Dot Graphics on the IMSAI-VIO

by Gary Sabot

Run TRS-80, Apple or PET graphics programs on your IMSAI-VIO and similar video boards with this program.

Several of the personal computers in wide use today are capable of displaying low resolution graphics. This means that they are able to display dots, lines, pictures, or even animated characters on their screens. The TRS-80 and the Apple have this capability. Because of the widespread use of these two computers, there are many programs available which make use of their graphics capability. This article presents a program that will enable owners of the Imsai VIO (or similar memory-mapped displays, such as the Polymorphics VTI) to utilize these programs. By making a few simple changes, *all* TRS-80 graphics programs, most Apple graphics programs (those programs which do not make extensive use of color), and some PET graphics programs will run on your machine.

The Imasi VIO is capable of displaying special "graphics characters." (See Figure 1.) Each graphics character contains six squares. By using the proper graphics character, it is possible to turn each of these six squares on (white), or off (black) independently. The problem is: how can a single square be turned on, without disturbing the five squares that surround it?

My solution to this problem is in the form of a machine language program. (See listing #1.) It allows a Basic program to quickly and easily plot points using the VIO. It can be modified to work with other memory-mapped display boards, such as the Polymorphics VTI. The program is designed to be used in conjunction with Microsoft Basic and CP/M. (Of course, it can also be used by a machine language program.)

To plot a point, the proper graphics character must be selected; then this character must be placed in the correct memory location. If this process were to be implemented as a Basic program, it would take approximately one-half second to plot each point. If a program that plots several hundred points were run, however, those one-half seconds would add up, delaying the program. I have implemented the plotting program in machine language, because a machine language program is considerably faster than an equivalent program written in Basic. If a Basic program needs to turn a certain square "on," it simply passes its X-Y coordinates to this routine (see Figure 2) and calls it, using the USR function. The routine then turns the square "on," and subsequently returns to the Basic program. Analagous procedures may be used to determine the square's present color (black or white), or to turn it off (black).

Gary Sabot, 38 Woodland Rd., Roslyn, NY 11576.

Utilizing The Program

If you have a 30K CP/M system using the Imsai VIO, you can employ the program just as I assembled it. To use the routine (after you have POKE'd it into memory—see the Basic listing), first POKE the Y coordinate of the desired pixel into location 6889H, then POKE the X coordinate into 688AH, and POKE the function number into 688BH. The function number would be a 1 to set the pixel white. This is equivalent to the TRS-80's SET(X,Y) command. The function number would be a 0 to set the pixel black. This is identical to the TRS-80's RESET(X,Y) command. If the function number is a two, the plotting routine will determine the present status of the pixel, without disturbing it. This is similar to the TRS-80's POINT(X,Y) command. The status of the pixel is retrieved by a PEEK to 688BH. A 0 will be found there if

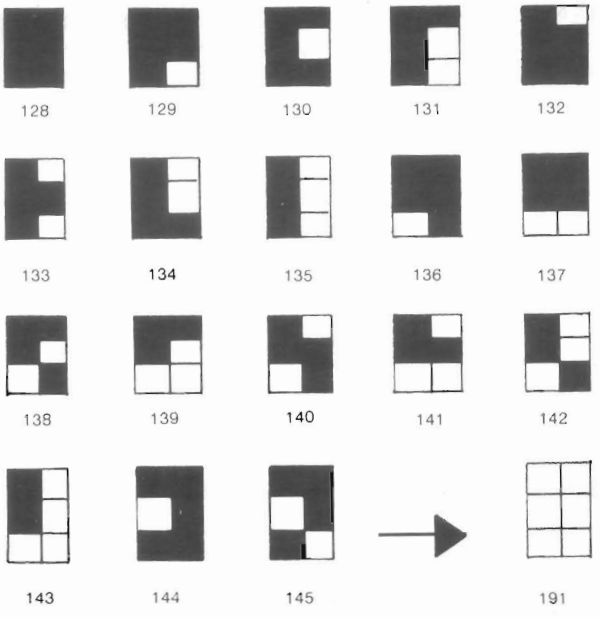

Figure 1. Some of the Imsai VIO's "graphic characters." The number of the desired character can be placed in the VIO's refresh memory; subsequently, the character will appear on the screen.

