to 1,1. The ROM BIOS cursor coordinate system, while functionally identical to that used by the BASIC Interpreter, numbers the screen rows from 1 to 24 and the columns from 1 to 32/40.

```
Address... 0A81H
```

This routine performs the CR operation for the CHPUT standard routine control code processor, CSRX is simply set to 01H .

```
Address... 0A85H
```

This routine performs the ESC,"M" function for the CHPUT standard routine control code processor. A CR operation is first performed to set the cursor column coordinate to the leftmost position. The number of rows from the current row to the bottom of the screen is then determined, if this is zero the current row is simply erased (0AECH). The row count is first used to scroll up the relevant section of LINTTB, the line termination table, by one byte. It is then used to scroll up the relevant section of the screen a row at a time. Starting at the row below the current row, each line is copied from the VDP Name Table into the LINWRK buffer (0BAAH) then copied back to the Name Table one row higher (0BC3H). Finally the lowest row on the screen is erased (0AECH).

```
Address... 0AB4H
```

This routine performs the ESC,"L" operation for the CHPDT standard routine control code processor. A CR operation is first performed to set the cursor column coordinate to the leftmost position. The number of rows from the current row to the bottom of the screen is then determined, if this is zero the current row is simply erased (0AECH). The row count is first used to scroll down the relevant section of LINTTB, the line termination table, by one byte. It is then used to scroll down the relevant section of the screen a row at a time. Starting at the next to last row of the screen, each line is copied from the VDP Name Table into the LINWRK buffer (0BAAH), then copied back to the Name Table one row lower (0BC3H). Finally the current row is erased (0AECH).

```
Address... 0AE3H
```

This routine is used to perform the DEL operation for the CHPUT standard routine control code processor. A LEFT operation is first performed. If this cannot be completed, because thecursor is already at the home position, then the routine terminates with no action. Otherwise a space is written to the VDP Name Table at the cursor's physical location (0BE6H).

```
Address... 0AECH
```

This routine performs the ESC,"l" operation for the CHPUT standard routine control code processor. The cursor column coordinate is set to 01H and control drops into the ESC,"K" routine.

```
Address... 0AEEH
```

This routine performs the ESC,"K" operation for the CHPHT standard routine control code processor. The row's entry in LINTTB, the line termination table, is first made non-zero to indicate that the logical line is not extended (0C29H). The cursor coordinates are converted to a physical address (0BF2H) in the VDP Name Table and the VDP set up for writes via the SETWRT standard routine. Spaces are then written directly to the VDP Data Port until the rightmost column, determined by LINLEN, is reached.

```
Address... 0B05H
```

This routine performs the ESC,"J" operation for the CHPUT standard routine control code processor. An ESC,"K" operation is performed on successive rows, starting with the current one, until the bottom of the screen is reached.

```
Address... 0B15H
Name...... ERAFNK
Entry..... None
Exit...... None
Modifies.. AF, DE, EI
```

Standard routine to turn the function key display off. CNSDFG is first zeroed and, if the VDP is in Graphics Mode or Multicolour Mode, the routine terminates with no further action. If the VDP is in 40x24 Text Mode or 32x24 Text Mode the last row on the screen is then erased (0AECH).

```
Address... 0B26H
Name...... FNKSB
Entry..... None
Exit...... None
Modifies.. AF, BC, DE, EI
```

Standard routine to show the function key display if it is enabled. If CNSDFG is zero the routine terminates with no action, otherwise control drops into the DSPFNK standard routine..

```
Address... 0B2BH
Name...... DSPFNK
Entry..... None
Exit...... None
Modifies.. AF, BC, DE, EI
```

Standard routine to turn the function key display on. CNSDFG is set to FFH and, if the VDP is in Graphics Mode or Multicolour Mode, the routine terminates with no further action. Otherwise the cursor row coordinate is checked and, if the cursor is on the last row of the screen, a LF code (0AH) issued to the OUTDO standard routine to scroll the screen up. Register pair HL is then set to point to either the unshifted or shifted function strings in the Workspace Area depending upon whether the SHIFT key is pressed. LINLEN has four subtracted, to allow a minimum of one space between fields, and is divided by five to determine the field size for each string. Successive characters are then taken from each function string, checked for graphic headers via the CNVCHR standard routine and placed in the LINWRK buffer until the string is exhausted or the zone is filled. When all five strings are completed the LINWRK buffer is written to the last row in the VDP Name Table (0BC3H).

```
Address... 0B9CH
```

This routine is used by the function key display related standard routines. The contents of register A are placed in CNSDFG then SCRMOD tested and Flag NC returned if the screen is in Graphics Mode or Multicolour Mode.

```
Address... 0BA5H
```

This routine copies eight bytes from the VDP VRAM into the LINWRK buffer, the VRAM physical address is supplied in register pair HL.

```
Address... 0BAAH
```

This routine copies a complete row of characters, with the length determined by LINLEN, from the VDP VRAM into the LINWRK buffer. The cursor row coordinate is supplied in register L.

```
Address... 0BBEH
```

This routine copies eight bytes from the LINWRK buffer into the VDP VRAM, the VRAM physical address is supplied in registerpair HL.

```
Address... 0BC3H
```

This routine copies a complete row of characters, with the length determined by LINLEN, from the LINWRK buffer into the VDP VRAM. The cursor row coordinate is supplied in register L.

```
Address... 0BD8H
```

This routine reads a single byte from the VDP VRAM into register C. The column coordinate is supplied in register H, the row coordinate in register L.

```
Address... 0BE6H
```

This routine converts a pair of screen coordinates, the column in register H and the row in register L, into a physical address in the VDP Name Table. This address is returned in register pair HL. The row coordinate is first multiplied by thirty-two or forty, depending upon the screen mode, and added to the column coordinate. This is then added to the Name Table base address, taken from NAMBAS, to produce an initial address. Because of the variable screen width, as contained in LINLEN, an additional offset has to be added to the initial address to keep the active region roughly centered within the screen. The difference between the "true" number of charactersper row, thirty-two or forty, and the current width is halved and then rounded up to produce the left hand offset. For a UK machine, with a thirty-seven character width in 40x24 Text Mode, this will result in two unused characters on the left hand side and one on the right. The statement PRINT (41-WID)\2, where WID is any screen width, will display the left hand column offset in 40x24 Text Mode. A complete BASIC program which emulates this routine is given below:

```
10 CPR=40:NAM=BASE(0):WID=PEEK(&HF3AE)
20 SCRMD=PEEK(&HFCAF):IF SCRMD=0 THEN 40
30 CPR=32:NAM=BASE(5):WID=PEEK(&HF3AF)
40 LH=(CPR+1-WID)\2
50 ADDR=NAM+(ROW-1)*CPR+(COL-1)+LH
```

This program is designed for the ROW and COL coordinate system used by the ROM BIOS where home is 1,1. Line 50 may be simplified, by removing the "-1" factors, if the BASIC Interpreter's coordinate system is to be used.

```
Address... 0C1DH
```

This routine calculates the address of a row's entry in LINTTB, the line termination table. The row coordinate is supplied in register L and the address returned in register pair DE.

```
Address... 0C29H
```

This routine makes a row's entry in LINTTB non-zero when entered at 0C29H and zero when entered at 0C2AH. The row coordinate is supplied in register L.

```
Address... 0C32H
```

This routine returns the number of rows on the screen in register A. It will normally return twenty-four if the function key display is disabled and twenty-three if it is enabled. Note that the screen size is determined by CRTCNT and may be modified with a BASIC statement, POKE &HF3B1H,14:SCREEN 0 for example.

```
Address... 0C3CH
Name...... KEYINT
Entry..... None
Exit...... None
Modifies.. EI
```

Standard routine to process Z80 interrupts, these are generated by the VDP once every 20 ms on a UK machine. The VDP Status Register is first read and bit 7 checked to ensure that this is a frame rate interrupt, if not the routine terminates with no action. The contents of the Status Register are saved in STATFL and bit 5 checked for sprite coincidence. If the Coincidence Flag is active then the relevant entry in TRPTBL is updated (0EF1H). INTCNT, the "INTERVAL" counter, is then decremented. If this has reached zero the relevant entry in TRPTBL is updated (0EF1H) and the counter reset with the contents of INTVAL. JIFFY, the "TIME" counter, is then incremented. This counter just wraps around to zero when it overflows. MUSICF is examined to determine whether any of the three music queues generated by the "PLAY" statement are active. For each active queue the dequeueing routine (113BH) is called to fetch the next music packet and write it to the PSG. SCNCNT is then decremented to determine if a joystick and keyboard scan is required, if not the interrupt handler terminates with no further action. This counter is used to increase throughput and to minimize keybounce problems by ensuring that a scan is only carried out every three interrupts. Assuming a scan is required joystick connector 1 is selected and the two Trigger bits read (120CH), followed by the two Trigger bits from joystick connector 2 (120CH) and the SPACE key from row 8 of the keyboard (1226H). These five

inputs, which are all related to the "STRIG" statement, are combined into a single byte where 0=Pressed, 1=Not pressed:

7	6	5	4	3	2	1	0
Joy 2	Joy 2	Joy 1	Joy 1	0	0	0	Space
Trg.B	Trg.A	Trg.B	Trg.A				

Figure 35: "STRIG" Inputs.

