Power Graphics

for

Memotech, MSX & Einstein Computers

by

AFW Software

CONTENTS

Trademarks

The following references &/or trademarks are acknowledge:

IBM	- International Business Machines
CPM	- Digital Research
MSX	- Microsoft
MTX	- Memotech Computers Ltd
TMS	- Texas Instruments
Z80A	- Zilog Corporation
Einstein	- Tatung

Note:

MTX Advanced Referenced Manual was updated to include MTX, MSX and Einstein computers and was renamed as VDP Discovered manual.

Chapter 1: The Beginning

1.0 Introduction

The objectives of this manual are too:

a) provide a suitable Z80 based graphics kernel for the VDP.
b) be compatible with the MTX, MSX and Einstein computers.
c) have the subroutines in a informative and simple format that they are self explanatory and ready to be used.
d) form a basis of a methodology, so that users design code that follows this easy to use format.

I have not provided too many graphic commands because I felt that the best way to learn a subject is to have hands on experience. However, the graphics kernel in chapter 2 does include all the key building block commands like : PLOTXY , DRAWLINE, GRPINK ,GRPPAPER, BORDER, SCRCLS. From this it is possible to design any command ie, Drawing a box , colouring a box , drawing a triangle ,these have been included as examples in how to use the building blocks. For in depth study on graphics, sound ,RAM and ROM, text and keyboards, I refer you to the "VDP Discovered" Manual by AFW Software.

Throughout this manual the prefix # is used to signify that the number following it is a hexadecimal number,ie #9000. However, some other computers/compilers use the ampersand sign to prefix hex numbers,ie &A000. Some other systems use the postfix H to indicate a hex number,ie DF99H. Use the nomenclature that your system uses.

1.1 Overview of the Graphics Chip

1.1.1 VDP and VRAM

The Video Display Processor, VDP , as used in the Memotech MTX series, the Tatung Einstein and the MSX microcomputers is the Texas Instruments TMS 9918/19/29 or compatible. The VDP is a dedicated graphics processor with eight write only registers [R0 - R7] and one read only register [R8]. These registers have specific functions like selecting the screen mode, or positioning pattern or colour tables or what the INK and PAPER colours should be. It is not the aim of this manual to cover this,see "VDP Discovered" manual by AFW Software, chapter 2.

Unlike many other computer systems that use CPU RAM to store graphic information, the VDP comes with its own dedicated Video Ram, called VRAM. This special RAM block is 16384 bytes long. VRAM is used to store key graphical information like the text font or sprite shapes or other graphical patterns. It also stores the colour information.Most importanatly part of VRAM is configured as the screen. Figure 1, illustrates the relationship between the VDP/VRAM and CPU/RAM systems.

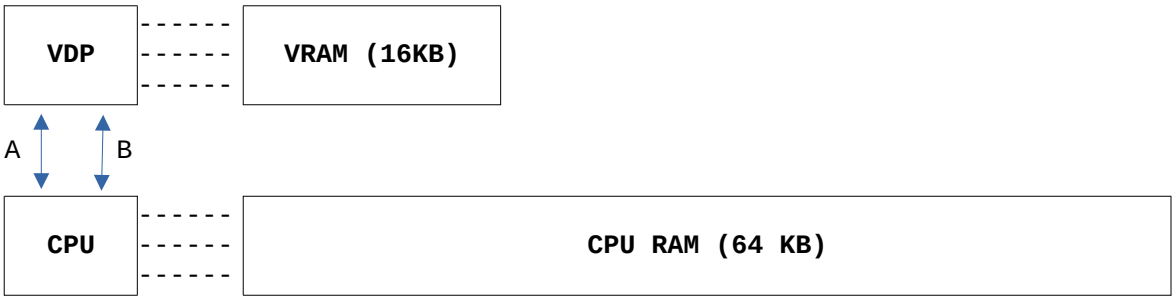

Figure 1: CPU/RAM and VDP/VRAM architecture

The advantages of a dedicated graphics processor with its own private RAM are threefold:

1. Programming space in CPU RAM is at a premium.

2. The VDP/VRAM architecture allows better and faster graphic manipulation as needed in arcade style games. The hardware sprites provide fast flicker-free animation.

3. Because the VDP is also a processor, just like the CPU, it is able to control all the graphic,text & sprite functions without CPU help. The consequence of this is that the CPU is released to do other tasks and thus the speed of CPU processing is greatly improved , which may be crucial in number crunching.

1.1.2 CPU-VDP Communication

The CPU communicates with this external graphics processor using two of the 256 possible IN/OUT mapped ports on the Z80A CPU. These ports or channels , as illustrated in figure 1, allow two-way communication between the CPU and VDP. Communication is only needed when access to the screen is required ,ie in order to read the screen for screen dumps or to write to the screen, ie to change to different fonts or sprites. Table 1, details the port numbers used by the CPU to communicate with the VDP on the Memotech, MSX and Einstein. Subroutines V1,V2 and V3 are the most important subroutines for communication between the VDP/VRAM and CPU/RAM, see section 1.2.2 .

Table 1: The ports used by the Memotech,MSX and Einstein Z80A microcomputers for VDP-CPU communication. Where r/w is for reading or writing data transfers.

Computer	**port A (data r/w)**	**port B (addressing)**
Memotech	#01	#02
Einstein	#08	#09
MSX	#98	#99

1.1.3 Screen Mode - G2

The VDP provides the programmer with 4 predefined screen modes: TEXT, Graphics 1 and 2 and MultiColour. The latter 3 screen modes can display upto 32 hardware sprites. This allows fast flicker-free animation with very little programming effort. However, this manual is only concerned with Graphics mode 2 (G2). For detailed discussion of all 4 modes , see "VDP Discovered" manual by AFW Software , chapters 2 to 7.

G2 or bit mapped or high resolution graphics mode allows the programmer the flexibility to address (access) every dot or point or pixel on the screen. This access is essential for serious graphic applications like CAD and DTP, and for normal graphic work like plotting points and drawing lines.

The VDP has a screen resolution of 256 by 192 dots. Therefore, we have the ability to plot upto 49152 (256*192) dots or bits of graphical information. Notice the use of the analogy between dots and bits. The Z80A and the VDP both prefer to handle 8 bits of data at a time, or in byte wise chunks. Thus, every 8 linear dots of graphical information is stored as one byte of screen data in the VRAM Pattern Generator Table (PGT). The PGT acts as the high resolution graphics screen in G2 mode. The PGT is 6144 bytes in length (49152/8) and is positioned in VRAM at 0000 to 6143.

1.1.4 Colour

The VDP provides the user with a colour palette of 16 , numbered 0-15, see table 2. As already stated, each dot on the screen can be either on (1) or off (0). All the ON dots will be set to the INK colour and all the OFF dots will be set to the PAPER colour. A palette of 16 colours can be represented by 4 bits as 2^4 equals 16, see table 2. Both the INK and PAPER colours use this representation.

Table 2: The 16 colour palette on the VDP in G2 mode.

