USEFUL ROM SUBROUTINES FOR ASSEMBLY PROGRAMMING

KEYBOARD SCANNING ROUTINE

The keyboard scanning routine resides at 2EF4 hex. This routine scans the keyboard once and returns. If a key is pressed, the A register will contain the code for that key; otherwise this register will contain zero. Registers AF, BC, DE and HL are all modified by the routine, so if the contents of these registers must be preserved they should be pushed onto the stack before the routine is called. The following example shows how the routine would be used to wait for the RETURN key to be pressed:

```
SCAN  CALL  2EF4H      ;scan keyboard once
      OR    A          ;any key pressed ?
      JR    Z,SCAN     ;back if not
      CP    0DH        ;was it RETURN key ?
      JR    NZ,SCAN    ;back if not
                       ;otherwise continue
```

CHARACTER OUTPUT SUBROUTINE

A routine which outputs a single character to the video display is located at 033A Hex. The code for the character to be displayed must be in the A register, while the character will be displayed on the screen at the position corresponding to the current value of the cursor pointer. All registers are preserved. Here is how the routine is called to display the word 'HI' followed by a carriage return:

```
LD    A,4lH   ;load reg A with code 'H'
CALL  033AH   ;and display
LD    A,4lH   ;same with 'I'
CALL  033AH
LD    A,0DH   ;now load A with CR code
CALL  033AH   ;and update screen
```

MESSAGE OUTPUT SUBROUTINE

A very useful subroutine located at 28A7 hex can display a string of character codes as a message on the screen. The string of character codes must end with a zero byte. The HL register pair must be set to the start of the string before the subroutine is called. All registers are used by the subroutine. Here is how it is used:

```
      LD    HL,MSG          ;load HL with start of string
      CALL  2BA7H           ;and call print subroutine

MSG   DEFM  'READY    '     ;main message string
      DEFB  ODH             ;carriage return
      DEFB  0               ;null byte to terminate
```

COMPARE SYMBOL (EXAMINE STRING) - RST 08H

A routine which is called using the RST 08H instruction can be used to compare a character in a string pointed to by the HL register, with the value in the location following the RST 08H instruction itself. If there is a match, control is returned to the instruction 2 bytes after the RST 08H, with the HL register incremented by 1 and the next character of the string in the A register. This allows repeated calls to check for an expected sequence of characters. Note that if a match is not found, the RST 08H routine does not return from where it is called, but jumps instead to the BASIC interpreter's input phase after ' printing the 'SYNTAX ERROR' message. Here is how the routine is used to check that the string pointed to by HL register is 'A=B=C':

```
RST 08H         ;test for 'A'
DEFB 41H        ;hex value of A for comparison
RST 0BH         ;must have found, so try for  '='
DEFB 3DH        ;hex value of '='
```

```
RST 08H             ;OK so far, try for 'B'
DEFB 42H
RST 08H             ;now look for second '='
DEFB 3DH
RST 08H             ;finally check for 'C'
DEFB 43H ...    ;must have been OK, so proceed
```

LOAD & CHECK NEXT CHARACTER IN STRING -- RST 10H

The RST 10H instruction may be used to call a routine which loads the A register with the next character of a string pointed to by the HL register, and clears the CARRY flag if character is alphanumeric. Blanks and control codes 09H and OBH are skipped automatically. The HL register is incremented before each character is loaded, therefore on the first call the HL register should be set to point to the address BEFORE the location of the first string character to be tested. The string must be terminated by a null byte.

Here is an example of this routine in use. Note that if it is used immediately after the RST 08H instruction as shown, the HL register will automatically be incremented to point to the next character in the string:

```
RST 08H         ;test for
DEFB 3DH
RST 10H         ;fetch & check next char
JR NC,VAR       ;will go to VAR if alpha
....            ;continues if numeral
```

COMPARE DE & HL REGISTER PAIRS - RST 18H

The instruction RST 18H may be used to call a routine which compares the contents of the DE and HL register pairs. The routine uses the A register only, but will only work for unsigned or positive numbers. Upon returning, the result of the comparison will be in the status register:

```
HL      <       DE : carry set
HL      >       DE : no carry
HL      <>      DE : NZ
HL      =       DE : Z
```