This reading is compared with the previous one, held in TRGFLG, to produce an active transition byte and TRGFLG is updated with the new reading. The active transition byte is normally zero but contains a 1 in each position where a transition from unpressed to pressed has occurred. This active transition byte is shifted out bit by bit and the relevant entry in TRPTBL updated (0EF1H) for each active device. A complete scan of the keyboard matrix is then performed to identify new key depressions, any found are translated into key codes and placed in KEYBUF (0D12H). If KEYBUF is found to be empty at the end of this process REPCNT is decremented to see whether the auto-repeat delay has expired, if not the routine terminates. If the delay period has expired REPCNT is reset with the fast repeat value (60 ms), the OLDKEY keyboard map is reinitialized and the keyboard scanned again (0D4EH). Any keys which are continuously pressed will show up as new transitions during this scan. Note that keys will only auto-repeat while an application program keeps KEYBUF empty by reading characters. The interrupt handler then terminates.

```
Address... 0D12H
```

This routine performs a complete scan of all eleven rows of the keyboard matrix for the interrupt handler. Each of the eleven rows is read in via the PPI and placed in ascending ' order in NEWKEY. ENSTOP is then checked to see if warm starts are enabled. If its contents are non-zero and the keys CODE, GRAPH, CTRL and SHIFT are pressed control transfers to the BASIC Interpreter (409BH) via the CALBAS standard routine. This facility is useful as even a machine code program can be terminated as long as the interrupt handler is running. The contents of NEWKEY are compared with the previous scan contained in OLDKEY. If any change at all has occurred REPCNT is loaded with the initial auto-repeat delay (780 ms). Each row 1, reading from NEWKEY is then compared with the previous one, held in OLDKEY, to produce an active transition byte and OLDKEY is updated with the new reading. The active transition byte is normally zero but contains a 1 in each position where a transition from unpressed to pressed has occurred. If the row contains any transitions these are decoded and placed in KEYBUF as key codes (0D89H). When all eleven rows have been completed the routine checks whether there are any characters in KEYBUF, by subtracting GETPNT from PUTPNT, and terminates.

```
Address... 0D6AH
Name...... CHSNS
Entry..... None
Exit...... Flag NZ if characters in KEYBUF
Modifies.. AF, EI
```

Standard routine to check if any keyboard characters are ready. If the screen is in Graphics Mode or Multicolour Mode then GETPNT is subtracted from PUTPNT (0D62H) and the routine terminates. If the screen is in 40x24 Text Mode or 32x24 Text Mode the state of the SHIFT key is also examined and the function key display updated, via the DSPFNK standard routine, if it has changed.

```
Address... 0D89H
```

This routine converts each active bit in a keyboard row transition byte into a key code. A bit is first converted into a key number determined by its position in the keyboard matrix:

7 (07H)	6 (06H)	5 (05H)	4 (04H)	3 (03H)	2 (02H)	1 (01H)	0 (00H)	Row 0
; (0FH)	] (0EH)	[(0DH)	\ (0CH)	= (0BH)	- (0AH)	9 (09H)	8 (08H)	Row 1
B (17H)	A (16H)	œ (15H)	/ (14H)	. (13H)	, (12H)	` (11H)	' (10H)	Row 2
J (1FH)	I (1EH)	H (1DH)	G (1CH)	F (1BH)	E (1AH)	D (19H)	C (18H)	Row3
R (27H)	Q (26H)	P (25H)	O (24H)	N (23H)	M (22H)	L (21H)	K (20H)	Row 4
Z (2FH)	Y (2EH)	X (2DH)	W (2CH)	V (2BH)	U (2AH)	T (29H)	S (28H)	Row 5
F3 (37H)	F2 (36H)	F1 (35H)	CODE (34H)	CAP (33H)	GRAPH (32H)	CTRL (31H)	SHIFT (30H)	Row 6
CR (3FH)	SEL (3EH)	BS (3DH)	STOP (3CH)	TAB (3BH)	ESC (3AH)	F5 (39H)	F4 (38H)	Row 7
RIGHT (47H)	DOWN (46H)	UP (45H)	LEFT (44H)	DEL (43H)	INS (42H)	HOME (41H)	SPACE (40H)	Row 8
4 (4FH)	3 (4EH)	2 (4DH)	1 (4CH)	0 (4BH)	(4AH)	(49H)	(48H)	Row 9
. (57H)	, (56H)	- (55H)	9 (54H)	8 (53H)	7 (52H)	6 (51H)	5 (50H)	Row 10
7	6	5	4	3	2	1	0	Column

Figure 36: Key Numbers.

The key number is then con|ed into a key code and placed in KEYBUF (1021H). When all eight possible bits have been processed the routine terminates.

```
Address... 0DA5H
```

This table contains the key codes of key numbers 00H to 2FH for various combinations of the control keys. A zero entry in the table means that no key code will be produced when that key is pressed:

	7	6	5	4	3	2	1	0	
NORMAL	37H	36H	35H	34H	33H	32H	31H	30H	Row 0
	3BH	5DH	5BH	5CH	3DH	2DH	39H	38H	Row 1
	62H	61H	9CH	2FH	2EH	2CH	60H	27H	Row 2
	6AH	69H	68H	67H	66H	65H	64H	63H	Row 3
	72H	71H	70H	6FH	6EH	6DH	6CH	6BH	Row 4
	7AH	79H	78H	77H	76H	75H	74H	73H	Row 5

	7	6	5	4	3	2	1	0	
SHIFT	26H	5EH	25H	24H	23H	40H	21H	29H	Row 0
	3AH	7DH	7BH	7CH	2BH	5FH	28H	2AH	Row 1
	42H	41H	9CH	3FH	3EH	3CH	7EH	22H	Row 2
	4AH	49H	48H	47H	46H	45H	44H	43H	Row 3
	52H	51H	50H	4FH	4EH	4DH	4CH	4BH	Row 4
	5AH	59H	58H	57H	56H	55H	54H	53H	Row 5

	7	6	5	4	3	2	1	0	
GRAPH	FBH	F4H	BDH	EFH	BAH	ABH	ACH	09H	Row 0
	06H	0DH	01H	1EH	F1H	17H	07H	ECH	Row 1
	11H	C4H	9CH	1DH	F2H	F3H	BBH	05H	Row 2
	C6H	DCH	13H	15H	14H	CDH	C7H	BCH	Row 3
	18H	CCH	DBH	C2H	1BH	0BH	C8H	DDH	Row 4
	0FH	19H	1CH	CFH	1AH	C0H	12H	D2H	Row 5

	7	6	5	4	3	2	1	0	
SHIFT GRAPH	00H	F5H	00H	00H	FCH	FDH	00H	0AH	Row 0
	04H	0EH	02H	16H	F0H	1FH	08H	00H	Row 1
	00H	FEH	9CH	F6H	AFH	AEH	F7H	03H	Row 2
	CAH	DFH	D6H	10H	D4H	CEH	C1H	FAH	Row 3
	A9H	CBH	D7H	C3H	D3H	0CH	C9H	DEH	Row 4
	F8H	AAH	F9H	D0H	D5H	C5H	00H	D1H	Row 5

	7	6	5	4	3	2	1	0	
CODE	E1H	E0H	98H	9BH	BFH	D9H	9FH	EBH	Row 0
	B7H	DAH	EDH	9CH	E9H	EEH	87H	E7H	Row 1
	97H	84H	9CH	A7H	A6H	86H	E5H	B9H	Row 2
	91H	A1H	B1H	81H	94H	8CH	8BH	8DH	Row 3
	93H	83H	A3H	A2H	A4H	E6H	B5H	B3H	Row 4
	85H	A0H	8AH	88H	95H	82H	96H	89H	Row 5

	7	6	5	4	3	2	1	0	
SHIFT CODE	00H	00H	9DH	9CH	BEH	9EH	ADH	D8H	Row 0
	B6H	EAH	E8H	00H	00H	00H	80H	E2H	Row 1
	00H	8EH	9CH	A8H	00H	8FH	E4H	B8H	Row 2
	92H	00H	B0H	9AH	99H	00H	00H	00H	Row 3
	00H	00H	E3H	00H	A5H	00H	B4H	B2H	Row 4
	00H	00H	00H	00H	00H	90H	00H	00H	Row 5

```
Address... 0EC5H
```

Control transfers to this routine, from 0FC3H, to complete decoding of the five function keys. The relevant entry in FNKFLG is first checked to determine whether the key is associated with an "ON KEY GOSUB" statement. If so, and provided that CURLIN shows the BASIC Interpreter to be in program mode, the relevant entry in TRPTBL is updated (0EF1H) and the routine terminates. If the key is not tied to an "ON KEY GOSUB" statement, or if the Interpreter is in direct mode, the string of characters associated with the function key is returned instead. The key number is multiplied by sixteen, as each string is sixteen characters long, and added to the starting address of the function key strings in the Workspace Area. Sequential characters are then taken from the string and placed in KEYBUF (0F55H) until the zero byte terminator is reached.

```
Address... 0EF1H
```

This routine is used to update a device's entry in TRPTBL when it has produced a BASIC program interrupt. On entry register pair HL points to the device's status byte in the table. Bit 0 of the status byte is checked first, if the device is not "ON" then the routine terminates with no action. Bit 2, the event flag, is then checked. If this is already set then the routine terminates, otherwise it is set to indicate that an event has occurred. Bit 1, the "STOP" flag, is then checked. If the device is stopped then the routine terminates with no further action. Otherwise ONGSBF is incremented to signal to the Interpreter Runloop that the event should now be processed.

```
Address... 0F06H
```

This section of the key decoder processes the HOME key only. The state of the SHIFT key is determined via row 6 of NEWKEY and the key code for HOME (0BH) or CLS (0CH) placed in KEYBUF (0F55H) accordingly.

```
Address... 0F10H
```

This section of the keyboard decoder processes key numbers 30H to 57H apart from the CAP, F1 to F5, STOP and HOME keys. The key number is simply used to look up the key code in the table at 1033H and this is then placed in KEYBUF (0F55H).

```
Address... 0F1FH
```