Colour	Bit Pattern	Hexadecimal	Decimal
Transparent	0 0 0 0	#00	00
Black	0 0 0 1	#01	01
Medium Green	0 0 1 0	#02	02
Light Green	0 0 1 1	#03	03
Dark Blue	0 1 0 0	#04	04
Light Blue	0 1 0 1	#05	05
Dark Red	0 1 1 0	#06	06
Cyanate	0 1 1 1	#07	07
Medium Red	1 0 0 0	#08	08
Light Red	1 0 0 1	#09	09
Dark Yellow	1 0 1 0	#0A	10
Light Yellow	1 0 1 1	#0B	11
Dark Green	1 1 0 0	#0C	12
Magenta	1 1 0 1	#0D	13
Grey	1 1 1 0	#0E	14
White	1 1 1 1	#0F	15

Therefore, one byte (8 bits) can hold two colours. The VDP uses a particular colour byte format to represent both the INK and PAPER on the screen,see figure 2.

```
INK   = #01            |<-- msn INK -->|<- lsn PAPER ->|
      = black          ---------------------------------
PAPER = #09            : 0 : 0 : 0 : 1 : 1 : 0 : 0 : 1 :
      = Light Red      ---------------------------------
```

Figure 2: The Colour byte format. Note the actual colour byte value for the above is 16*(INK) + PAPER = 25 or #19. Where msn/lsn = most/least significant nybble.

As shown in figure 2, one byte holds both the INK (msn) and PAPER (lsn) information. This colour byte sets the colour for 8 linear screen dots. The VDP can only resolve 2 colours per byte (or 8 dots) of graphical information. Therefore, the dot resolution is BIT mapped but the colour resolution is only BYTE mapped.

If it was possible to allow full individual colour addressing, for every dot on the screen, ie for every dot there was a corresponding separate colour byte ,from a palette of 16, would require 49152 bytes for the 49152 bits of screen data. This is physically impossible because the VDP is a 14 bit microprocessor, ie it can only access 2^14 or 16384 bytes. Therefore, one of the constraints of the system was to limit the colour resolution. Since only one colour byte (contains both INK & PAPER) per 8 screen bits, the colour table is only 6144 (49152/8) bytes long.

By a matter of coincidence!! this is the same length as the PGT. This makes life a lot easier when trying to colour the correct dot on the screen. When x,y coordinates are converted to VRAM locations (addresses) means that a dot can only be pinpointed to the nearest VRAM byte address. Which bit in the in the byte is also calculated and the desired bit is set to on (1). Well, ignoring the latter part, this VRAM address or offset (since the PGT starts at zero) will correspond to a similar displacement in the Colour Table. In actual fact ,the correct colour byte will be + 8192 further forward, because the colour table starts at address 8192 in VRAM.

1.2 The VDP Subroutines & Listing Format

1.2.1 The Format

This manual uses a standard listing format that should help improve program design and subsequent documentation. What I mean by helping in program design, is that when you use any of the included subroutines, you will know what data variables that are needed, what other subroutines that are needed to run the desired subroutine and most importantly of all, demos are provided to remind you , how the subroutine is to be used in a program. Even, if you only use a subroutine ,once in a blue moon, you will always know what the code does , what the code is and how to use the code constructively. The format is described below and for examples see 1.2.2 and chapter 2.

Each program is assigned a simple code number for quick indexing, ie V1 , V2 for VDP subroutine 1 or 2 , and a title. The title is the subroutine name, ie PLOTXY. Where possible I have tried to use names that correspond to BASIC commands with a similar syntax or structure.

Lines / Bytes / Stacksize and Datasize are a summary of the actual code. The Stacksize is the amount of RAM reserved to hold any PUSHed registers or CALL addresses used in the subroutine. Every PUSH or CALL counts as two bytes. The Datasize is the reserved RAM needed to hold static data variables, a bit like variables in BASIC ,ie LET a=10 .

Language tells the user what language the listing was written in, ie Z80A or PASCAL or C or BASIC or etc. Any specific Hardware that the code refers to,is noted , ie VDP (graphics & text) or PSG (sound) or PRT (printer port) or etc. Lastly on this line,we say whether error checking has been included in the subroutine or not. The error checking may be a check to see if a note is out of the sound range or to stop a piece of text running over the edge of the screen or etc.

Machine, notes which computers this piece of code can be run on, ie MSX or MTX or Einstein or CPM or IBM or etc. The Description is a brief note on what the subroutine does and any other notes needed for running the code.

For instance, Z80 register A ,must hold the ink colour before calling GRPINK. Details of the subroutine design, for instance, what the algorithm used in the subroutine was, should be referenced, see later.

The Data Variables and Service Subroutines go hand in hand. The service subroutines are the subroutines that this subroutine requires in order to function properly. The data variables are all variables used, this includes those for the service subroutines and the current subroutine. The data variables are reserved RAM locations which the subroutines uses to store data. The order of these data variables in RAM is very important. The data variables should be read from left to right and from top to bottom. For Example:

```
LSBPGTADDR: DS 1  MSBPGTADDR: DS 1
COLOUR:    DS 1
```

this should be read thus:

```
LSBPGTADDR: DS 1
MSBPGTADDR: DS 1
COLOUR:     DS 1
```

and NOT:

```
LSBPGTADDR: DS 1
COLOUR:     DS 1
MSBPGTADDR: DS 1
```

The reason for this is when the Z80 register pair HL, reads the location LSBPGTADDR, HL is actually storing the contents of the location LSBPGTADDR in L and MSBPGTADDR in H. HL now holds a PGT address in the Z80 LSB/MSB format. However, in the second case, the colour is taken as the MSB of the PGT address and the VRAM pointer is positioned at the wrong address. This will cause a BUG in your program with unknown consequences so BEWARE.

When the current subroutine is eventually run, none or some or all of the registers (as used in Z80 or C or VDP or PSG) may be changed from its initial value. This may be important and in many situations is the primary cause of program crashes. The Reference section is where , I have tried to indicate where further reading on a particular subroutine can be obtained. The reference usually includes algorithms , figures and text to help explain the subroutine.

I have included a "How to use" section because in a few days or weeks or months time you will have forgotten all about this subroutine and may need to be reminded of its function or purpose and how you can use it in your listings. Finally the Code. The code is the actual subroutine code. For Z80 users I haven't included an ORG address, I leave that upto you where you want the code to be placed. Memotech users can use the built in Z80 assembler if so desired. No ORG is needed in this case.

1.2.2 VDP Subroutines

V1. Title: VDPREGSET8 (Configure VRAM for G2 mode)

Lines = 11 Bytes = Stacksize = 0+2 Datasize = 8

Language = Z80A Hardware = VDP Error Checking = No

System = MSX , Memotech (MTX) and Einstein

Description:

This subroutine loads the 8 VDP registers with the appropriate data to configure VRAM to G2 mode. This involves positioning the PGT at 0000 and the Colour Table at 8192. For a more detailed description, I refer you to the " VDP Discovered" manual by AFW Software. Also, register HL must be pointing to the G2DATA, prior to calling VDPREGSET8.

