Memotech Owners Club Public Domain Software Library
Document LL02 - VDP Chip explained

Last month I included details of the RST 10 commands, well John Hodgson has included some more details which will, amongst other things solve the problems with using the SPRITE command through RST 10. This information can be found in the Letters pages.

I would like to demonstrate that although RST commands appear to be easy to use (once you have got the hang of them°°) they are in fact still not making the full use of the machines speed. There are following, two programs which do approximately the same thing, they fill the screen with `*'s. The first is a RST 10 version, you'll notice how short it is, it is faster than basic but not as fast as the second program, this program accesses the VDP and VRAM directly and so is of optimum speed.

```
;
; RST 10 Screen fill routine.
; CLS Before entry
          LD B,48
LOOP:     RST 10
          DB £94,"********************"
          DJNZ LOOP
          RET
;
; Alternative Screen fill routine
;
          LD DE,7168;TOP L/H CORNER OF TEXT SCREEN
          CALL VSET
          LD IX,960 ;LOOP COUNTER
LOOP:     LD E,42   ;NUMERICAL VALUE OF "*"
          CALL VOUT
          DEC IX
          PUSH IX
          POP BC
          LD A,B
          CP 0
          JP NZ,LOOP
          LD A,C
          CP 0
          JP NZ,LOOP
          RET
VSET:     PUSH AF
          LD A,E    ;SET UP VRAM ADDRESS POINTER FOR DATA OUT
          OUT (2),A
          LD A,D
          OR 64     ;SET WRITE TO VRAM MODE
          AND 127   ;
          OUT (2),A
          POP AF     CONT'D OVERLEAF
          RET
VOUT:     PUSH AF
          LD A,E    ;OUTPUT BYTE TO SCREEN
```

```
        OUT (1),A
        POP AF
        RET
```

To type these programs in all you have to do is enter into assembler (ASSEM 10 éRETç éRETç).
With both programs it is probably best to use the CLS command to clear the screen before the routine and use the PAUSE command to enable you to see whats happened.

It is the second program that requires some explaining as it contains several interesting points which will go someway to help with the understanding of the machine and Z80 assembler.

The first line contains the address of the first location on the screen (7168 Dec or 1C00 Hex).You may or may not know that the screen is memory mapped, that is each screen location corresponds to a memory location, if the manual had a block diagram of the VRAM you could see what I mean, but i'm afraid it does not, so this makes an explanation impossible at this stage. Anyway, onto the routine called VSET, this sets the VDP into write mode and sets the address for the write. The Truth Table below shows the two possible modes:-

```
   Bit  6  ;  7
       -----------
        1  ;  0  Write data to VRAM
        0  ;  0  Read data from VRAM
```

VDP address's are 14 bits long leaving the two most significant bits (above) for the mode setting.It should be noted that the VDP register loaded with the `write' start address is an auto incrementing register so enabling sequential data transfers. The bits 6 & 7 are set to their correct values by the lines OR 64 & AND 127, these are two logical operators and they perform the following tasks. Two Truth Tables and a demo will hopefully clarify things somewhat:-

```
      0 AND 0 = 0          0 OR 0 = 0
      0 AND 1 = 0          0 OR 1 = 1  TRUTH TABLES FOR LOGICAL OPS
      1 AND 0 = 0          1 OR 0 = 1  OR & AND
      1 AND 1 = 1          1 OR 1 = 1
```

As a demo i'll use the numbers from the program:-

```
Accumulator   00011100 = 1C Hex
OR 64         01000000 = 64 Dec
              --------
Accumulator   01011100
AND 127       01111111 = 127 Dec
              --------
              01011100
```

You can see that bits 6 & 7 (two left hand bits) are now set as required.

VOUT only performs a bit output the the VRAM address set by VSET. The IX register is used as a counter to output 960 (24*40) "*" characters to the screen. I hope that this is fast enough for you°°°°.