This section of the keyboard decoder processes the DEAD key found on European MSX machines. On UK machines the key in row 2, column 5 always generates the pound key code (9CH) shown in the table at 0DA5H. On European machines this table will have the key code FFH in the same locations. This key code only serves as a flag to indicate that the next key pressed, if it is a vowel, should be modified to produce an accented graphics character. The state of the SHIFT and CODE keys is determined via row 6 of NEWKEY and one of the following placed in KANAST: 01H=DEAD, 02H=DEAD+SHIFT, 03H=DEAD+CODE, 04H=DEAD+SHIFT+CODE.

```
Address... 0F36H
```

This section of the keyboard decoder processes the CAP key. The current state of CAPST is in|ed and control drops into the CHGCAP standard routine.

```
Address... 0F3DH
Name...... CHGCAP
Entry..... A=ON/OFF Switch
Exit...... None
Modifies.. AF
```

Standard routine to turn the Caps Lock LED on or off as determined by the contents of register A: 00H=On, NZ=Off. The LED is modified using the bit set/reset facility of the PPI Mode Port. As CAPST is not changed this routine does not affect the characters produced by the keyboard.

```
Address... 0F46H
```

This section of the keyboard decoder processes the STOP key. The state of the CTRL key is determined via row 6 of NEWKEY and the key code for STOP (04H) or CTRL/STOP (03H) produced as appropriate. If the CTRL/STOP code is produced it is copied to INTFLG, for use by the ISCNTC standard routine, and then placed in KEYBUF (0F55H). If the STOP code is produced it is also copied to INTFLG but is not placed in KEYBUF, instead only a click is generated (0F64H). This means that an application program cannot read the STOP key code via the ROM BIOS standard routines.

```
Address... 0F55H
```

This section of the keyboard decoder places a key code in KEYBUF and generates an audible click. The correct address in the keyboard buffer is first taken from PUTPNT and the code placed there. The address is then incremented (105BH). If it has wrapped round and caught up with GETPNT then the routine terminates with no further action as the keyboard buffer is full. Otherwise PUTPNT is updated with the new address. CLIKSW and CLIKFL are then both checked to determine whether a click is required. CLIKSW is a general enable/disable switch while CLIKFL is used to prevent multiple clicks when the function keys are pressed. Assuming a click is required the Key Click output is set via the PPI Mode Port and, after a delay of 50 æs, control drops into the CHGSND standard routine.

```
Address... 0F7AH
Name...... CHGSND
Entry..... A=ON/OFF Switch
Exit...... None
Modifies.. AF
```

Standard routine to set or reset the Key Click output via the PPI Mode Port: 00H=Reset, NZ=Set. This audio output is AC coupled so absolute polarities should not be taken too seriously.

```
Address... 0F83H
```

This section of the keyboard decoder processes key numbers 00H to 2FH. The state of the SHIFT, GRAPH and CODE keys is determined via row 6 of NEWKEY and combined with the key number to form a look-up address into the table at 0DA5H. The key code is then taken from the table. If it is zero the routine terminates with no further action, if it is FFH control transfers to the DEAD key processor (0F1FH). If the code is in the range 40H to 5FH or 60H to 7FH and the CTRL key is pressed then the corresponding control code is placed in KEYBUF (0F55H). If the code is in the range 01H to 1FH then a graphic header code (01H) is first placed in KEYBUF (0F55H) followed by the code with 40H added. If the code is in the range 61H to 7BH and CAPST indicates that caps lock is on then it is con|ed to upper case by subtracting 20H. Assuming that KANAST contains zero, as it always will on UK machines, then the key code is placed in KEYBUF (0F55H) and the routine terminates. On European MSX machines, with a DEAD key instead of a pound key, then the key codes corresponding to the vowels a, e, i, o, u may be further modified into graphics codes.

```
Address... 0FC3H
```

This section of the keyboard decoder processes the five function keys. The state of the SHIFT key is examined via row 6 of NEWKEY and five added to the key number if it is pressed. Control then transfers to 0EC5H to complete processing.

```
Address... 1021H
```

This routine searches the table at 1B97H to determine which group of keys the key number supplied in register C belongs to. The associated address is then taken from the table and control transferred to that section of the keyboard decoder. Note that the table itself is actually patched into the middle of the OUTDO standard routine as a result of the modifications made to the Japanese ROM.

```
Address... 1033H
```

This table contains the key codes of key numbers 30H to 57H other than the special keys CAP, F1 to F5, STOP and HOME. A zero entry in the table means that no key code will be produced when that key is pressed:

00H	00H	00H	00H	00H	00H	00H	00H	Row 6
0DH	18H	08H	00H	09H	1BH	00H	00H	Row 7
1CH	1FH	1EH	1DH	7FH	12H	0CH	20H	Row 8
34H	33H	32H	31H	30H	00H	00H	00H	Row 9
2EH	2CH	2DH	39H	38H	37H	36H	35H	Row 10
7	6	5	4	3	2	1	0	Column

```
Address... 105BH
```

This routine simply zeroes KANAST and then transfers control to 10C2H.

```
Address... 1061H
```

This table contains the graphics characters which replace the vowels a, e, i, o, u on European machines.

```
Address... 10C2H
```

This routine increments the keyboard buffer pointer, either PUTPNT or GETPNT, supplied in register pair HL. If the pointer then exceeds the end of the keyboard buffer it is wrapped back to the beginning.

```
Address... 10CBH
Name...... CHGET
Entry..... None
Exit...... A=Character from keyboard
Modifies.. AF, EI
```

Standard routine to fetch a character from the keyboard buffer. The buffer is first checked to see if already contains a character (0D6AH). If not the cursor is turned on (09DAH), the buffer checked repeatedly until a character appears (0D6AH) and then the cursor turned off (0A27H). The character is taken from the buffer using GETPNT which is then incremented (10C2H).

```
Address... 10F9H
Name...... CKCNTC
Entry..... None
Exit...... None
Modifies.. AF, EI
```

Standard routine to check whether the CTRL-STOP or STOP keys have been pressed. It is used by the BASIC Interpreter inside processor-intensive statements, such as "WAIT" and "CIRCLE", to check for program termination. Register pair HL is first zeroed and then control transferred to the ISCNTC standard routine. When the Interpreter is running register pair HL normally contains the address of the current character in the BASIC program text. If ISCNTC is CTRL-STOP terminated this address will be placed in OLDTXT by the "STOP" statement handler (63E6H) for use by a later "CONT" statement. Zeroing register pair HL beforehand signals to the "CONT" handler that termination occurred inside a statement and it will issue a "Can't CONTINUE" error if continuation is attempted.

```
Address... 1102H
Name...... WRTPSG
Entry..... A=Register number, E=Data byte
Exit...... None
Modifies.. EI
```

Standard routine to write a data byte to any of the sixteen PSG registers. The register selection number is written to the PSG Address Port and the data byte written to the PSG Data Write Port.

```
Address... 110EH
Name...... RDPSG
Entry..... A=Register number
Exit...... A=Data byte read from PSG
Modifies.. A
```

Standard routine to read a data byte from any of the sixteen PSG registers. The register selection number is written to the PSG Address Port and the data byte read from the PSG Data Read Port.

```
Address... 1113H
Name...... BEEP
Entry..... None
Exit...... None
Modifies.. AF, BC, E, EI
```

Standard routine to produce a beep via the PSG. Channel A is set to produce a tone of 1316Hz then enabled with an amplitude of seven. After a delay of 40 ms control transfers to the GICINI standard routine to reinitialize the PSG.

```
Address... 113BH
```

This routine is used by the interrupt handler to service a music queue. As there are three of these, each feeding a PSG channel, the queue to be serviced is specified by supplying its number in register A: 0=VOICAQ, 1=VOICBQ and 2=VOICCQ. Each string in a "PLAY" statement is translated into a series of data packets by the BASIC Interpreter. These are placed in the appropriate queue followed by an end of data byte (FFH). The task of dequeueing the packets, decoding them and setting the PSG is left to the interrupt handler. The Interpreter is thus free to proceed immediately to the next statement without having to wait for notes to finish. The first two bytes of any packet specify its byte count and duration. The three most significant bits of the first byte specify the number of bytes following the header in the packet. The remainder of the header specifies the event duration in 20 ms units. This duration count determines how long it will be before the next packet is read from the queue.

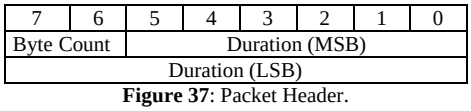


Figure 37: Packet Header.

The packet header may be followed by zero or more blocks, in any order, containing frequency or amplitude information:

7	6	5	4	3	2	1	0
0	0	x	x	Frequency (MSB)			
Frequency (LSB)							
Frequency Block							

7	6	5	4	3	2	1	0
X	1	x	x	x	x	x	x
Envelope Frequency (MSB)							
Envelope Frequency (LSB)							
Envelope Block							

7	6	5	4	3	2	1	0
1	x	x	Mode	Amplitude/Shape			
Amplitude Block							

Figure 38: Packet Block Types.