Data Variables:

```
G2DATA: DB #02,#C2,#0F,#FF,#03,#7E,#07,#F5 ; MTX G2 mode
```

or

```
G2DATA: DB #02,#C2,#0F,#FF,#03,#76,#03,#F4 ;Einstein G2 mode
```

or

```
G2DATA: DB #02,#C2,#06,#FF,#03,#36,#07,#F5 ; MSX G2 mode
```

Service Subroutines:

None

Registers Altered:

None

How to use:

```
INITVDPREG: LD HL,G2MODE            ; could use MTXG2 or EING2 or MSXG2
            CALL VDPREGSET8
            RET
```

Reference:

"VDP Discovered", chapter 2

Code:

```
VDPREGSET8: LD BC,#0800
REGWRTVDP:  LD A,(HL)
            OUT (#02),A       ; #09 =EINSTEIN    #99 =MSX
            LD A,C
            OR #80
            OR #40
            OUT (#02),A       ; #09 = EINSTEIN   #99 =MSX
            INC C
            INC HL
            DJNZ REGWRTVDP
            RET
```

V2. Title: VDPWRTSEL (Writing to VDP/VRAM)

Lines = 8   Bytes =   Stacksize = 2+2   Datasize = 0

Language = Z80A   Hardware = VDP   Error Checking = No

System = MSX , Memotech (MTX) and Einstein

Description:

Whenever, the CPU is required to write information to VRAM,it would need to tell the VDP to select VRAM WRITE mode and then it would pass the appropriate VRAM addresses to be written to. Therefore this function is needed for all CPU-VDP address transfers. The VRAM address will be held in register HL prior to calling VDPWRTSEL.

Data Variables:

None

Service Subroutines:

None

Registers Altered:

None

How to use:

Refer to chapter 2 of this manual,there are many examples.

Reference:

"VDP Discovered", chapters 2 - 7

Code:

```
VDPWRTSEL:  PUSH AF
            LD A,L
            OUT (#02),A       ; #09 =EINSTEIN    #99 =MSX
            LD A,H
            OR #40            ; tell VDP to go into WRITE mode
            OUT (#02),A       ; #09 =EINSTEIN    #99 =MSX
            POP AF
            RET
```

V3. Title: VDPREADSEL (Reading from VDP/VRAM)

Lines = 8   Bytes =   Stacksize = 2+2   Datasize = 0

Language = Z80A   Hardware = VDP   Error Checking = No

System = MSX , Memotech (MTX) and Einstein

Description:

Whenever, the CPU is required to READ information from VRAM, it would need to tell the VDP to select VRAM READ mode and then it would pass the appropriate VRAM addresses to be read from. Therefore this function is needed for all CPU-VDP address transfers. The VRAM address will be held in register HL prior to calling VDPREADSEL.

Data Variables:

None

Service Subroutines:

None

Registers Altered:

None

How to use:

Refer to chapter 2 of this manual,there are many examples.

Reference:

"VDP Discovered", chapters 2 - 7

Code:

```
VDPREADSEL: PUSH AF
            LD A,L
            OUT (#02),A       ; #09 =EINSTEIN    #99 =MSX
            LD A,H
            AND #3F           ; tell VDP to go into READ mode
            OUT (#02),A       ; #09 =EINSTEIN    #99 =MSX
            POP AF
            RET
```

1.3 Worked Example

As a postscript to this chapter, I thought I would provide you with a short demo of one of the commands in action. Using an ASCII editor, type in the listing below. Compile at the desired ORG address. Memotech users can use the inline Z80 assembler instead,remember no need to specify the ORG address, use a BASIC line of 40. Once the code has been successfully checked and compiled, save to disk or tape as: DEMOCAD .

Now reload the desired code into the appropriate RAM address. The Memotech inline assembler versions will be reload as if a BASIC listing. Now insert the following code to run the demo:

```
10 ORG=#9000 :    REM &9000 or 9000H  or use decimal 36864
20 DEF USR(ORG) : REM some systems use RAND USR (ORG)
30 GOTO 30 : REM some computers run the code & default to TEXTSCR
40 STOP
```

type RUN

For users of the MTX inline assembler use:

```
LOAD "DEMOCAD"

10 GOSUB 40
20 GOTO 20
30 STOP
40 CODE : REM the listing would be here
50 RETURN

ASS.40
```

<CLS> <RET> to exit. The inline assembler will move the code up to take into account the BASIC text before it.