Here is an example of its use. Assume the DE pair contains a number and we want to check that it falls within a certain range - say between 100 and 500 (decimal):

```
LD      HL,500      ;load HL with upper limit
RST     18H         ;& call comparison routine
JR      C,ERR       ;carry means num>500
LD      HL,100      ;now set for lower limit
RST     18H         ;& try again
JR      NC,ERR      ;no carry means num < 100
....                ;if still here, must be OK
```

SOUND DRIVER

Located at 345C hex is a routine which can be used to produce sounds via the VZ200/300's internal piezo speaker. Before calling the routine, the HL register pair must be loaded with a number representing the pitch (frequency) of the tone to be produced, while the BC register pair must be loaded with the number of cycles of the tone required (ie the duration in cycles). All registers are used. The frequency coding used is inversely proportional to frequency, ie the smaller the number loaded into the HL register pair, the higher the frequency. As a guide, the low C produced by the VZ200/300's SOUND command in BASIC can be produced using the decimal number 526, the middle C using 529 and the high C using 127. Here is how you would use the routine to get say 75 cycles of the middle C:

```
LD      HL,259  ;set frequency code
LD      BC,75   ;set number of cycles
CALL    345CH   ;& call sound routine
```

'BEEP' ROUTINE

The routine which is used by BASIC to produce the short 'beep' when a key is pressed is located at the address 3450 hex.
It disturbs all registers except the HL pair. To make a beep:

```
CALL    3450H      ;make a 'beep'
```

CLEAR SCREEN

A routine located at 0lC9 hex may be used to clear the video screen, home the cursor and select display mode (0). it disturbs all registers. Again it is used by simply calling it.

```
CALL    01C9H           ;clear screen, home cursor etc.
```

PRINTER DRIVER

The printer driver routine is located at 058D hex. To send a character to the printer, load the chracter's ASCII code into the C register and call the driver - After printing, the character code will be returned in both the A and C registers. All other registers are disturbed. For example to print the letter 'A' (ASCII code 97 decimal), you would use:

```
LD C,97                 ;set up code in C register
CALL 058DH    ;& call printer driver
```

A line feed character (OAH) is automatically inserted after a carriage return (ODH). If the driver is called with a null byte in the C register, it will simply check the printer status and return with bit 0 of the A register either set or cleared. The routine does check for a BREAK key depression, and if one is detected, it will return with the carry flag set.

CHECK PRINTER STATUS

A routine to check printer status is located at 05C4 hex. When called it loads the printer status (I/O port 00H) into the A register and returns. Bit 0 will be set (1) if the printer is busy, or cleared (0) if it is ready. No other registers are disturbed. An example:

```
TEST    CALL    054CH       ;check is printer ready
        BIT     0,A         ;test bit 0
        JR      NZ,TEST     ;loop if busy -continue if ready
```

SEND CR-LF TO PRINTER

A routine located at 03AE2 hex may be used to send a carriage return and line feed combination to the printer. No registers need be set up before calling, but all registers are disturbed. If the break key is pressed while printing occurs (or while the printer driver is waiting for the printer to signal 'ready'), the routine will return with the carry flag set:

```
CALL    3AE2H       ;go send CR-LF to printer
JP      C,BRK       ;check if BREAK key is pressed
...                 ;apparently not
```

CONSTANT KEYBOARD SCAN

Scans the Keyboard until a key is pressed. Will not return until a key is pressed. If CTRL-BREAK is pressed, it is returned like other keys.
Returns with A containing character pressed.

```
CALL 0049H          ; scan keyboard
```

DELAY

Load BC with the size of the delay in mS.

```
        LD BC,delay
        CALL 0060H              ; delay
```

BASIC COLOR

Example routine (self explanatory)

```
        LD A,colour     ; 1-8
        LD HL,zero      ; HL points to 00h byte
        CALL 38A5h      ; change colour

zero    DEFB 00h
```

If A=0 or A>8 it will display a "Function Code Error" and exit.