The routine first locates the current duration counter in the relevant voice buffer (VCBA, VCBB or VCBC) via the GETVCP standard routine and decrements it. If the counter has reached zero then the next packet must be read from the queue, otherwise the routine terminates. The queue number is placed in QUEUEN and a byte read fromthe queue (11E2H). This is then checked to see if it is the endof data mark (FFH), if so the queue terminates (11B0H). Otherwise the byte count is placed in register C and the duration MSB in the relevant voice buffer. The second byte is read (11E2H) and the duration LSB placed in the relevant voice buffer. The byte count is then examined, if there are no bytes to follow the packet header the routine terminates. Otherwise successive bytes are read from the queue, and the appropriate action taken, until the byte count is exhausted. If a frequency block is found then a second byte is read and both bytes written to PSG Registers 0 and 1, 2 and 3 or 4 and 5 depending on the queue number. If an amplitude block is found the Amplitude and Mode bits are written to PSG Registers 8, 9 or 10 depending on the queue number. If the Mode bit is 1, selecting modulated rather than fixed amplitude, then the byte is also written to PSG Register 13 to set the envelope shape. If an envelope block is found, or if bit 6 of an amplitude block is set, then a further two bytes are read from the queue and written to PSG Registers 11 and 12.

```
Address... 11B0H
```

This routine is used when an end of data mark (FFH) is found in one of the three music queues. An amplitude value of zero is written to PSG Register 8 9 or 10, depending on the queue number, to shut the channel down. The channel's bit in MUSICF is then reset and control drops into the STRTMS standardroutine.

```
Address... 11C4H
Name...... STRTMS
Entry..... None
Exit...... None
Modifies.. AF, HL
```

Standard routine used by the "PLAY" statement handler to initiate music dequeueing by the interrupt handler. MUSICF is first examined, if any channels are already running the routine terminates with no action. PLYCNT is then decremented, if thereare no more "PLAY" strings queued up the routine terminates. Otherwise the three duration counters, in VCBA, VCBB and VCBC, are set to 0001H, so that the first packet of the new group will be dequeued at the next interrupt, and MUSICF is set to 07H to enable all three channels.

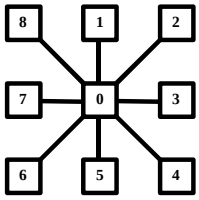
```
Address... 11E2H
```

This routine loads register A with the current queue number, from QUEUEN, and then reads a byte from that queue (14ADH).

```
Address... 11EEH
Name...... GTSTCK
Entry..... A=Joystick ID (0, 1 or 2)
Exit...... A=Joystick position code
Modifies.. AF, B, DE, HL, EI
```

Standard routine to read the position of a joystick or the four cursor keys. If the supplied ID is zero the state of the cursor keys is read via PPI Port B (1226H) and con|ed to a position code using the look-up table at 1243H. Otherwise

joystick connector 1 or 2 is read (120CH) and the four direction bits con|ed to a position code using the look-up table at 1233H. The returned position codes are:




```
Address... 120CH
```

This routine reads the joystick connector specified by the contents of register A: 0=Connector 1, 1=Connector 2. The current contents of PSG Register 15 are read in then written back with the Joystick Select bit appropriately set. PSG Register 14 is then read into register A (110CH) and the routine terminates.

```
Address... 1226H
```

This routine reads row 8 of the keyboard matrix. The current contents of PPI Port C are read in then written back with the four Keyboard Row Select bits set for row 8. The column inputs are then read into register A from PPI Port B.

```
Address... 1253H
Name...... GTTRIG
Entry..... A=Trigger ID (0, 1, 2, 3 or 4)
Exit...... A=Status code
Modifies.. AF, BC, EI
```

Standard routine to check the joystick trigger or space key status. If the supplied ID is zero row 8 of the keyboard matrix is read (1226H) and con|ed to a status code. Otherwise joystick connector 1 or 2 is read (120CH) and con|ed to a status code. The selection IDs are:

```
0=SPACE KEY
1=JOY 1, TRIGGER A
2=JOY 2, TRIGGER A
3=JOY 1, TRIGGER B
4=JOY 2, TRIGGER B
```

The value returned is FFH if the relevant trigger is pressed and zero otherwise.

```
Address... 1273H
Name...... GTPDL
Entry..... A=Paddle ID (1 to 12)
Exit...... A=Paddle value (0 to 255)
Modifies.. AF, BC, DE, EI
```

Standard routine to read the value of any paddle attached to a joystick connector. Each of the six input lines (four direction plus two triggers) per connector can support a paddle so twelve are possible altogether. The paddles attached to joystick connector 1 have entry identifiers 1, 3, 5, 7, 9 and 11. Those attached to joystick connector 2 have entry identifiers 2, 4, 6, 8, 10 and 12. Each paddle is basically a one-shot pulse generator, the length of the pulse being controlled by a variable resistor. A start pulse is issued to the specified joystick connector via PSG Register 15. A count is then kept of how many times PSG Register 14 has to be read until the relevant input times out. Each unit increment represents an approximate period of 12 æs on an MSX machine with one wait state.

```
Address... 12ACH
Name...... GTPAD
Entry..... A=Function code (0 to 7)
Exit...... A=Status or value
Modifies.. AF, BC, DE, HL, EI
```

Standard routine to access a touchpad attached to either of the joystick connectors. Available functions codes for joystick connector 1 are:

```
0=Return Activity Status
1=Return "X" coordinate
2=Return "Y" coordinate
3=Return Switch Status
```

Function codes 4 to 7 have the same effect with respect to joystick connector 2. The Activity Status function returns FFH if the Touchpad is being touched and zero otherwise. The Switch Status function returns FFH if the switch is being pressed and zero otherwise. The two coordinate request functions return the coordinates of the last location touched. These coordinates are actually stored in the Workspace Area variables PADX and PADY when a call with function code 0 or 4 detects activity. Note that these variables are shared by both joystick connectors.

```
Address... 1384H
Name...... STMOTR
Entry..... A=Motor ON/OFF code
Exit...... None
Modifies.. AF
```

Standard routine to turn the cassette motor relay on or off via PPI Port C: 00H=Off, 01H=On, FFH=Reverse current state.

```
Address... 1398H
Name...... NMI
Entry..... None
Exit...... None
Modifies.. None
```

Standard routine to process a Z80 Non Maskable Interrupt, simply returns on a standard MSX machine.

```
Address... 139DH
Name...... INIFNK
Entry..... None
Exit...... None
Modifies.. BC, DE, HL
```

Standard routine to initialize the ten function key strings to their power-up values. The one hundred and sixty bytes of data commencing at 13A9H are copied to the FNKSTR buffer in the Workspace Area.

```
Address... 13A9H
```

This area contains the power-up strings for the ten function keys. Each string is sixteen characters long, unused positions contain zeroes:

F1 to F5	F6 to F10
color	color 15,4,4 CR
auto	cload"
goto	cont CR
list	list. CR UP UP
run	run CLS CR

```
Address... 1449H
Name...... RDVDP
Entry..... None
Exit...... A=VDP Status Register contents
Modifies.. A
```

Standard routine to input the contents of the VDP Status Register by reading the Command Port. Note that reading the VDP Status Register will clear the associated flags and may affect the interrupt handler.

```
Address... 144CH
Name...... RSLREG
Entry..... None
Exit...... A=Primary Slot Register contents
Modifies.. A
```

Standard routine to input the contents of the Primary slotRegister by reading PPI Port A.

```
Address... 144FH
Name..... WSLREG
Entry..... A=Value to write
Exit..... None
Modifies.. None
```

Standard routine to set the Primary Slot Register by writing to PPI Port A.

```
Address... 1452H
Name..... SNSMAT
Entry..... A=Keyboard row number
Exit..... A=Column data of keyboard row
Modifies.. AF, C, EI
```

Standard routine to read a complete row of the keyboard matrix. PPI Port C is read in then written back with the row number occupying the four Keyboard Row Select bits. PPI Port B is then read into register A to return the eight column inputs. The four miscellaneous control outputs of PPI Port C are unaffected by this routine.

```
Address... 145FH
Name..... ISFLIO
Entry..... None
Exit..... Flag NZ if file I/O active
Modifies.. AF
```

Standard routine to check whether the BASIC Interpreter is currently directing its input or output via an I/O buffer. This is determined by examining PTRFIL. It is normally zero but will contain a buffer FCB (File Control Block) address while statements such as "PRINT#1", "INPUT#1", etc. are being executed by the Interpreter.

```
Address... 146AH
Name..... DCOMPR
Entry..... HL, DE
Exit..... Flag NC if HL>DE, Flag Z if HL=DE, Flag C if HL<DE
Modifies.. AF
```

Standard routine used by the BASIC Interpreter to check the elative values of register pairs HL and DE.

```
Address... 1470H
Name..... GETVCP
Entry..... A=Voice number (0, 1, 2)
Exit..... HL=Address in voice buffer
Modifies.. AF, HL
```

Standard routine to return the address of byte 2 in the specified voice buffer (VCBA, VCBB or VCBC).

```
Address... 1474H
Name..... GETVC2
Entry..... L=Byte number (0 to 36)
Exit..... HL=Address in voice buffer
Modifies.. AF, HL
```

Standard routine to return the address of any byte in the voice buffer (VCBA, VCBB or VCBC) specified by the voice number in VOICEN.

```
Address... 148AH
Name..... PHYDIO
Entry..... None
Exit..... None
Modifies.. None
```

Standard routine for use by Disk BASIC, simply returns on standard MSX machines.

```
Address... 148EH
Name...... FORMAT
Entry..... None
Exit...... None
Modifies.. None
```

Standard routine for use by Disk BASIC, simply returns on standard MSX machines.

```
Address... 1492H
Name...... PUTQ
Entry..... A=Queue number, E=Data byte
Exit...... Flag Z if queue full
Modifies.. AF, BC, HL
```

Standard routine to place a data byte in one of the three music queues. The queue's get and put positions are first taken from QUETAB (14FAH). The put position is temporarily incremented and compared with the get position, if they are equal the routine terminates as the queue is full. Otherwise the queue's address is taken from QUETAB and the put position added to it. The data byte is placed at this location in the queue, the put position is incremented and the routine terminates. Note that the music queues are circular, if the get or put pointers reach the last position in the queue they wrap around back to the start.

```
Address... 14ADH
```