type RUN

I hope you find this manual very useful.

```
ORG #9000              ; don't use with MTX built in assembler

; main program

            DI    ; disable BASIC interrupts

            LD HL,NEWSTACK
            EX (SP),HL        ; the stack pointer now at NEWSTACK
            PUSH HL           ; save old stack

            LD HL,MTXG2       ;  or EING2 or MSXG2
            CALL VDPREGSET8   ; set to G2 mode

            LD HL,#4040       ; x1,y1 is 64,64
            LD (X1),HL
            PUSH HL
            LD HL,#8080       ; x2,y2 is 128,128
            LD (X2),HL
            PUSH HL

            LD A,15           ; white ink
            CALL GRPINK
            LD A,5            ; blue paper
            CALL GRPPAPER

            CALL DRAWBOX

            LD A,1            ; black ink
            CALL GRPINK

            POP HL            ; restore x2,y2
            LD (X2),HL
            POP HL            ; restore x1,y1
            LD (X1),HL

            CALL FILLBOX

            POP HL            ; restore old stack
            LD (SP),HL

            EI                ; enable interrupts

            RET               ; exit program


; subroutines from library

PLOTXY:     PUSH AF
            PUSH BC
            PUSH HL
            CALL CARTXYTOVRAM
            LD C,A
            LD A,(BITNUMBER)
            LD HL,DOTVALUE
            ADD A,L
            LD L,A
            LD A,(HL)
```

```
            OR C
            LD HL,(LSBPGTADDR)
            CALL VDPWRTSEL
            OUT (#01),A       ; #08 =Einstein  #98 =MSX
            LD BC,8192
            ADD HL,BC
            CALL VDPWRTSEL
            LD A,(COL)
            OUT (#01),A       ; #08 =Einstein  #98 =MSX
            POP HL
            POP BC
            POP AF
            RET


CARTXYTOVRAM:PUSH BC
            PUSH HL
            XOR A
            LD A,(X)
            SRL A
            SRL A
            SRL A
            AND A             ; INT (X/8)
            SLA A
            SLA A
            SLA A             ; INT(X/8)*8
            LD C,A
            XOR A
            LD A,(Y)
            SRL A
            SRL A
            SRL A
            AND A             ; INT(Y/8)
            LD B,A
            LD A,23
            SUB B             ; 23-INT(Y/8)
            LD H,A
            LD (MSBPGTADDR),A
            LD A,(Y)
            AND 7
            LD B,A
            LD A,7
            SUB B
            ADD A,C
            LD L,A
            LD (LSBPGTADDR),A
GETBITNUM:  LD A,(X)
            AND 7
            LD (BITNUMBER),A
            CALL VDPREADSEL
            IN A,(#01)        ; #08 =Einstein  #98 =MSX
            POP HL
            POP BC
            RET
```

```
DRAWBOX:    PUSH HL
            PUSH DE
            PUSH AF
            LD HL,(X1)
            LD DE,(X2)
            PUSH HL
            PUSH DE
            LD D,H
            LD (X2),DE
            CALL DRAWLINE     ; x1,y1 to x2,y1
            EX DE,HL
            LD (X1),HL
            POP DE
            LD (X2),DE
            CALL DRAWLINE     ; x2,y1 to x2,y2
            POP HL
            PUSH HL
            LD H,D
            LD (X1),HL
            CALL DRAWLINE     ; x1,y2 to x2,y2
            POP DE
            EX DE,HL
            LD (X2),DE
            LD (X1),HL
            CALL DRAWLINE     ; x1,y1 to x1,y2
            POP AF
            POP DE
            POP HL
            RET
FILLBOX:    PUSH HL
            PUSH DE
            LD HL,(X1)        ; ASSUMES BOX IS DRAWN FIRST.
            LD DE,(X2)
            PUSH BC
            PUSH DE
            PUSH HL
            INC H
            DEC L
            DEC D
            INC E
            LD A,H
            CP D
            JR C,DGTH
            PUSH AF
            SUB D
YCOUNT:     LD B,A
            POP AF
FBOXLOOP:   LD (Y1),A
            LD (Y2),A
            PUSH AF
            CALL DRAWLINE
            POP AF
            DEC A
            DJNZ FBOXLOOP
            POP HL
            POP DE
            LD (X1),HL
```

```
            LD (X2),DE
            POP BC
            POP DE
            POP HL
            RET

DGTH:       LD A,D
            PUSH AF
            SUB H
            JR YCOUNT

DRAWLINE:   PUSH HL
            PUSH DE
            PUSH BC
            CALL INITVARS
            CALL SETINCX
            CALL SETINCY
            CALL GETDIST
            CALL DLINE
            POP BC
            POP DE
            POP HL
            RET

INITVARS:   XOR A
            LD (YERR),A
            LD (XERR),A
            LD (SGNX),A
            LD (SGNY),A
            LD HL,(X1)
            LD DE,(X2)
            LD A,E
            CP L
            JR NC,SXFLAG
            LD A,1
            LD (SGNX),A
            LD A,L
            LD L,E
SXFLAG:     SUB L
            LD (DX),A
            LD A,D
            CP H
            JR NC,SYFLAG
            LD A,1
            LD (SGNY),A
            LD A,H
            LD H,D
SYFLAG:     SUB H
            LD (DY),A
            RET

NEGX:       XOR A
            INC A
            JR SXFLAG
```

```
NEGY:       XOR A
            INC A
            JR SYFLAG

SETINCX:    LD BC,(DX)
            LD A,1
            CP B
            JR Z,SDX
            DEC A
            CP C
            JR Z,SDX
            INC A
SDX:        LD (INCX),A
            RET

SETINCY:    LD BC,(DY)
            LD A,1
            CP B
            JR Z,SDY
            DEC A
            CP C
            JR Z,SDY
            INC A
SDY:        LD (INCY),A
            RET

GETDIST:    LD A,(DY)
            LD B,A
            LD A,(DX)
            CP B
            JR C,DYGTDX
DXGTDY:     LD (DISTANCE),A
            RET
DYGTDX:     LD A,B
            LD (DISTANCE),A
            RET

DLINE:      LD HL,(X1)
            LD (X),HL
            XOR A
            LD B,A
DRAWLOOP:   LD A,(DISTANCE)
            INC A
            CP B
            RET C
            CALL PLOTXY
            LD A,(DY)
            LD C,A
            LD A,(YERR)
            ADD A,C
            LD (YERR),A
            PUSH AF
            LD A,(DX)
            LD C,A
            LD A,(XERR)
            ADD A,C
            LD (XERR),A
```

```
            PUSH AF
XERRGTDIST: LD A,(DISTANCE)
            LD C,A
            POP AF
            PUSH BC
            CP C
            JR C,YERRGTDIST
            JR Z,YERRGTDIST
            SUB C
            LD (XERR),A
            LD A,(INCX)
            LD C,A
            LD A,(SGNX)
            CP 0
            JR Z,ADDINCX
SUBINCX:    LD A,L
            SUB C
RETINCX:    LD L,A
YERRGTDIST: POP BC
            POP AF
            CP C
            JR C,UPDXY
            JR Z,UPDXY
            SUB C
            LD (YERR),A
            LD A,(INCY)
            LD C,A
            LD A,(SGNY)
            CP 0
            JR Z,ADDINCY
SUBINCY:    LD A,H
            SUB C
RETINCY:    LD H,A
UPDXY:      LD (X),HL
            INC B
            JR DRAWLOOP
            RET

ADDINCX:    LD A,L
            ADD A,C
            JR RETINCX

ADDINCY:    LD A,H
            ADD A,C
            JR RETINCY


GRPINK:     LD (INKCOL),A
            CALL GETCOL
            RET


GRPPAPER:   LD (PAPERCOL),A
            CALL GETCOL
            RET
```

```
GETCOL:     PUSH BC
            PUSH AF
            LD A,(INKCOL)
            AND #0F
            SLA A
            SLA A
            SLA A
            SLA A
            LD B,A
            LD A,(PAPERCOL)
            AND #0F
            ADD A,B
            LD (COL),A
            POP AF
            POP BC
            RET

VDPREGSET8: LD BC,#0800
REGWRTVDP:  LD A,(HL)
            OUT (#02),A       ; #09 =EINSTEIN    #99 =MSX
            LD A,C
            OR #C0
            OUT (#02),A       ; #09 =EINSTEIN    #99 =MSX
            INC C
            INC HL
            DJNZ REGWRTVDP
            RET

VDPWRTSEL:  PUSH AF
            LD A,L
            OUT (#02),A       ; #09 =EINSTEIN    #99 =MSX
            LD A,H
            OR #40
            OUT (#02),A       ; #09 =EINSTEIN    #99 =MSX
            POP AF
            RET

VDPREADSEL: PUSH AF
            LD A,L
            OUT (#02),A       ; #09 =EINSTEIN    #99 =MSX
            LD A,H
            AND #3F
            OUT (#02),A       ; #09 =EINSTEIN    #99 =MSX
            POP AF
            RET

; Data Variables:

X:          DS 1
Y:          DS 1
LSBPGTADDR: DS 1
MSBPGTADDR: DS 1
X1:         DS 1
Y1:         DS 1
BITNUMBER:  DS 1
X2:         DS 1
Y2:         DS 1
```

```
DOTVALUE:   DB 128,64,32,16,8,4,2,1
DX:         DS 1
SGNX:       DS 1
INCX:       DS 1
XERR:       DS 1
DY:         DS 1
SGNY:       DS 1
INCY:       DS 1
YERR:       DS 1
DISTANCE:   DS 1
INKCOL:     DS 1
PAPERCOL:   DS 1
COL:  DS 1
NEWSTACK:   DS 100            ; 14+8+22+18+24+10  +4 to round up
MTXG2:      DB #02,#C2,#0F,#FF,#03,#7E,#07,#F5
EING2:      DB #02,#C2,#0F,#FF,#03,#76,#03,#F4
MSXG2:      DB #02,#G2,#06,#FF,#03,#36,#07,#F5
```

page 19 is intentionally left blank.

Chapter 2: Graphics Kernel Subroutines (Z80)

G01. Title : PLOTXY (PLOT x,y)

Lines = 23 Bytes = Stacksize = 12+2 Datasize = 14

Language = Z80A Hardware = VDP Error Checking = No

System = MSX , Memotech (MTX) and Einstein

Description:

This subroutine requires that the x and y coordinates have previously been loaded into the appropriate data variables. Also that the x coordinate is in the range 0-255 and the y coordinate in the range 0-191. The VDP must be set to G2 mode (high resolution).