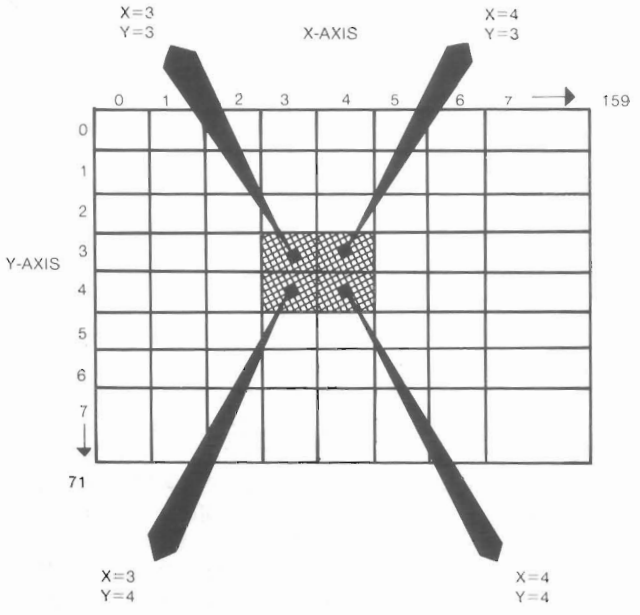

Figure 2. This is the coordinate system used to address the squares, or pixels, on the screen. It is the same as the system used by the TRS-80, except that the VIO's display is larger (160 by 72, compared with the TRS-80's 128 by 48).

the pixel is black and a 1 will be found there if the pixel is white. Once the proper values for the X,Y coordinates, and the function number have been POKED into memory, jump to 6800H using the USR function to execute the plotting routine.

I have provided a sample Basic program which uses my point plotting routine. (See listing #2.) It is a modification of program on page 33 of *Introduction to Low Resolution Graphics*, by Nat Wadsworth. The program draws lines between random points on the screen.

This plotting routine enables you to adapt to your computer the many TRS-80, Apple, and PET programs employing graphics which are in the public domain.

Because the plotting program is written in machine language, it is necessary to reserve memory for it when MBasic is run. To do this, instead of simply typing "MBasic" to run MBasic, type "MBasic /M:&H67FF". This sets 67FF as the highest address available for use by MBasic. The preface "&H" indicates that a hex number will follow.

If your system is larger than 30K, you might want to reassemble the routine at a higher address, in order to fully utilize your memory. For every kilobyte of memory that you have above 30K, add 400H to 67FFH. Then reassemble the routine at the resulting address. Finally, substitute the new origin for the 67FFH in "MBasic /M:&H67FF". Of course, the locations for POKING and PEEKING change each time a different version of this program is assembled.

If you are using a video board other than the VIO, you will probably have to reassemble the program, making one or more of the following changes (most involve the "EQUates" in the beginning of the program):

1) If your video board is addressed at a location other than 61440 (F000H), change the value in the line beginning "SCREENAD EQU..." in the listing to the correct screen address.

2) If the line length of your display is different than 80 characters per line, change the line length in the listing ("LINE EQU...") to the proper value, and reassemble it.

3) If a black pixel is represented by a 1, and not a 0, substitute 1 for 0 in the listing where it now reads "BLK EQU 0".

4) Determine the value of a blank (all black) character cell and substitute this for the 80H in the line "BLKCHR EQU 80H".

5) Find out what the proper CHRAND and CHRCPI values for your display should be (refer to the comments in the program listing) and insert the correct values into the program.

6) If the progression from black to white on your video board is different than that shown in Figure 1, you will have to modify the portion of the program (6843H thru 6863H) that calculates the bit mask. This can be a very involved process!

Now that your computer has dot graphics capability, make it work for you. Programs can be written with output in the form of a graph, instead of in numbers and letters. Programs can even be designed to imitate arcade games. One of the most exciting possibilities of this plotting routine is that it enables you to adapt to your computer the many TRS-80, Apple, and PET programs employing graphics which are in public domain. □