This routine is used by the interrupt handler to read a byte from one of the three music queues. The queue number is supplied in register A, the data byte is returned in register A and the routine returns Flag Z if the queue is empty. The queue's get and put positions are first taken from QUETAB (14FAH). If the putback flag is active then the data byte is taken from QUEBAK and the routine terminates (14D1H), this facility is unused in the current versions of the MSX ROM. The put position is then compared with the get position, if they are equal the routine terminates as the queue is empty. Otherwise the queue's address is taken from QUETAB and the get position added to it. The data byte is read from this location in the queue, the get position is incremented and the routine terminates.

```
Address... 14DAH
```

This routine is used by the GICINI standard routine to initialize a queue's control block in QUETAB. The control block is first located in QUETAB (1504H) and the put, get and putback bytes zeroed. The size byte is set from register B and the queue address from register pair DE.

```
Address... 14EBH
Name...... LFTQ
Entry..... A=Queue number
Exit...... HL=Free space left in queue
Modifies.. AF, BC, HL
```

Standard routine to return the number of free bytes left in a music queue. The queue's get and put positions are taken from QUETAB (14FAH) and the free space determined by subtracting put from get.

```
Address... 14FAH
```

This routine returns a queue's control parameters from QUETAB, the queue number is supplied in register A. The control block is first located in QUETAB (1504H), the put position is then placed in register B, the get position in register C and the putback flag in register A.

```
Address... 1504H
```

This routine locates a queue's control block in QUETAB. The queue number is supplied in register A and the control block address returned in register pair HL. The queue number is simply multiplied by six, as there are six bytes per block, and added to the address of QUETAB as held in QUEUES.

```
Address... 1510H
Name...... GRPPRT
Entry..... A=Character
Exit...... None
Modifies.. EI
```

Standard routine to display a character on the screen in either Graphics Mode or Multicolour Mode, it is functionally equivalent to the CHPUT standard routine The CNVCHR standard routine is first used to check for a graphic character, if the character is a header code (01H) then the routine terminates with no action. If the character is a con|ed graphic one then the control code decoding section is skipped. Otherwise the character is checked to see if it is a control code. Only the CR code (0DH) is recognized (157EH), all other characters smaller than 20H are ignored. Assuming the character is displayable its eight byte pixel pattern is copied from the ROM character set into the PATWRK buffer (0752H) and FORCLR copied to ATRBYT to set its colour. The current graphics coordinates are then taken from GRPACX and GRPACY and used to set the current pixel physical address via the SCALXY and MAPXYC standard routines. The eight byte pattern in PATWRK is processed a byte at a time. At the start of each byte the current pixel physical address is obtained via the FETCHC standard routine and saved. The eight bits are then examined in turn. If the bit is a 1 the associated pixel is set by the SETC standard routine, if it is a 0 no action is taken. After each bit the current pixel physical address is moved right (16ACH). When the byte is finished, or the right hand edge of the screen is reached, the initial current pixel physical address is restored and moved down one position by the TDOWNC standard routine. When the pattern is complete, or the bottom of the screen has been reached, GRPACX is updated. In Graphics Mode its value is increased by eight, in Multicolour Mode by thirty-two. If GRPACX then exceeds 255, the right hand edge of the screen, a CR operation is performed (157EH).

```
Address... 157EH
```

This routine performs the CR operation for the GRPPRT standard routine, this code functions as a combined CR,LF. GRPACX is zeroed and eight or thirty-two, depending on the screen mode, added to GRPACY. If GRPACY then exceeds 191, the bottom of the screen, it is set to zero. GRPACX and GRPACY may be manipulated directly by an application program to compensate for the limited number of control functions available.

```
Address... 1599B
Name...... SCALXY
Entry..... BC=X coordinate, DE=Y coordinate
Exit...... Flag NC if clipped
Modifies.. AF
```

Standard routine to clip a pair of graphics coordinates if necessary. The BASIC Interpreter can produce coordinates in the range -32768 to +32767 even though this far exceeds the actual screen size. This routine modifies excessive coordinate values to fit within the physically realizable range. If the X coordinate is greater than 255 it is set to 255, if the Y coordinate is greater than 191 it is set to 191. If either coordinate is negative (greater than 7FFFH) it is set to zero. Finally if the screen is in Multicolour Mode both coordinates are divided by four as required by the MAPXYC standard routine.

```
Address... 15D9H
```

This routine is used to check the current screen mode, it returns Flag Z if the screen is in Graphics Mode.

```
Address... 15DFH
Name...... MAPXYC
Entry..... BC=X coordinate, DE=Y coordinate
Exit...... None
Modifies.. AF, D, HL
```

Standard routine to con| a graphics coordinate pair into the current pixel physical address. The location in the Character Pattern Table of the byte containing the pixel is laced in CLOC. The bit mask identifying the pixel within that byte is placed in CMASK. Slightly different conversion methods are used for Multicolour Mode and Graphics Mode, equivalent programs in BASIC are:

Multicolour Mode

```
10 INPUT"X,Y Coordinates";X,Y
20 X=X\4:Y-Y\4
30 A=(Y\8)*256+(Y AND 7)+(X*4 AND &HF8)
40 PRINT"ADDR=";HEX$(BASE(17)+A);"H ";
50 IF X MOD 2=0 THEN MS="11110000" ELSE MS="00001111"
60 PRINT"MASK=";M$
70 GOTO 10
```

Graphics Mode

```
10 INPUT"X,Y Coordinates";X,Y
20 A=(Y\8)*256+(Y AND 7)+(X AND &HF8)
30 PRINT"ADDR=";HEX$(Base(12)+A);"H ";
40 RESTORE 100
50 FOR N=0 TO (X AND 7):READ M$: NEXT N
60 PRINT"MASK=";M$
70 GOTO 10
100 DATA 10000000
110 DATA 01000000
120 DATA 00100000
130 DATA 00010000
140 DATA 00001000
150 DATA 00000100
160 DATA 00000010
170 DATA 00000001
```

The allowable input range for both programs is X=0 to 255 and Y=0 to 191. The data statements in the Graphics Mode program correspond to the eight byte mask table commencing at 160BH in the MSX ROM. Line 20 in the Multicolour Mode program actually corresponds to the division by four in the SCALXY standard routine. It is included to make the coordinate system consistent for both programs.

```
Address... 1639H
Name...... FETCHC
Entry..... None
Exit...... A=CMASK, HL=CLOC
Modifies.. A, HL
```

Standard routine to return the current pixel physical address, register pair HL is loaded from CLOC and register A from CMASK.

```
Address... 1640H
Name...... STOREC
Entry..... A=CMASK, HL=CLOC
Exit...... None
Modifies.. None
```

Standard routine to set the current pixel physical address, register pair HL is copied to CLOC and register A is copied to CMASK.

```
Address... 1647H
Name...... READC
Entry..... None
Exit...... A=Colour code of current pixel
Modifies.. AF, EI
```

Standard routine to return the colour of the current pixel. The VRAM physical address is first obtained via the FETCHC standard routine. If the screen is in Graphics Mode the byte pointed to by CLOC is read from the Character Pattern Table via the RDVRM standard routine. The required bit is then isolated by CMASK and used to select either the upper or lower four bits of the corresponding entry in the Colour Table. If the screen is in Multicolour Mode the byte pointed to by CLOC is read from the Character Pattern Table via the RDVRM standard routine. CMASK is then used to select either the upper or lower four bits of this byte. The value returned in either case will be a normal VDP colour code from zero to fifteen.

```
Address... 1676H
Name...... SETATR
Entry..... A=Colour code
Exit...... Flag C if illegal code
Modifies.. Flags
```

Standard routine to set the graphics ink colour used by the SETC and NSETCX standard routines. The colour code, from zero to fifteen, is simply placed in ATRBYT.

```
Address... 167EH
Name...... SETC
Entry..... None
Exit...... None
Modifies.. AF, EI
```

Standard routine to set the current pixel to any colour, the colour code is taken from ATRBYT. The pixel's VRAM physical address is first obtained via the FETCHC standard routine. In Graphics Mode both the Character Pattern Table and Colour Table are then modified (186CH). In Multicolour Mode the byte pointed to by CLOC is read from the Character Pattern Table by the RDVRM standard routine. The contents of ATRBYT are then placed in the upper or lower four bits, as determined by CMASK, and the byte written back via the WRTVRM standard routine

```
Address... 16ACH
```

This routine moves the current pixel physical address one position right. If the right hand edge of the screen is exceeded it returns with Flag C and the physical address is unchanged. In Graphics Mode CMASK is first shifted one bit right, if the pixel still remains within the byte the routine terminates. If CLOC is at the rightmost character cell (LSB=F8H to FFH) then the routine terminates with Flag C (175AH). Otherwise CMASK is set to 80H, the leftmost pixel, and 0008H added to CLOC. In Multicolour Mode control transfers to a separate routine (1779H).

```
Address... 16C5H
Name...... RIGHTC
Entry..... None
Exit...... None
Modifies.. AF
```

Standard routine to move the current pixel physical address one position right. In Graphics Mode CMASK is first shifted one bit right, if the pixel still remains within the byte the routine terminates. Otherwise CMASK is set to 80H, the leftmost pixel, and 0008H added to CLOC. Note that incorrect addresses will be produced if the right hand edge of the screen is exceeded. In Multicolour Mode control transfers to a separate routine (178BH).

```
Address... 16D8H
```

This routine moves the current pixel physical address one position left. If the left hand edge of the screen is exceeded it returns Flag C and the physical address is unchanged. In Graphics Mode CMASK is first shifted one bit left, if the pixel still remains within the byte the routine terminates. If CLOC is at the leftmost character cell (LSB=00H to 07H) then the routine terminates with Flag C (175AH). Otherwise CMASK is set to 01H, the rightmost pixel, and 0008H subtracted from CLOC. In Multicolour Mode control transfers to a separate routine (179CH).

```
Address... 16EEH
Name...... LEFTC
Entry..... None
Exit...... None
Modifies.. AF
```