This subroutine plots a point at the desired x,y location according to the cartesian coordinate map ,ie origin at the bottom left hand corner of the screen , 0,0 . The point or pixel (short for picture element) will be drawn in the INK colour set as default or to the colour specified by you , see G11.

Data Variables:

```
X: DS 1    Y: DS 1    LSBPGTADDR: DS 1   MSBPGTADDR: DS 1
COL: DS 1   BITNUMBER: DS 1
DOTVALUE: DB 128,64,32,16,8,4,2,1
```

Service Subroutines:

CARTXYTOVRAM ; VDPWRTSEL ; VDPREADSEL

Registers Altered:

None

How to Use:

```
LD HL,#80A0             ; H=#80 (128)  and  L=#A0 (160)
LD (X),HL               ; H is loaded into Y and L into X.
CALL PLOTXY
RET
```

Reference:

"VDP Discovered" , chapter 6 ,page 57

Code:

```
PLOTXY:     PUSH AF
            PUSH BC
            PUSH HL
            CALL CARTXYTOVRAM
```

```
LD C,A
LD A,(BITNUMBER)
LD HL,DOTVALUE
ADD A,L
LD L,A
LD A,(HL)
OR C
LD HL,(LSBPGTADDR)
CALL VDPWRTSEL
OUT (#01),A       ; #08 =Einstein  #98 =MSX
LD BC,8192
ADD HL,BC
CALL VDPWRTSEL
LD A,(COL)
OUT (#01),A       ; #08 =Einstein  #98 =MSX
POP HL
POP BC
POP AF
RET
```

G02. Title : POINTXY (POINT x,y)

Lines = 18 Bytes = Stacksize = 6+2 Datasize = 14

Language = Z80A Hardware = VDP Error Checking = No

System = MSX , Memotech (MTX) and Einstein

Description:

This subroutine requires that the x and y coordinates have previously been loaded into the appropriate data variables. Also that the x coordinate is in the range 0-255 and the y coordinate in the range 0-191. The VDP must be set to G2 mode (high resolution).

This subroutine,looks at the x,y position on the screen and tests to see if the pixel is on (1) or off (0) and returns the appropriate number in register A or in the data variable POINTSTATUS. The screen is unchanged.

Data Variables:

```
X: DS 1    Y: DS 1    LSBPGTADDR: DS 1   MSBPGTADDR: DS 1
BITNUMBER: DS 1   POINTSTATUS: DS 1
DOTVALUE: DB 128,64,32,16,8,4,2,1
```

Service Subroutines:

CARTXYTOVRAM ; VDPREADSEL

Registers Altered:

AF

How to Use:

```
LD HL,#4041             ; H= #40 (64) and L= #41 (65)
LD (X),HL               ; x = L  and  y = H
CALL POINTXY            ; result in reg A or in POINTSTATUS.
CP 1                    ; IS pixel on
JR Z,PIXELON            ; if on goto pixelon code,else pixeloff
RET
```

Reference:

"VDP Discovered" , chapter 6 ,page 59

Code:

```
POINTXY:    PUSH BC
            PUSH HL
            CALL CARTXYTOVRAM
            LD C,A
            LD A,(BITNUMBER)
            LD HL,DOTVALUE
            ADD A,L
            LD L,A
            LD A,(HL)
            AND C
            JR Z,POINTON
POINTOFF:   XOR A
            JR STOREPOINT
POINTON:    LD A,1
STOREPOINT: LD (POINTSTATUS),A
            POP HL
            POP BC
            RET
```

G03. Title : TOGGLEXY   (TOGGLE x,y)

Lines = 18   Bytes =   Stacksize = 10+2   Datasize = 14

Language = Z80A   Hardware = VDP   Error Checking = No

System = MSX , Memotech (MTX) and Einstein

Description:

This subroutine requires that the x and y coordinates have previously been loaded into the appropriate data variables. Also that the x coordinate is in the range 0-255 and the y coordinate in the range 0-191. The VDP must be set to G2 mode (high resolution).

This subroutine, like POINTXY , looks at the status of the pixel located at x,y. If the pixel is on ,then it is switched off and if the pixel was off ,then it was switched on. Unlike POINTXY , this subroutine carries out an on screen action. Therefore the screen is changed by this process. This is equivalent to the XOR functions on most CAD programs.

Data Variables:

```
X: DS 1    Y: DS 1    LSBPGTADDR: DS 1   MSBPGTADDR: DS 1
COL: DS 1   BITNUMBER: DS 1
DOTVALUE: DB 128,64,32,16,8,4,2,1
```

Service Subroutines:

```
CARTXYTOVRAM   ;  VDPWRTSEL  ; VDPREADSEL
```

Registers Altered:

None

How to Use:

```
LD HL,#0201                    ; H= #02  and   L= #01
LD (X),HL                      ; x = L   and   y = H
CALL TOGGLEXY
RET
```

Reference:

"VDP Discovered" ,chapter ,page 60

Code:

```
TOGGLEXY:   PUSH AF
            PUSH BC
            PUSH HL
            CALL CARTXYTOVRAM
            LD C,A
            LD A,(BITNUMBER)
            LD HL,DOTVALUE
            ADD A,L
            LD L,A
            LD A,(HL)
            XOR C
            LD HL,(LSBPGTADDR)
            CALL VDPWRTSEL
            OUT (#01),A       ; #08 =Einstein  #98 =MSX
            POP HL
            POP BC
            POP AF
            RET
```

G04. Title : CARTXYTOVRAM (x,y to VRAM address)

Lines = 39 Bytes =   Stacksize = 6+2   Datasize = 13

Language = Z80A   Hardware = VDP   Error Checking = No

System = MSX , Memotech (MTX) and Einstein

Description:

This subroutine requires that the x and y coordinates have previously been loaded into the appropriate data variables. Also that the x coordinate is in the range 0-255 and the y coordinate in the range 0-191. The VDP must be set to G2 mode (high resolution) and the PGT must be located at #0000.

The x,y coordinate values which we are so familar with, have to be converted into VRAM addresses that the VDP can handle and understand. The PGT in mode G2 or graphics screen is 6144 bytes long. As the last sentences suggests; the VDP handles positions on the screen as bytes, whereas pixels are more like bit size. Therefore the VRAM address is only accurate to 8 bits or one byte. This VRAM address is stored in the data variables: LSB/MSB PGTADDR. The subroutine also calculates which bit in the byte the x,y coordinates actually refers too.This is stored in BITNUMBER.

Data Variables:

```
X: DS 1    Y: DS 1    LSBPGTADDR: DS 1   MSBPGTADDR: DS 1
BITNUMBER: DS 1   DOTVALUE: DB 128,64,32,16,8,4,2,1
```

Service Subroutines:

VDPREADSEL

Registers Altered:

AF

How to Use:

This subroutine is an integral part of the PLOTXY subroutine. Therefore refer to G01,for use.