Any machine code enthusiast who has tried to understand the operation of the TMS9918 Video Processor and it's associated 16K video ram from the description written in the MTX Users Manual may appreciate how confused I felt after reading it for the first time. This article attempts to provide a clearer and more practical approach to using and understanding the processor and it's VRAM.

Memory Map

The VRAM memory is mapped by BASIC as shown below, this `map' is for both text and graphics. Graphics Mode 2 as the manual says°

Address in Decimal	
16255	; End of Sprite Attribute table
"	
16128	; Start of Sprite Attribute table
16127	; End of Pattern Name Table
"	(Graphics Display)
15360	; Start of Pattern Name Table
15359	; End of Sprite Generator Table
"	
14336	; Start of Sprite Generator Table
14335	; End of Pattern Colour Table
"	
8192	; Start of Pattern Colour Table
8191	; End of Text Name Table
"	(Text Display)
7168	; Start of Text Name Table
7167	; End Text Pattern Library
"	
6144	; Start Text Pattern Library
6143	; End of Pattern Generator Table
"	
0	; Start of Pattern Generator Table

Table Showing How VRAM is Mapped By Basic

Your'e probably still saying what does it all mean, well, starting with text mode (6144 to 8191), this 2K block of memory takes care of text mode. In text mode the screen measures 40*24 characters, that's 960 characters in all, starting at 7168 which represents the top left hand corner of the text screen, each screen location corresponds to a memory location, hence the Text Name Table is 1K long. Held in these memory locations is the ASCII number of the actual character, ie. if the 10th location along from the top (7178) contained number 31 then the screen location 10 across from the top would show a "1".(For a table of ASCII characters see your manual page 174, Appendix 1). Incidentally the 128 ASCII characters are stored in the Text Pattern Library, each entry in the library takes 8 bytes, therefore, the library is 1K long. Notice also that the ASCII characters are stored in the order of the table in your manual and so placing a 31 in a display location causes the processor to look a the 31st entry in the Text library and print that pattern to the screen.

The graphic's display is laid out in a similar way to that of the text display except more memory is needed and there is more attention paid to detail.

In graphics mode (mode 2 in the manual) the screen is divided into three sections each of which has 256 pattern positions, each pattern position is capable of displaying it's own unique graphic character as defined by the programmer. Each pattern is made up in a 8 bit by 8 byte grid the same as for sprites.

To make things a little clearer let us look how patterns are positioned on the top 1/3 of the screen. Firstly the patterns are defined on the 8 * 8 grid and then are entered in their hex format into the Pattern Generator Table starting at 0K. Thus if patterns x,y and z are defined then they will occupy a total of 24 bytes from £00 to £18.

The choice of pattern and it's position on screen is determined by the contents of the associated Pattern Name Table, which in this case starts at 15360 Dec and is 768 (24*32) bytes long. As you can see this is much the same as for the Text Screen. It should be pointed out however that patterns defined for one 1/3 of the screen may not be used for another area of the screen unless they are defined in the associated Pattern Table.

Ink and Paper colours for the defined patterns are set by the contents of the 3 1/3rd's of the Pattern Colour Tables, again one table for each 1/3 of the screen. Each byte in the colour table is directly related to the byte entries in the pattern table. Ink colour is determined by the contents of the most significant half of each byte and Paper colour by the contents of the least significant half. It follows therefore, that each byte of a pattern definition may have it's own ink and paper colours.

Next month I'll explain how the sprite generator table and the sprite attribute table are set up by basic and then move on to explain how the VDP's registers are set up by Basic.

Many thanks to Paddy Thompson for the help that he has provided in the writing of this article°°°

Continued from last months article about VRAM mapping from Basic, we still have the Sprite Attribute and Generator tables to look at.

Well, these obviously take care of the Sprites used by the GENPAT command in Modes 4&5, it can be seen that the Sprite generator table is 1K long (see last months mag.) thus allowing room for

128 size 0 sprites(8*8)ä8*128 bytesü or 32 size 1 sprites (16*16)ä32*4*8 bytesü.