Standard routine to move the current pixel physical address one position left. In Graphics Mode CMASK is first shifted one bit left, if the pixel still remains within the byte the routine terminates. Otherwise CMASK is set to 01H, the leftmost pixel, and 0008H subtracted from CLOC. Note that incorrect addresses will be produced if the left hand edge of the screen is exceeded. In Multicolour Mode control transfers to a separate routine (17ACH).

```
Address... 170AH
Name...... TDOWNC
Entry..... None
Exit...... Flag C if off screen
Modifies.. AF
```

Standard routine to move the current pixel physical address one position down. If the bottom edge of the screen is exceeded it returns Flag C and the physical address is unchanged. In Graphics Mode CLOC is first incremented, if it still remains within an eight byte boundary the routine terminates. If CLOC was in the bottom character row (CLOC>=1700H) then the routine terminates with Flag C (1759H). Otherwise 00F8H is added to CLOC. In Multicolour Mode control transfers to a separate routine (17C6H).

```
Address... 172AH
Name...... DOWNC
Entry..... None
Exit...... None
Modifies.. AF
```

Standard routine to move the current pixel physical address one position down. In Graphics Mode CLOC is first incremented, if it still remains within an eight byte boundary the routine terminates. Otherwise 00F8H is added to CLOC. Note that incorrect addresses will be produced if the bottom edge of the screen is exceeded. In Multicolour Mode control transfers to a separate routine (17DCH).

```
Address... 173CH
Name...... TUPC
Entry..... None
Exit...... Flag C if off screen
Modifies.. AF
```

Standard routine to move the current pixel physical address one position up. If the top edge of the screen is exceeded it returns with Flag C and the physical address is unchanged. In Graphics Mode CLOC is first decremented, if it still remains within an eight byte boundary the routine terminates. If CLOC was in the top character row (CLOC<0100H) then the routine terminates with Flag C. Otherwise 00F8H is subtracted from CLOC. In Multicolour Mode control transfers to a separate routine (17E3H).

```
Address... 175DH
Name...... UPC
Entry..... None
Exit...... None
Modifies.. AF
```

Standard routine to move the current pixel physical address one position up. In Graphics Mode CLOC is first decremented, if it still remains within an eight byte boundary the routine terminates. Otherwise 00F8H is subtracted from CLOC. Note that incorrect addresses will be produced if the top edge of the screen is exceeded. In Multicolour Mode control transfers to a separate routine (17F8H).

```
Address... 1779H
```

This is the Multicolour Mode version of the routine at 16ACH. It is identical to the Graphics Mode version except that CMASK is shifted four bit positions right and becomes F0H if a cell boundary is crossed.

```
Address... 178BH
```

This is the Multicolour Mode version of the RIGHTC standard routine. It is identical to the Graphics Mode version except that CMASK is shifted four bit positions right and becomes F0H if a cell boundary is crossed.

```
Address... 179CH
```

This is the Multicolour Mode version of the routine at 16D8H. It is identical to the Graphics Mode version except that CMASK is shifted four bit positions left and becomes 0FH if a cell boundary is crossed.

```
Address... 17ACH
```

This is the Multicolour Mode version of the LEFTC standard routine. It is identical to the Graphics Mode version except that CMASK is shifted four bit positions left and becomes 0FH if a cell boundary is crossed.

```
Address... 17C6H
```

This is the Multicolour Mode version of the TDOWNC standard routine. It is identical to the Graphics Mode version except that the bottom boundary address is 0500H instead of 1700H. There is a bug in this routine which will cause it to behave unpredictably if MLTCGP, the Character Pattern Table base address, is changed from its normal value of zero. There should be an EX DE,HL instruction inserted at address 17CEH. If the Character Pattern Table base is increased the routine will think it has reached the bottom of the screen when it actually has not. This routine is used by the "PAINT" statement so the following demonstrates the fault:

```
10 BASE(17)=&H1000
20 SCREEN 3
30 PSET(200,0)
40 DRAW"D180L100U180R100"
50 PAINT(150,90)
60 GOTO 60
```

```
Address... 17DCH
```

This is the Multicolour Mode version of the DOWNC standard routine, it is identical to the Graphics Mode version.

```
Address... 17E3H
```

This is the Multicolour Mode version of the TUPC standard routine. It is identical to the Graphics Mode version except that is has a bug as above, this time there should be an EX DE,HL instruction at address 17EBH. If the Character Pattern Table base address is increased the routine will think it is within the table when it has actually exceeded the top edge of the screen. This may be demonstrated by removing the "R100" part of Line 40 in the previous program.

```
Address... 17F8H
```

This is the Multicolour Mode version of the UPC standard routine, it is identical to the Graphics Mode version.

```
Address... 1809H
Name...... NSETCX
Entry..... HL=Pixel fill count
Exit...... None
Modifies.. AF, BC, DE, HL, EI
```

Standard routine to set the colour of multiple pixels horizontally rightwards from the current pixel physical address. Although its function can be duplicated by the SETC and RIGHTC standard routines this would result in significantly slower operation. The supplied pixel count should be chosen so that the right-hand edge of the screen is not passed as this will produce anomalous behaviour. The current pixel physical address is unchanged by this routine. In Graphics Mode CMASK is first examined to determine the number of pixels to the right within the current character cell. Assuming the fill count is large enough these are then set (186CH). The remaining fill count is divided by eight to determine the number of whole character cells. Successive bytes in the Character Pattern Table are then zeroed and the corresponding bytes in the Colour Table set from ATRBYT to fill these whole cells. The remaining fill count is then con|ed to a bit mask, using the seven byte table at 185DH, and these pixels are set (186CH). In Multicolour Mode control transfers to a separate routine(18BBH).

```
Address... 186CH
```

This routine sets up to eight pixels within a cell to a specified colour in Graphics Mode. ATRBYT contains the colour code, register pair HL the address of the relevant byte in the Character Pattern Table and register A a bit mask, 11100000 for example, where every 1 specifies a bit to be set. If ATRBYT matches the existing 1 pixel colour in the corresponding Colour Table byte then each specified bit is set to 1 in the Character Pattern Table byte. If ATRBYT matches the existing 0 pixel colour in the corresponding Colour Table byte then each specified bit is set to 0 in the Character Pattern Table byte. If ATRBYT does not match either of the existing colours in the Colour Table Byte then normally each specified bit is set to 1 in the Character Pattern Table byte and the 1 pixel colour changed in the Colour Table byte. However if this would result in all bits being set to 1 in the Character Pattern Table byte then each specified bit is set to 0 and the 0 pixel colour changed in the Colour Table byte.

```
Address... 18BBH
```

This is the Multicolour Mode version of the NSETCX standard routine. The SETC and RIGHTC standard routines are called until the fill count is exhausted. Speed of operation is not so important in Multicolour Mode because of the lower screen resolution and the consequent reduction in the number of operations required.

```
Address... 18C7H
Name...... GTASPC
Entry..... None
Exit...... DE=ASPCT1, HL=ASPCT2
Modifies.. DE, HL
```

Standard routine to return the "CIRCLE" statement default aspect ratios.

```
Address... 18CFH
Name...... PNTINI
Entry..... A=Boundary colour (0 to 15)
Exit...... Flag C if illegal colour
Modifies.. AF
```

Standard routine to set the boundary colour for the "PAINT" statement. In Multicolour Mode the supplied colour code is placed in BDRATR. In Graphics Mode BDRATR is copied from ATRBYT as it is not possible to have separate paint and boundary colours.

```
Address... 18E4H
Name...... SCANR
Entry..... B=Fill switch, DE=Skip count
Exit...... DE=Skip remainder, HL=Pixel count
Modifies.. AF, BC, DE, HL, EI
```

Standard routine used by the "PAINT" statement handler to search rightwards from the current pixel physical address until a colour code equal to BDRATR is found or the edge of the screen is reached. The terminating position becomes the current pixel physical address and the initial position is returned in CSAVEA and CSAVEM. The size of the traversed region is returned in register pair HL and FILNAM+1. The traversed region is normally filled in but this can be inhibited, in Graphics Mode only, by using an entry parameter of zero in register B. The skip count in register pair DE determines the maximum number of pixels of the required colour that may be ignored from the initial starting position. This facility is used by the "PAINT" statement handler to search for gaps in a horizontal boundary blocking its upward progress.

```
Address... 197AH
Name...... SCANL
Entry..... None
Exit...... HL=Pixel count
Modifies.. AF, BC, DE, HL, EI
```

Standard routine to search leftwards from the current pixel physical address until a colour code equal to BDRATR is found or the edge of the screen is reached. The terminating position becomes the current pixel physical address and the size of the traversed region is returned in register pair HL. The traversed region is always filled in.

```
Address... 19C7H
```

This routine is used by the SCANL and SCANR standard routines to check the current pixel's colour against the boundary colour in BDRATR.

```
Address... 19DDH
Name...... TAPOOF
Entry..... None
Exit...... None
Modifies.. EI
```

Standard routine to stop the cassette motor after data has been written to the cassette. After a delay of 550 ms, on MSX machines with one wait state, control drops into the TAPIOF standard routine.

```
Address... 19E9H
Name...... TAPIOF
Entry..... None
Exit...... None
Modifies.. EI
```

Standard routine to stop the cassette motor after data has been read from the cassette. The motor relay is opened via the PPI Mode Port. Note that interrupts, which must be disabled during cassette data transfers for timing reasons, are enabled as this routine terminates.

```
Address... 19F1H
Name...... TAPOON
Entry..... A=Header length switch
Exit...... Flag C if CTRL-STOP termination
Modifies.. AF, BC, HL, DI
```