Reference:

"VDP Discovered" ,chapter 6 ,pages 56-57

Code:

```
CARTXYTOVRAM:PUSH BC
             PUSH HL
             XOR A
             LD A,(X)
             SRL A
             SRL A
             SRL A
             AND A             ; INT (X/8)
             SLA A
             SLA A
             SLA A             ; INT(X/8)*8
             LD C,A
             XOR A
```

```
            LD A,(Y)
            SRL A
            SRL A
            SRL A
            AND A             ; INT(Y/8)
            LD B,A
            LD A,23
            SUB B             ; 23-INT(Y/8)
            LD H,A
            LD (MSBPGTADDR),A
            LD A,(Y)
            AND 7
            LD B,A
            LD A,7
            SUB B
            ADD A,C
            LD L,A
            LD (LSBPGTADDR),A
GETBITNUM:  LD A,(X)
            AND 7
            LD (BITNUMBER),A
            CALL VDPREADSEL
            IN A,(#01)        ; #08 =Einstein  #98 =MSX
            POP HL
            POP BC
            RET
```

G05. Title : INVCOLBYTE (INVERSE COLOUR at x,y)

Lines = 20 Bytes = Stacksize = 12+2 Datasize = 7

Language = Z80A Hardware = VDP Error Checking = No

System = MSX , Memotech (MTX) and Einstein

Description:

This subroutine requires that the x and y coordinates have previously been loaded into the appropriate data variables. Also that the x coordinate is in the range 0-255 and the y coordinate in the range 0-191. The VDP must be set to G2 mode (high resolution),with the PGT at #0000 and the colour table at #2000 (8192).

The VDP contains the colour information in one byte per VRAM address. As this suggest,we only get 2 colours per 8 bits. The colour byte has a MSN (upper 4 bits)= INK colour between 0 and 15 and a LSN (lower 4 bits) = PAPER colour again between 0-15. The subroutine also requires that both the INK and PAPER colours had already been converted into the colour byte form , see G11.

The subroutine calculates the PGT VRAM address closest to the x,y coordinate and adds 8192 to get the colour table equivalent. The colour byte at this location is inverted , ie new INK = old PAPER and new PAPER = old INK. The screen is

changed.

Data Variables:

```
X: DS 1    Y: DS 1    LSBPGTADDR: DS 1   MSBPGTADDR: DS 1
COL: DS 1  INKCOL: DS 1  PAPERCOL: DS 1
```

Service Subroutines:

```
CARTXYTOVRAM   ;  VDPREADSEL   ;  VDPWRTSEL    ;    GETCOL
GRPPAPER   ;    GRPINK
```

Registers Altered:

None

How to Use:

```
LD A,#0F                ; set to white ink
CALL GRPINK
LD A,#01                ; set to black paper
CALL GRPPAPER           ; colour byte setup.
LD HL,#8101             ; H = #81 (129)  L = #01 (1)
LD (X),HL               ; x = L    and    y = H
CALL INVCOLBYTE
RET
```

Reference:

"VDP Discovered" ,chapter 6 ,page 63

Code:

```
INVCOLBYTE: PUSH HL
            PUSH DE
            PUSH AF
            LD HL,(X)
            CALL CARTXYTOVRAM
            LD HL,(LSBPGTADDR)
            LD DE,8192
            ADD HL,DE
            CALL VDPREADSEL
            IN A,(#01)              ; #08 =Einstein  #98 =MSX
INVERT:     RLCA
            RLCA
            RLCA
            RLCA
            CALL VDPWRTSEL
            OUT (#01),A             ; #08 =Einstein  #98 =MSX
            POP AF
            POP DE
            POP HL
            RET
```

G06. Title : GRPSCRCLS (CLS)

Lines = 14 Bytes = Stacksize = 6+2 Datasize = 4

Language = Z80A Hardware = VDP Error Checking = No

System = MSX , Memotech (MTX) and Einstein

Description:

Requires that the VDP is in G2 mode and that the PGT is located at #0000. The PGT holds the graphic information what we see on the screen. By clearing or setting every one of the 6144 bytes in the PGT to ZERO ,we have effectively cleared the PGT or the screen. Note that this subroutine doesn't clear the clour attributes of the screen. This must be done be setting the colour table to the default INK and PAPER colours. This can be done by loading PGTBASE with #2000 instead of #0000 and changing XOR A to LD A,(COL). Note that the PGT and Colour tables are both 6144 bytes long.

Data Variables:

```
PGTBASE: DS 2    PGTLEN: DS 2
```

Service Subroutines:

```
VDPWRTSEL    ;   VDPREADSEL
```

Registers Altered:

None

How to Use:

```
LD HL,#0000             ; the setup of PGTBASE and PGTLEN would
LD (PGTBASE),HL         ; carried at the program initialisation
LD HL,6144 ;
LD (PGTLEN),HL
CALL GRPSCRCLS
RET
```

Reference:

"VDP Discovered" ,chapter 6 ,page 50

Code:

```
GRPSCRCLS:  PUSH AF
            PUSH HL
            LD HL,(PGTBASE)
            CALL VDPWRTSEL
            LD HL,(PGTLEN)
GRPSCRCLS1: XOR A
            OUT (#01),A       ; #08 =Einstein  #98 =MSX
            DEC HL
            LD A,H
            OR L
            JR NZ,GRPSCRCLS1
```

```
            POP HL
            POP AF
            RET
```

G07   Title : DRAWBOX   (DRAWBOX x1,y1,x2,y2)

Lines = 29 Bytes =   Stacksize = 20+2   Datasize = 27

Language = Z80A   Hardware = VDP   Error Checking = No

System = MSX , Memotech (MTX) and Einstein

Description:

This subroutine requires that the x1,y1 and x2,y2 coordinates have previously been loaded into the appropriate data variables. Also that the x1 and x2 coordinates are in the range 0-255 and the y1 and y2 coordinates are in the range 0-191. The VDP must be set to G2 mode (high resolution).

The x1,y1 and x2,y2 coordinates can be anywhere on the screen but must be in opposite diagonal corners, so that the box can be drawn. The box is drawn in the INK and PAPER colours defined in the colour byte: COL. The screen is changed.

Data Variables:

```
X: DS 1    Y: DS 1     LSBPGTADDR: DS 1   MSBPGTADDR: DS 1
X1: DS 1   Y1: DS 1    COL: DS 1 BITNUMBER: DS 1
X2: DS 1   Y2: DS 1    DOTVALUE: DB 128,64,32,16,8,4,2,1
DX: DS 1       SGNX: DS 1      INCX: DS 1      XERR: DS 1
DY: DS 1       SGNY: DS 1      INCY: DS 1      YERR: DS 1
DISTANCE:   DS 1
```

Service Subroutines:

CARTXYTOVRAM ; VDPWRTSEL ; VDPREADSEL ; DRAWLINE ; PLOTXY

Registers Altered:

None

How to Use:

```
LD HL,#4080             ; H= #40 (64)   L= #80  (128)
LD (X1),HL              ; x1 = L   y1 = H
LD HL,#8080             ; H= #80 (128)  L= #80  (128)
LD (X2),HL              ; x2 = L   y2 = H
CALL DRAWBOX
RET
```

Reference:

None

Code:

```
DRAWBOX:    PUSH HL
            PUSH DE
            PUSH AF
            LD HL,(X1)
            LD DE,(X2)
            PUSH HL
            PUSH DE
            LD D,H
            LD (X2),DE
            CALL DRAWLINE     ; x1,y1 to x2,y1
            EX DE,HL
            LD (X1),HL
            POP DE
            LD (X2),DE
            CALL DRAWLINE     ; x2,y1 to x2,y2
            POP HL
            PUSH HL
            LD H,D
            LD (X1),HL
            CALL DRAWLINE     ; x1,y2 to x2,y2
            POP DE
            EX DE,HL
            LD (X2),DE
            LD (X1),HL
            CALL DRAWLINE     ; x1,y1 to x1,y2
            POP AF
            POP DE
            POP HL
            RET
```

G08. Title : FILLBOX (FILLBOX x1,y1,x2,y2)

Lines = 37 Bytes = Stacksize = 16+2 Datasize = 18

Language = Z80A Hardware = VDP Error Checking = No

System = MSX , Memotech (MTX) and Einstein

Description:

This subroutine requires that the x1,y1 and x2,y2 coordinates have previously been loaded into the appropriate data variables. Also that the x1 and x2 coordinates are in the range 0-255 and the y1 and y2 coordinates are in the range 0-191. The VDP must be set to G2 mode (high resolution).

The coordinates entered are those for the box already drawn. The x1,y1 and x2,y2 coordinates can be anywhere on the screen but must be in opposite diagonal corners, so that the box can be drawn. The box is drawn in the INK and PAPER colours defined in the colour byte: COL. The area of the box is also coloured likewise,although the filled area can be of a different colour because,the Fillbox subroutine doesn't redraw the box perimeter.

Note that during any fill area subroutines,if BASIC/OS is running in the background,then it will be interrupting the execution of our code for things like checking if BRK has been pressed,or the screen needs updating, etc. During,this screen glitches can occur. This is whereby,one or two pixels are not coloured during the fill subroutine. Enclosing, FILLBOX with DI/CALL FILLBOX/EI will disable the interrupts to give a better picture.

Data Variables:

```
X: DS 1    Y: DS 1    LSBPGTADDR: DS 1   MSBPGTADDR: DS 1
X1:DS 1    Y1:DS 1    COL: DS 1 BITNUMBER: DS 1
X2:DS 1    Y2:DS 1    DOTVALUE: DB 128,64,32,16,8,4,2,1
DX: DS 1       SGNX: DS 1      INCX: DS 1      XERR: DS 1
DY: DS 1       SGNY: DS 1      INCY: DS 1      YERR: DS 1
DISTANCE:   DS 1
```

Service Subroutines:

DRAWLINE ; PLOTXY ; CARTXYTOVRAM ; VDPWRTSEL ; VDPREADSEL

Registers Altered:

AF

How to Use:

```
LD HL,#4080             ; H= #40 (64)  L= #80 (128)
LD (X1),HL              ; x1 = L    y1 = H
LD HL,#8080             ; H= #80 (128)  L= #80 (128)
LD (X2),HL              ; x2 = L    y2 = H
DI                      ; avoid glitches
CALL FILLBOX
EI
RET
```

Reference:

None

Code:

```
FILLBOX:    PUSH HL
            PUSH DE
            LD HL,(X1)        ; ASSUMES BOX IS DRAWN FIRST.
            LD DE,(X2)
            PUSH BC
            PUSH DE
            PUSH HL
            INC H
            DEC L
            DEC D
            INC E
            LD A,H
            CP D
```

```
            JR C,DGTH
            PUSH AF
            SUB D
YCOUNT:     LD B,A
            POP AF
FBOXLOOP:   LD (Y1),A
            LD (Y2),A
            PUSH AF
            CALL DRAWLINE
            POP AF
            DEC A
            DJNZ FBOXLOOP
            POP HL
            POP DE
            LD (X1),HL
            LD (X2),DE
            POP BC
            POP DE
            POP HL
            RET

DGTH:       LD A,D
            PUSH AF
            SUB H
            JR YCOUNT
```

G09. Title : DRAWLINE (DRAWLINE x1,y1,x2,y2)

Lines = 137 Bytes = Stacksize = 22+2 Datasize = 27

Language = Z80A Hardware = VDP Error Checking = No

System = MSX , Memotech (MTX) and Einstein

Description:

This subroutine requires that the x1,y1 and x2,y2 coordinates have previously been loaded into the appropriate data variables. Also that the x1 and x2 coordinates are in the range 0-255 and the y1 and y2 coordinates are in the range 0-191. The VDP must be set to G2 mode (high resolution).

This subroutine uses the Bresenham algorithm for drawing lines. The advantages of this method, are that it avoids divisions and floating point arithmetic. Therefore,it is quick and efficient on the Z80. Basically, what this method does is to work out where every point on the line is on the screen and the subroutine PLOTXY , plots it. The screen is changed.

Data Variables:

```
X: DS 1    Y: DS 1    LSBPGTADDR: DS 1   MSBPGTADDR: DS 1
X1:DS 1    Y1:DS 1    COL: DS 1 BITNUMBER: DS 1
X2:DS 1    Y2:DS 1    DOTVALUE: DB 128,64,32,16,8,4,2,1
DX: DS 1       SGNX: DS 1      INCX: DS 1      XERR: DS 1
```

```
DY: DS 1       SGNY: DS 1      INCY: DS 1      YERR: DS 1
DISTANCE:   DS 1
```

Service Subroutines:

```
CARTXYTOVRAM ; PLOTXY ; VDPWRTSEL ; VDPREADSEL
```

Registers Altered:

```
AF
```

How to Use:

```
LD HL,#3040             ; H= #30 (48)   L= #40 (64)
LD (X1),HL              ; x1 = L   y1 = H
LD HL,#A020             ; H= #A0 (160)  L= #20 (32)
LD (X2),HL              ; x2 = L   y2 = H
CALL DRAWLINE
RET
```

Reference:

"Draw the Line" by A.Redfern ,PCW March 1989 ,pages 216-218
"C:The Complete Reference" by H.Schildt ,ch 24, pages 650-651