The Sprite Attribute table is 128 bytes long and controls the 32 sprite planes, each sprite plane is controlled by 4 bytes, the format for these looks like this :

Byte

1 This byte contains the value which is the number of pixels from the top left hand corner of the sprite to the top of the screen.

2 Contains the value which is the number of pixels from the top left hand corner or the sprite to the left of the screen.

3 The contents of the third byte determine which shape the sprites will be and is selected from one of the pre-defined shapes in the sprite generator table.

4 The lower four bits of the fourth byte select the sprite colour and the most significant bit of this byte may be set to allow the sprite to 'bleed' in from the left hand side of the screen.

Knowing now a little of what a Pattern Table, Generator Table and Attribute table are, you should be able to make some sense of the Technical data on the VDP in the black manual. You should also appreciate the effects of such Basic commands as:- CTLSPR, GENPAT,ADJSPR etc..

The only other thing that we have not talked much about is the VDP's 8 Write Only Registers.(See Table 2 :VDP Registers, Pg221 of manual). These registers contain all the information necessary to form the VRAM table, Basic sets these on 'start up' to conform to Mode 2 graphics, it is possible to set up these as you wish and use them from assembler. The manual describes the VDP being used in Mode 1 & Multicolour mode, this is possible and with time and care could produce some interesting results, however all is not so simple as it is necessary to disable Basic otherwise this will corrupt your efforts. This is done by making your first assembler command DI (Disable Interrupt) and your last, on return to basic RETI (Return from Interrupt). Another way of returning to basic is to make your last line JP £0000, this being called a 'warm boot' back to Basic in computer jargon.

It is totally safe to muck about with the VDP in this way so feel free to try anything, the worst that can happen is that the computer will 'hang-up'.

The technique for setting up the VDP registers is as follows:-
All register set-ups take place via port 2 in two stages. The Z80 `D' and `E' registers are pre loaded with the register number and the data for that register respectively, then the data is output first to port 2 followed by the register number also to port 2. It should be noted that before the second write (the register number) the most significant bit (7) must be set to '1' and bits 6,5,4 and 3 must be set to '0'. A simple program which does this would look like this :

```
; ROUTINE TO SET UP VDP REGISTERS
; DE REGISTER LOADED WITH DATA AND REGISTER ON ENTRY
  (DO NOT Type in the above lines, they are for reference only)

VDPREG: PUSH AF      ;SAVE REGISTERS
        PUSH BC
        LD A,E       ;LOAD ACCUMULATOR WITH DATA
        OUT (£02),A;OUTPUT DATA
        LD A,D       ;LOAD ACCUMULATOR WITH REGISTER NO.
        AND 7        ;SET UP CORRECT CONTROL BITS
        OR 128       ;SET MSB TO '1'
        OUT (£02),A;OUTPUT REGISTER NUMBER
        POP BC
        POP AF       ;RESTORE REGISTERS
        RET
```

Using this type of format it is possible to manipulate the VDP as you wish, remember that :

Port 2 is used for Address transfers
i.e. The registers, including the auto-incrementing address register mentioned last month.
Port 1 is used for Data transfers
i.e. Accessing the Vram.

Also, address set ups and data transfers require a certain minimum amount of time between processes, this is 11 micro seconds between address set ups and 8 micro seconds between data transfers. (Not long enough to go and make a cup of tea°°)

It is also possible to change from Text Mode to Graphics Mode with alarming speed as only two Registers have to be altered. The alterations are as follows :
(See pages 221 & 222 of black manual)
Bits M1,M2 and M3 control the Mode of operation and simply changing these using the above program will change you from Text to Graphics.

M1	M2	M3	
0	0	1	Graphics mode 2
1	0	0	Text Mode

Thats about how simple the VDP is, probing about with simple little routines will help with understanding these things even fuller.
MOCPDSL - 01/04/88 LL02