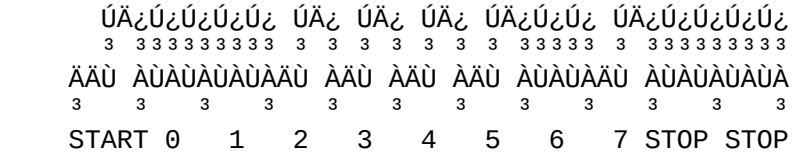
Standard routine to turn the cassette motor on, wait 550 ms for the tape to come up to speed and then write a header to thecassette. A header is a burst of HI cycles written in front of every data block so the baud rate can be determined when the data is read back. The length of the header is determined by the contents of register A: 00H=Short header, NZ=Long header. The BASIC cassette statements "SAVE", "CSAVE" and "BSAVE" all generate a long header at the start of the file, in front of the identification block, and thereafter use short headers between data blocks. The number of cycles in the header is also modified by the current baud rate so as to keep its duration constant:

```
1200 Baud SHORT ... 3840 Cycles ... 1.5 Seconds
1200 Baud LONG ... 15360 Cycles ... 6.1 Seconds
2400 Baud SHORT ... 7936 Cycles ... 1.6 Seconds
2400 Baud LONG ... 31744 Cycles ... 6.3 Seconds
```

After the motor has been turned on and the delay has expired the contents of HEADER are multiplied by two hundred and fiftysix and, if register A is non-zero, by a further factor of four to produce the cycle count. HI cycles are then generated (1A4DH) until the count is exhausted whereupon control transfers to the BREAKX standard routine. Because the CTRL-STOP key is only examined at termination it is impossible to break out part way through this routine.

```
Address... 1A19H
Name...... TAPOUT
Entry..... A=Data byte
Exit...... Flag C if CTRL-STOP termination
Modifies.. AF, B, HL
```

Standard routine to write a single byte of data to the cassette. The MSX ROM uses a software driven FSK (Frequency Shift Keyed) method for storing information on the cassette. At the 1200 baud rate this is identical to the Kansas City Standard used by the BBC for the distribution of BASICODE programs. At 1200 baud each 0 bit is written as one complete 1200 Hz LO cycle and each 1 bit as two complete 2400 Hz HI cycles. The data rate is thus constant as 0 and 1 bits have the same duration. When the 2400 baud rate is selected the two frequencies change to 2400 Hz and 4800 Hz but the format is otherwise unchanged. A byte of data is written with a 0 start bit (1A50H), eight data bits with the least significant bit first, and two 1 stop bits (1A40H). At the 1200 baud rate a single byte will have a nominal duration of 11 x 833 æs = 9.2 ms. After the stop bits have been written control transfers to the BREAKX standard routine to check the CTRL-STOP key. The byte 43H is shown below as it would be written to cassette:




```
1 = two "short" transitions (as STOP BITS)
0 = one "long" transition (as START BIT)
```

Figure 39: Cassette Data Byte

It is important not to leave too long an interval between bytes when writing data as this will increase the error rate. An inter-byte gap of 80 æs, for example, produces a read failure rate of approximately twelve percent. If a substantial amount of processing is required between each byte then buffering should be used to lump data into headered blocks. The BASIC "SAVE" format is of this type.

```
Address... 1A39H
```

This routine writes a single LO cycle with a length of approximately 816 æs to the cassette. The length of each half of the cycle is taken from LOW and control transfers to the general cycle generator (1A50H).

```
Address... 1A40H
```

This routine writes two HI cycles to the cassette. The first cycle is generated (1A4DH) followed by a 17 æs delay and then the second cycle (1A4DH).

```
Address... 1A4DH
```

This routine writes a single HI cycle with a length of approximately 396 æs to the cassette. The length of each half of the cycle is taken from HIGH and control drops into the general
cycle generator.

```
Address... 1A50H
```

This routine writes a single cycle to the cassette. The length of the cycle's first half is supplied in register L and its second half in register H. The first length is counted down and then the Cas Out bit set via the PPI Mode Port. The second length is counted down and the Cas Out bit reset. On all MSX machines the Z80 runs at a clock frequency of 3.579545 MHz (280 ns) with one wait state during the M1 cycle. As this routine counts every 16T states each unit increment in the length count represents a period of 4.47 æs. There is also a fixed overhead of 20.7 æs associated with the routine whatever the length count.

```
Address... 1A63H
Name...... TAPION
Entry..... None
Exit...... Flag C if CTRL-STOP termination
Modifies.. AF, BC, DE, HL, DI
```

Standard routine to turn the cassette motor on, read the cassette until a header is found and then determine the baud rate. Successive cycles are read from the cassette and the length of each one measured (1B34H). When 1,111 cycles have been found with less than 35 æs variation in their lengths a header has been located. The next 256 cycles are then read (1B34H) and averaged to determine the cassette HI cycle length. This figure is multiplied by 1.5 and placed in LOWLIM where it defines the minimum acceptable length of a 0 start bit. The HI cycle length is placed in WINWID and will be used to discriminate between LO and HI cycles.

```
Address... 1ABCH
Name...... TAPIN
Entry..... None
Exit...... A=Byte read, Flag C if CTRL-STOP or I/O error
Modifies.. AF, BC, DE, L
```

Standard routine to read a byte of data from the cassette. The cassette is first read continuously until a start bit is found. This is done by locating a negative transition, measuring the following cycle length (1B1FH) and comparing this to see if it is greater than LOWLIM. Each of the eight data bits is then read by counting the number of transitions within a fixed period of time (1B03H). If zero or one transitions are found it is a 0 bit, if two or three are found it is a 1 bit. If more than three transitions are found the routine terminates with Flag C as this is presumed to be a hardware error of some sort. After the value of each bit has been determined a further one or two transitions are read (1B23H) to retain synchronization. With an odd transition count one more will be read, with an even transition count two more.

```
Address... 1B03H
```

This routine is used by the TAPIN standard routine to count the number of cassette transitions within a fixed period of time. The window duration is contained in WINWID and is approximately 1.5 times the length of a HI cycle as shown below:

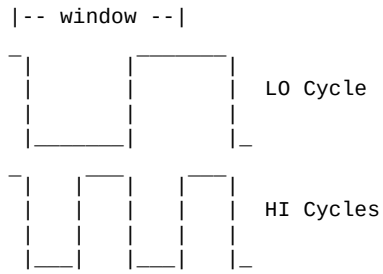


Figure 40: Cassette Window.

The Cas Input bit is continuously sampled via PSG Register 14 and compared with the previous reading held in register E. Each time a change of state is found register C is incremented. The sampling rate is once every 17.3 æs so the value in WINWID, which was determined by the TAPION standard routine with a count rate of 11.45 æs, is effectively multiplied one and a half times.

```
Address... 1B1FH
```

This routine measures the time to the next cassette input transition. The Cassette Input bit is continuously sampled via PSG Register 14 until it changes from the state supplied in register E. The state flag is then in|ed and the duration count returned in register C, each unit increment represents a period of 11.45 æs.

```
Address... 1B34H
```

This routine measures the length of a complete cassette cycle from negative transition to negative transition. The Cassette Input bit is sampled via PSG Register 14 until it goes to zero. The transition flag in register E is set to zero and the time to the positive transition measured (1B23H). The time to the negative transition is then measured (1B25H) and the total returned in register C.

```
Address... 1B45H
Name...... OUTDO
Entry..... A=Character to output
Exit...... None
Modifies.. EI
```

Standard routine used by the BASIC Interpreter to output a character to the current device. The ISFLIO standard routine is first used to check whether output is currently directed to an I/O buffer, if so control transfers to the sequential output driver (6C48H) via the CALBAS standard routine. If PRTFLG is zero control transfers to the CHPUT standard routine to output the character to the screen. Assuming the printer is active RAWPRT is checked. If this is non-zero the character is passed directly to the printer (1BABH), otherwise control drops into the OUTDLP standard routine.

```
Address... 1B63H
Name...... OUTDLP
Entry..... A=Character to output
Exit...... None
Modifies.. EI
```

Standard routine to output a character to the printer. If the character is a TAB code (09H) spaces are issued to the OUTDLP standard routine until LPTPOS is a multiple of eight (0, 8, 16 etc.). If the character is a CR code (0DH) LPTPOS is zeroed if it is any other control code LPTPOS is unaffected, if it is a displayable character LPTPOS is incremented. If NTMSXP is zero, meaning an MSX-specific printer is connected, the character is passed directly to the printer (1BABH). Assuming a normal printer is connected the CNVCHR standard routine is used to check for graphic characters. If the character is a header code (01H) the routine terminates with no action. If it is a con|ed graphic character it is replaced by a space, all other characters are passed to the printer (1BACH).

```
Address... 1B97H
```

This twenty byte table is used by the keyboard decoder to find the correct routine for a given key number:

```
KEY NUMBER  TO     FUNCTION
-------------------------------------
00H to 2FH  0F83H  Rows 0 to 5
30H to 32H  0F10H  SHIFT, CTRL, GRAPH
33H         0F36H  CAP
34H         0F10H  CODE
35H to 39H  0FC3H  F1 to F5
3AH to 3BH  0F10H  ESC, TAB
3CH         0F46H  STOP
3DH to 40H  0F10H  BS, CR, SEL, SPACE
41H         0F06H  HOME
42H to 57H  0F10H  INS, DEL, CURSOR
```

```
Address... 1BABH
```

This routine is used by the OUTDLP standard routine to pass a character to the printer. It is sent via the LPTOUT standard routine, if this returns Flag C control transfers to the "Device I/O error" generator (73B2H) via the CALBAS standard routine.

```
Address... 1BBFH
```