Code:

```
DRAWLINE:   PUSH HL
            PUSH DE
            PUSH BC
            CALL INITVARS
            CALL SETINCX
            CALL SETINCY
            CALL GETDIST
            CALL DLINE
            POP BC
            POP DE
            POP HL
            RET

INITVARS:   XOR A
            LD (YERR),A
            LD (XERR),A
            LD (SGNX),A
            LD (SGNY),A
            LD HL,(X1)
            LD DE,(X2)
            LD A,E
            CP L
            JR NC,SXFLAG
            LD A,1
            LD (SGNX),A
            LD A,L
            LD L,E
SXFLAG:     SUB L
            LD (DX),A
            LD A,D
```

```
            CP H
            JR NC,SYFLAG
            LD A,1
            LD (SGNY),A
            LD A,H
            LD H,D
SYFLAG:     SUB H
            LD (DY),A
            RET

NEGX:       XOR A
            INC A
            JR SXFLAG

NEGY:       XOR A
            INC A
            JR SYFLAG

SETINCX:    LD BC,(DX)
            LD A,1
            CP B
            JR Z,SDX
            DEC A
            CP C
            JR Z,SDX
            INC A
SDX:        LD (INCX),A
            RET

SETINCY:    LD BC,(DY)
            LD A,1
            CP B
            JR Z,SDY
            DEC A
            CP C
            JR Z,SDY
            INC A
SDY:        LD (INCY),A
            RET

GETDIST:    LD A,(DY)
            LD B,A
            LD A,(DX)
            CP B
            JR C,DYGTDX
DXGTDY:     LD (DISTANCE),A
            RET
DYGTDX:     LD A,B
            LD (DISTANCE),A
            RET

DLINE:      LD HL,(X1)
            LD (X),HL
            XOR A
            LD B,A
DRAWLOOP:   LD A,(DISTANCE)
            INC A
```

```
            CP B
            RET C
            CALL PLOTXY
            LD A,(DY)
            LD C,A
            LD A,(YERR)
            ADD A,C
            LD (YERR),A
            PUSH AF
            LD A,(DX)
            LD C,A
            LD A,(XERR)
            ADD A,C
            LD (XERR),A
            PUSH AF
XERRGTDIST: LD A,(DISTANCE)
            LD C,A
            POP AF
            PUSH BC
            CP C
            JR C,YERRGTDIST
            JR Z,YERRGTDIST
            SUB C
            LD (XERR),A
            LD A,(INCX)
            LD C,A
            LD A,(SGNX)
            CP 0
            JR Z,ADDINCX
SUBINCX:    LD A,L
            SUB C
RETINCX:    LD L,A
YERRGTDIST: POP BC
            POP AF
            CP C
            JR C,UPDXY
            JR Z,UPDXY
            SUB C
            LD (YERR),A
            LD A,(INCY)
            LD C,A
            LD A,(SGNY)
            CP 0
            JR Z,ADDINCY
SUBINCY:    LD A,H
            SUB C
RETINCY:    LD H,A
UPDXY:      LD (X),HL
            INC B
            JR DRAWLOOP
            RET

ADDINCX:    LD A,L
            ADD A,C
            JR RETINCX

ADDINCY:    LD A,H
```

```
            ADD A,C
            JR RETINCY
```

G10. Title: TRIANGLE (TRIANGLE x1,y1,x2,y2,x3,y3)

Lines = 17 Bytes = Stacksize = 12+2 Datasize = 29

Language = Z80A Hardware = VDP Error Checking = No

System = MSX , Memotech (MTX) and Einstein

Description:

This subroutine requires that the x1,y1 and x2,y2 and x3,y3 coordinates have previously been loaded into the appropriate data variables. Also that the x1,x2 and x3 coordinates are in the range 0-255 and the y1,y2 and y3 coordinates are in the range 0-191.The VDP must be set to G2 mode (high resolution).

This subroutine draws lines to the three corners of a triangle in the INK and PAPER colours held in the colour byte. The screen is changed.

Data Variables:

```
X: DS 1    Y: DS 1    LSBPGTADDR: DS 1   MSBPGTADDR: DS 1
X1:DS 1    Y1:DS 1    COL: DS 1 BITNUMBER: DS 1
X2:DS 1    Y2:DS 1    DOTVALUE: DB 128,64,32,16,8,4,2,1
X3:DS 1    Y3:DS 1
DX: DS 1       SGNX: DS 1      INCX: DS 1      XERR: DS 1
DY: DS 1       SGNY: DS 1      INCY: DS 1      YERR: DS 1
DISTANCE:   DS 1
```

Service Subroutines:

PLOTXY ; CARTXYTOVRAM ; DRAWLINE ; VDPWRTSEL ; VDPREADSEL

Registers Altered:

AF

How to Use:

```
LD HL,#4040             ; H= #40 (64)   L= #40 (64)
LD (X1),HL              ; x1 = L     y1 = H
LD HL,#8080             ; H= #80 (128)  L= #80 (128)
LD (X2),HL              ; x2 = L     y2 = H
LD HL,#2020             ; H= #20 (32)   L= #20 (32)
LD (X3),HL              ; x3 = L     y3 = H
CALL TRIANGLE
RET
```

Reference:

None

Code:

```
TRIANGLE:   PUSH HL
            LD HL,(X1)
            PUSH HL
            LD HL,(X2)
            PUSH HL
            CALL DRAWLINE
            LD HL,(X3)
            LD (X2),HL
            CALL DRAWLINE
            POP HL
            LD (X1),HL
            CALL DRAWLINE
            LD (X2),HL
            POP HL
            LD (X1),HL
            POP HL
            RET
```

G11. Title : GRPINK / GRPPAPER / GETCOL (INK i / PAPER p)

Lines = 22 Bytes = Stacksize = 8+2 Datasize = 3

Language = Z80A Hardware = VDP Error Checking = No

System = MSX , Memotech (MTX) and Einstein

Description:

The VDP needs to be in G2 mode. INK and PAPER colours are limited to the 16 (0-15) provided by the VDP. See G05,for a description of the colour byte.

Register A holds the INK colour prior to calling GRPINK. This subroutine will update the colour byte. Register A holds the PAPER colour prior to calling GRPPAPER. This subroutine will update the colour byte.

GETCOL subroutine is used by both GRPINK and GRPPAPER to get the colour byte in the correct format.

Data Variables:

```
INKCOL: DS 1          PAPERCOL: DS 1          COL: DS 1
```

Service Subroutines:

None

Registers Altered:

AF

How to Use:

```
LD A,#0E                    ; grey INK
CALL GRPINK
LD A,#01                    ; black PAPER
CALL GRPPAPER
RET
```

Reference:

"VDP Discovered" ,chapter 3 ,pages 23-25

Code:

```
GRPINK:LD   (INKCOL),A
            CALL GETCOL
            RET

GRPPAPER:   LD (PAPERCOL),A
            CALL GETCOL
            RET

GETCOL:     PUSH BC
            PUSH AF
            LD A,(INKCOL)
            AND #0F
            SLA A
            SLA A
            SLA A
            SLA A
            LD B,A
            LD A,(PAPERCOL)
            AND #0F
            ADD A,B
            LD (COL),A
            POP AF
            POP BC
            RET
```

G12. Title : BORDER (BORDER b)

Lines = 7 Bytes = Stacksize = 0+2 Datasize = 1

Language = Z80A Hardware = VDP Error Checking = No

System = MSX , Memotech (MTX) and Einstein

Description:

Can be in any mode. The border colour (0-15) is passed to the BORDER subroutine via register A. In TEXT mode, the BORDERCOL is the same as the PAPER colour. In G1,G2 and MC modes, the border colour is independent of the INK and PAPER colours. The screen is changed.

Data Variables:

```
BORDERCOL: DS 1
```

Service Subroutines:

None

Registers Altered:

AF   and the VDP register number 7.

How to Use:

```
LD A,#0F                    ; set border colour to White
CALL BORDER
RET
```

Reference:

"VDP Discovered" ,chapter 2 ,page 16 and chapter 3,page 25.

Code:

```
BORDER:     AND #0F
            LD (BORDERCOL),A
            OUT (#02),A       ; #09 =Einstein  #99 =MSX
            LD A,7
            OR #C0
            OUT (#02),A       ; #09 =Einstein  #99 =MSX
            RET
```