BASIC SET/RESET/POINT

Load HL with the return address and PUSH it onto the stack. Load register A with one of the following:
00h = POINT 01H = RESET 80h = SET
PUSH AF onto the stack. Load A with the X-Coordinate and PUSH onto the stack. Point HL to ')' and NULL.
Load A with the Y Coordinate and JP 0150h
To SET (23,50):

```
        LD HL,return
        PUSH HL
        LD A,80h
        PUSH AF
        LD A,23
        PUSH AF
        LD HL,rbkt
        LD A,50
        JP 0150h
return  ......

rbkt    DEFB 29h
        DEFB 00h
```

DATA MOVE

DE=source location. HL=destination location. A=how much to move.

```
        LD DE,src
        LD HL,dst
        LD A,amt
        CALL 09D6h
```

DATA MOVE

DE=source location. HL=destination location. B=how much to move.

```
        LD DE,7000h
        LD HL,7080h
        LD B,20h
        CALL 09D7h
```

HL=DE-HL

Subtract HL from DE. Result in HL.

```
LD HL,20
LD DE,100
CALL 0BC7h
```

HL=DE+HL

Add HL to DE. Result in HL.

Subtract HL from DE. Result in HL.

```
LD HL,20
LD DE,100
CALL 0BD2h
```

*HL=DE*HL*

Multiply HL by DE. Result in HL.

Subtract HL from DE. Result in HL.

```
LD HL,20
LD DE,100
CALL 0BF2h
```

OUTPUT HL

Display HL value on the screen.

```
LD HL,20000
CALL 0FAFh
```

Displays "20000" on the screen.

OUTPUT STRING TO CURRENT DEVICE

HL points to start of string. Store Device Type in 789Ch. 0 = Video 1 = Printer 255 (-1) = Cassette

```
LD A,00h
LD (789Ch),A
LD HL,mesg
CALL 2B75h

mesg    DEFM 'BEWARE OF THE LEOPARD'
        DEFB 00h
```

BASIC COPY

CALL 3912h

MODE (0) or MODE(1)

Point HL to a location containing a ')'. Also clears screen in either mode.

```
        LD HL,rbkt
        CALL 2E71h      ; MODE(1)

rbkt    DEFB 29h        ; ')'
        DEFB 00h
```

* OR *

```
        LD HL,rbkt
        CALL 2E7Dh      ; MODE(0)

rbkt    DEFB 29h        ; ')'
        DEFB 00h
```

BASIC SOUND freq,dur

Put the Frequency into 7AD2h and the Duration into A.

```
        LD A,freq
        LD (7AD2h),A
        LD A,dur
        LD HL,zero
        CALL 2C05h

zero    DEFB 00h
```

* OR *

Set up a BASIC STATEMENT emulation of the Freq,Dur as you would with a BASIC SOUND command. Remember, if you use any illegal values (ie note <1 or >31 OR duration <1 or >9) you will get a "Function Code" error.

```
        LD HL,notes
        CALL 2BF5h

notes   DEFM "20,5;30,2;30,2;1,3;1,3"
        DEFB 00h
```

INT ASCII into A register

Load HL with the integer ASCII value.

```
        LD HL,value-1
        CALL 2B1Bh

value   DEFM '31'
```

A should now contain 31 decimal.

THE STRUCTURE OF THE DOS

The DOS is a ROM based DOS which is located in 4000H to 5FFFH. When the computer is powered up, the BASIC interpreter will jump to the DOS after initializing the BASIC pointers. The DOS will reserve a DOS vector of 310 bytes at the top of memory available. The DOS vector is pointed to by the index register IY and this vector is used to keep track of all DOS operations. Programmers should avoid modifying the IY register, otherwise the DOS will probably crash.

The DOS vectors contain the following elements:

DOSVTR = IY

NAME	BYTES	OFFSET	
FILNO	1	IY+O	FILE#
FNAM	8	IY+l	FILENAME
TYPE	2	IY+9	FILE TYPE
DK	1	IY+ll	SELECTED DRIVE# PATTERN
RQST	1	IY+12	REQUEST CODE
SOURCE	1	IY+13	SOURCE DRIVE FOR DCOPY
UBFR	2	IY+14	USER BUFFER ADDRESS
DESTIN	1	IY+16	DEST DRIVE FOR DCOPY
SCTR	1	IY+17	USER SPEC. SECTOR NUMBER
TRCK	1	IY+18	USER SPEC. TRACK NUMBER
RETRY	1	IY+19	RETRY COUNT
DTRCK	1	IY+20	CURRENT TRACK NUMBER
NSCT	1	IY+21	NEXT SCTR NUMBER
NTRK	1	IY+22	NEXT TRK NUMBER
FCB1	13	IY+23	FILE CONTROL BLOCK 1 OPEN FLAG, STATUS, FNAM, TRK#, SCTR#, ENTRY IN SCTR
FCB2	13	IY+36	FILE CONTROL BLOCK 2 OPEN FLAG, STATUS, FNAM, TRK#, SCTR#, ENTRY IN SCTR
DBFR	2	IY+49	DATA BUFFER ADDRESS
LTHCPY	1	IY+51	COPY OF LATCH
MAPADR	2	IY+52	TRACK/SECTOR MAP ADDRESS
TRKCNT	1	IY+54	TRK CNT FOR DCOPY
TRKPTR	1	IY+55	TRK PTR FOR DCOPY
PHASE	1	IY+56	STEPPER PHASE

DISK STRUCTURE

The DOS uses TRK 0, sector 0 to sector 14 as the directory. TRK 0 sector 15 is used to hold the track map of the disk with one bit corresponding to a sector used. Each directory entry contains 16 bytes. Therefore 1 sector can hold 8 entries and 1 diskette can have a maximum of 112 entries.

File type	1	byte
Delimitor (3AH)	1	byte
File name	8	byte
Start address	2	byte
End address	2	byte
Start track	1	byte
Start sector	1	byte

DOS ENTRY POINTS

A jump table to the DOS subroutines is positioned at the fixed address from 4008H to 4044H. The jump table contains the following elements:

ADDRESS	CONTENT	DOS SUBROUTINE
4008H	JP PWRON	Disk power ON
40OBH	JP PWOFF	Disk power OFF
400EH	JP ERROR	Error handling routine
4011H	JP RDMAP	Read the track map of the disk
4014H	JP CLEAR	Clear a sector of the disk

4017H	JP SVMAP	Save the track map to the disk
401AH	JP INIT	Initialize the disk
401DH	JP CSI	Command string interpreter
4020H	JP HEX	Convert ASCII to HEX
4023H	JP IDAM	Read identification address mark
4026H	JP CREATE	Create an entry in directory
4029H	JP MAP	Search for empty sector
402CH	JP SEARCH	Search for file in directory
402FH	JP FIND	Search empty space in directory
4032H	JP WRITE	Write a sector to disk
4035H	JP READ	Read a sector from disk
4038H	JP DLY	Delay mS in reg C
403BH	JP STPIN	Step in
403EH	JP STPOUT	Step out
4041H	JP DKLOAD	Load a file from disk
4044H	JP SAVEOB	Save a file to disk

DOS SUBROUTINES

PWRON

Turn ON the power of the drive selected in DOS vector IY+DK. To turn ON drive 1 , 10H should be written to IY+DK. To turn ON drive 2, 80H should be written to IY+DK before calling PWRON.

Entry parameter: None
Exit parameter: None
Registers affected: A

PWROFF

Turn OFF the power to the disk. Both disks are turned OFF with the write request line set high at the same time.

Entry parameter: None
Exit parameter: None
Registers affected: A

ERROR

This subroutine reads the content of register A and prints the .error message before going back to BASIC.

Entry parameter: Error code in A
Exit parameter: None
Registers affected: The subroutine will re-initialize the BASIC pointers and jump to BASIC.

ERROR CODE	ERROR
0	No error
1	Syntax error
2	File already exists
3	Directory full
4	Disk write protected
5	File not open
6	Disk I/O error
7	Disk full
8	File already open
9	Sector not found
10	Checksum error
11	Unsupported device

12	File type mismatch
13	File not found
14	Disk buffer full
15	Illegal read
16	Illegal write
17	Break

RDMAP

Read the track map from the disk and place it.,into the address pointed to by IY+MAPADR.

Entry parameter:	Disable interrupt
Exit parameter:	Error code in A
Registers affected:	A, BC, DE, IIL

CLEAR

Clear the sector specified in IY+TRCK and IY+SCTR.