The following 2 KB contains the power-up character set. The first eight bytes contain the pattern for character code 00H, the second eight bytes the pattern for character code 01H and so on to character code FFH.

```
Address... 23BFH
Name...... PINLIN
Entry..... None
Exit...... HL=Start of text, Flag C if CTRL-STOP termination
Modifies.. AF, BC, DE, HL, EI
```

Standard routine used by the BASIC Interpreter Mainloop to collect a logical line of text from the console. Control transfers to the INLIN standard routine just after the point where the previous line has been cut (23E0H).

```
Address... 23CCH
Name...... QINLIN
Entry..... None
Exit...... HL=Start of text, Flag C if CTRL-STOP termination
Modifies.. AF, BC, DE, HL, EI
```

Standard routine used by the "INPUT" statement handler to collect a logical line of text from the console. The characters "? " are displayed via the OUTDO standard routine and control drops into the INLIN standard routine.

```
Address... 23D5H
Name...... INLIN
Entry..... None
Exit...... HL=Start of text, Flag C if CTRL-STOP termination
Modifies.. AF, BC, DE, HL, EI
```

Standard routine used by the "LINE INPUT" statement handler to collect a logical line of text from the console. Characters are read from the keyboard until either the CR or CTRL-STOP keys are pressed. The logical line is then read from the screen character by character and placed in the Workspace Area text buffer BUF. The current screen coordinates are first taken from CSRX and CSRY and placed in FSTPOS. The screen row immediately above the current one then has its entry in LINTTB made non-zero (0C29H) to stop it extending logically into the current row. Each keyboard character read via the CHGET standard routine is checked (0919H) against the editing key table at 2439H. Control then transfers to one of the editing routines or to the default key handler (23FFH) as appropriate. This process 4 continues until Flag C is returned by the CTRL-STOP or CR routines. Register pair HL is then set to point to the start of BUF and the routine terminates. Note that the carry, flag is cleared when Flag NZ is also returned to distinguish between a CR or protected CTRL-STOP termination and a normal CTRL-STOP termination.

```
Address... 23FFH
```

This routine processes all characters for the INLIN standard routine except the special editing keys. If the character is a TAB code (09H) spaces are issued (23FFH) until CSRX is a multiple of eight plus one (columns 1, 9, 17, 25, 33). If the character is a graphic header code (01H) it is simply echoed to the OUTDO standard routine. All other control codes

smaller than 20H are echoed to the OUTDO standard routine after which INSFLG and CSTYLE are zeroed. For the displayable characters INSFLG is first checked and a space inserted (24F2H) if applicable before the character is echoed to the OUTDO standard routine.

Address... 2439H

This table contains the special editing keys recognized by the INLIN standard routine together with the relevant addresses:

```
CODE TO    FUNCTION
-----------------------------------------
08H  2561H BS, backspace
12H  24E5H INS, toggle insert mode
1BH  23FEH ESC, no action
02H  260EH CTRL-B, previous word
06H  25F8H CTRL-F, next word
0EH  25D7H CTRL-N, end of logical line
05H  25B9H CTRL-E, clear to end of line
03H  24C5H CTRL-STOP, terminate
0DH  245AH CR, terminate
15H  25AEH CTRL-U, clear line
7FH  2550H DEL, delete character
```

Address... 245AH

This routine performs the CR operation for the INLIN standard routine. The starting coordinates of the logical line are found (266CH) and the cursor removed from the screen (0A2EH). Up to 254 characters are then read from the VDP VRAM (0BD8H) and placed in BUF. Any null codes (00H) are ignored, any characters smaller than 20H are replaced by a graphic header code (01H) and the character itself with 40H added. As the end of each physical row is reached LINTTB is checked (0C1DH) to see whether the logical line extends to the next physical row. Trailing spaces are then stripped from BUF and a zero byte added as an end of text marker. The cursor is restored to the screen (09E1H) and its coordinates set to the last physical row of the logical line via the POSIT standard routine. A LF code is issued to the OUTDO standard routine, INSFLG is zeroed and the routine terminates with a CR code (0DH) in register A and Flag NZ,C. This CR code will be echoed to the screen by the INLIN standard routine mainloop just before it terminates.

Address... 24C5H

This routine performs the CTRL-STOP operation for the INLIN standard routine. The last physical row of the logical line is found by examining LINTTB (0C1DH), CSTYLE is zeroed, a zero byte is placed at the start of BUF and all music variables are cleared via the GICINI standard routine. TRPTBL is then examined (0454H) to see if an "ON STOP" statement is active, if so the cursor is reset (24AFH) and the routine terminates with Flag NZ,C. BASROM is then checked to see whether a protected ROM is running, if so the cursor is reset (24AFH) and the routine terminates with Flag NZ,C. Otherwise the cursor is reset (24B2H) and the routine terminates with Flag Z,C.

Address... 24E5H

This routine performs the INS operation for the INLIN standard routine. The current state of INSFLG is in|ed and control terminates via the CSTYLE setting routine (242CH).

Address... 24F2H

This routine inserts a space character for the default key section of the INLIN standard routine. The cursor is removed (0A2EH) and the current cursor coordinates taken from CSRX and CSRY. The character at this position is read from the VDP VRAM (0BD8H) and replaced with a space (0BE6H). Successive characters are then copied one column position to the right until the end of the physical row is reached. At this point LINTTB is examined (0C1DH) to determine whether the logical line is extended, if so the process continues on the next physical row. Otherwise the character taken from the last column position is examined, if this is a space the routine terminates by replacing the cursor (09E1H). Otherwise the physical row's entry in LINTTB is zeroed to indicate an extended logical line. The number of the next physical row is compared with the number of rows on the screen (0C32H). If the next row is the last one the screen is scrolled up (0A88H), otherwise a blank row is inserted (0AB7H) and the copying process continues.

```
Address... 2550H
```

This routine performs the DEL operation for the INLIN standard routine. If the current cursor position is at the rightmost column and the logical line is not extended no action is taken other than to write a space to the VDP VRAM (2595H). Otherwise a RIGHT code (1CH) is issued to the OUTDO standard routine and control drops into the BS routine.

```
Address... 2561H
```

This routine performs the BS operation for the INLIN standard routine. The cursor is first removed (0A2EH) and the cursor column coordinate decremented unless it is at the leftmost position and the previous row is not extended. Characters are then read from the VDP VRAM (0BD8H) and written back one position to the left (0BE6H) until the end of the logical line is reached. At this point a space is written to the VDP VRAM (0BE6H) and the cursor character is restored (09E1H).

```
Address... 25AEH
```

This routine performs the CTRL-U operation for the INLIN standard routine. The cursor is removed (0A2EH) and the start of the logical line located (266CH) and placed in CSRX and CSRY. The entire logical line is then cleared (25BEH).

```
Address... 25B9H
```

This routine performs the CTRL-E operation for the INLIN standard routine. The cursor is removed (0A2EH) and the remainder of the physical row cleared (0AEEH). This process is repeated for successive physical rows until the end of the logical line is found in LINTBB (0C1DH). The cursor is then restored (09E1H), INSFLG zeroed and CSTLYE reset to a block cursor (242DH).

```
Address... 25D7H
```

This routine performs the CTRL-N operation for the INLIN standard routine. The cursor is removed (0A2EH) and the last physical row of the logical line found by examination of LINTTB (0C1DH). Starting at the rightmost column of this physical row characters are read from the VDP VRAM (0BD8H) until a non-space character is found. The cursor coordinates are then set one column to the right of this position (0A5BH) and the routine terminates by restoring the cursor (25CDH).

```
Address... 25F8H
```

This routine performs the CTRL-F operation for the INLIN standard routine. The cursor is removed (0A2EH) and moved successively right (2624H) until a non-alphanumeric character is found. The cursor is then moved successively right (2624H) until an alphanumeric character is found. The routine terminates by restoring the cursor (25CDH).

```
Address... 260EH
```

This routine performs the CTRL-B operation for the INLIN standard routine. The cursor is removed (0A2EH) and moved successively left (2634H) until an alphanumeric character is found. The cursor is then moved successively left (2634H) until a non-alphanumeric character is found and then moved one position right (0A5BH). The routine terminates by restoring the cursor (25CDH).

```
Address... 2624H
```

This routine moves the cursor one position right (0A5BH), loads register D with the rightmost column number, register E with the bottom row number and then tests for an alphanumeric character at the cursor position (263DH).

```
Address... 2634H
```

This routine moves the cursor one position left (0A4CH), loads register D with the leftmost column number and register E with the top row number. The current cursor coordinates are compared with these values and the routine terminates Flag Z if the cursor is at this position. Otherwise the character at this position is read from the VDP VRAM (0BD8H) and checked to see if it is alphanumeric. If so the routine terminates Flag NZ,C otherwise it terminates Flag NZ,NC. The alphanumeric characters are the digits "0" to "9" and the letters "A" to "Z" and "a" to "z". Also included are the

graphics characters 86H to 9FH and A6H to FFH, these were originally Japanese letters and should have been excluded during the conversion to the UK ROM.

```
Address... 266CH
```

This routine finds the start of a logical line and returns its screen coordinates in register pair HL. Each physical row above the current one is checked via the LINTTB table (0C1DH) until a non-extended row is found. The row immediately below this on the screen is the start of the logical line and its row number is placed in register L. This is then compared with FSTPOS, which contains the row number when the INLIN standard routine was first entered, to see if the cursor is still on the same line. If so the column coordinate in register H is set to its initial position from FSTPOS. Otherwise register H is set to the leftmost position to return the whole line.

```
Address...2680H
```
JP to power-up initialize routine (7C76H).

```
Address...2683H
```
JP to the SYNCHR standard routine (558CH).

```
Address...2686H
```
JP to the CHRGTR standard routine (4666H).

```
Address...2689H
```
JP to the GETYPR standard routine (5597H).