Entry parameters:	Disable interrupt Track number in IY+TRCK Sector number in IY+SCTR
Exit parameter:	Error code in A
Registers affected:	A, BC, DE, HL

SVMAP

Save the track map in the address pointed by IY+MAPADR to track 0 sector 15 of the disk.

Entry parameter:	Disable interupt
Exit parameter:	Error code in A
Registers affected:	A, BC, DE, HL

INIT

Initialize a blank disk.

Entry parameter:	None
Exit parameter:	None
Registers affected:	A, BC, DE

CSI

This subroutine reads the user specified filename and puts into IY+FNAM if the syntax is correct.

Entry parameter:	Input message pointed to by HL
Exit parameter:	Error code in A
Registers affected:	A, BC, HL

HEX

This subroutine converts 4 bytes of ASCII pointed to by HL into DE reg pair.

Entry parameter:	HL points to 4 bytes of ASCII
Exit parameters:	Carry=l if error found, DE invalid
	Carry=O if no error, DE=2 bytes of HEX HL advanced by 4
Registers affected:	A, DE, HL

IDAM

Search for the identification address mark (IDAM) of the disk.

Entry parameters:	Desired track in IY+TRCK
	Desired sector in IY+SCTR
	Disable interrupt
Exit parameter:	Error code in A
Registers affected:	A, BC, DE, HL

CREATE

Generate an entry in the directory.

Entry parameters:	File name in IY+ENAM
	File type in IY+TYPE
	Disable interrupt
Exit parameter:	Error code in A
Registers affected:	A, BC, DE, HL

MAP

Search for an empty sector in the track map.

Entry parameter:	Track map in buffer pointed to by IY+MPADR
Exit parameters:	Error code in A
	Next sector available in IY+NSCT
	Next track available in IY + NTRK
Registers affected:	A, BC, DE, HL

SEARCH

Search for matching of filename in IY+FNAM with that in the directory.

Entry parameters:	Disable interrupt.
	File name in IY + FNAM
Exit parameter:	Error code in A
Registers affected:	A, BC, DE, HL

FIND

Search for an empty space in the directory.

Entry parameter:	Disable interrupt
Exit parameter:	Error code in A
Registers affected:	A, BC, DE, HL

WRITE

Write the content of the buffer pointed to by IY+DBFR to the track#, sector# specified by IY+TRCK and IY+SCTR.

Entry parameters:	Track number in IY+TRCK
	Sector number in TRK+SCTR
	Data to be written in buffer pointed to by IY+DBFR (128 bytes)
Exit parameter:	Error code in A.
Registers affected:	A, BC, DE, HL, BC', DE', HL'

READ

Read the content of track#, sector# specified by IY+TRCK and IY+SCTR into the buffer pointed to by IY+DBFR.

Entry parameters:	Track number in IY+TRCK
	Sector number in IY+SCTR
	Disable interrupt
Exit parameter:	Error code in A
	Read data in buffer pointed to by IY+DBFR (128 bytes)
Registers affected:	A, BC, DE, HL

DLY

Delay N mS specified by B.

Entry parameters:	Disable interrupt
	Number of mS to be delayed in B
Exit parameter:	None
Registers affected:	A, BC

STPIN

St - ep the stepper N tracks inwards specified by register B.

Entry parameters:	Disable interrupt
	Number of tracks to be stepped in B.
Exit parameter:	None
Registers affected:	A, BC

STPOUT

Step the stepper N tracks outwards specified by register B.

Entry parameters:	Disable interrupt
	Number of tracks to be stepped in B.
Exit parameter:	None
Registers affected:	A, BC

DKLOAD

Load the file specified in IY+FNAM to the memory.

Entry parameters:	Disable interrupt
	Filename in IY+FNAM

Exit parameters:	Error code in A. File in memory
Registers affected:	A, BC, DE, HL

SAVEOB

Save the filename	specified in IY+FNAM and pointed to by 78A4H to the disk.
Entry parameters:	Disable interrupt Filename in IY+FNAM File start address in 78A4H File end address in 78F9H File type in IY+TYPE
Exit parameter:	Error code in A
Registers affected:	A, BC, DE, HL, BC', DE', HL'