—PROGRAM BEGINS NEXT PAGE—

POINT PLOT LISTING

LISTING #1

```
                        ;******* POINT PLOT ROUTINE FOR MEMORY MAPPED DISPLAYS *******
                        ;******** BY GARY SABOT 11/27/79 ********
F000 =          SCREENAD        EQU     61440   ;ADDRESS OF VIDEO BOARD
0050 =          LINE            EQU     80      ;LINE LENGTH
0000 =          BLK             EQU     0       ;BIT THAT REPRESENTS A
                                                ;PIXEL THAT IS "OFF" (BLACK)
0080 =          BLKCHR          EQU     80H     ;CONTENTS OF A BLANK (BLACK)
00C0 =          CHRAND          EQU     0C0H
0080 =          CHRCPI          EQU     80H
                        ;CHRAND IS "ANDED" WITH THE CONTENTS OF A CHARACTER CELL.  IF
                        ;THE CELL CONTAINS A VALID GRAPHICS CHARACTER (NON-ALPHABETIC)
                        ;THE RESULT SHOULD EQUAL CHRCPI.  IF IT DOES NOT, A BLANK
                        ;CHARACTER WILL BE PLACED IN THE CELL.
6800                            ORG     6800H
                        ;
                        ;DATA SHOULD BE STORED IN FORM Y, X, FUNCTION #
                        ;FUNCTION #-- 0 MEANS SET BLACK, 1 MEANS SET WHITE,
                        ;ALL ELSE MEANS RETURN DOT STATUS (0=BLACK, 1=WHITE)
                        ;
6800 218968                     LXI     H,DATA          ;SET POINTER TO DATA
6803 0E00                       MVI     C,0             ;LOAD C WITH ZERO
6805 7E                         MOV     A,M             ;LOAD Y INTO A
6806 D603               DIV:    SUI     3               ;DIVIDE Y BY 3,
                                                        ;PUT RESULT IN C
6808 DA1068                     JC      DIVX
680B 0C                         INR     C
680C A7                         ANA     A
680D C20668                     JNZ     DIV
6810 328C68             DIVX:   STA     YREM            ;SAVE REMAINDER FROM
                                                        ;DIVISION OF Y
6813 23                         INX     H               ;READ IN VALUE OF X
6814 7E                         MOV     A,M
6815 0F                         RRC                     ;DIVIDE X BY 2
6816 E67F                       ANI     7FH             ;STRIP OFF FIRST BIT
6818 2100F0                     LXI     H,SCREENAD      ;LOAD ADDRESS OF VIDEO BOARD
681B 5F                         MOV     E,A             ;CALCULATE CHARACTER'S
                                                        ;LOCATION IN MEMORY
681C 1600                       MVI     D,0
681E 19                         DAD     D               ;ADD X TO BASE ADDRESS
681F 115000                     LXI     D,LINE          ;PREPARE TO MULTIPLY
6822 79                         MOV     A,C
6823 A7                         ANA     A
6824 CA3068                     JZ      ADFND
6827 3D                 MULT:   DCR     A               ;ADD Y*LINE LENGTH TO
                                                        ;BASE ADDRESS
6828 CA2F68                     JZ      ONEMOR          ;PUT ADDRESS IN HL
682B 19                         DAD     D
682C C32768                     JMP     MULT
682F 19                 ONEMOR: DAD     D
                        ;
                        ADFND:
                        ;FIRST CHECK IF A GRAPHICS CHARACTER IS ALREADY IN CHARACTER
                        ;CELL.  IF NOT, STORE A BLACK CHARACTER THERE.
                        ;THEN CALCULATE BIT MASK
6830 7E                         MOV     A,M             ;LOAD CHARACTER
6831 E6C0                       ANI     CHRAND          ;CHECK IF GRAPHICS CHARACTER
6833 FE80                       CPI     CHRCPI
6835 CA3B68                     JZ      FNDBIT          ;YES, NOW CONTINUE
6838 3E80                       MVI     A,BLKCHR        ;NO, STORE AN ALL
```

```
                                                    ;BLACK CHARACTER
683A 77                 MOV     M,A
683B 3A8A68     FNDBIT: LDA     DATA+1              ;LOAD VALUE OF X
683E E601               ANI     1                   ;GET REMAINER FROM
                                                    ;DIVISION BY 2
6840 CA4868             JZ      XRZER               ;REMAINDER IS ZERO
6843 3E01               MVI     A,1                 ;BEGIN TO FORM BIT MASK IN A
6845 C34A68             JMP     CONT                ;CONTINUE
6848 3E08       XRZER:  MVI     A,8
684A 47         CONT:   MOV     B,A                 ;SAVE PARTIALLY FORMED MASK
684B 3A8C68             LDA     YREM                ;LOAD REMAINER FROM
                                                    ;DIVISION OF Y
684E A7                 ANA     A                   ;SET FLAGS
684F CA6068             JZ      SHIFT2              ;NEEDS TO BE SHIFTED TWICE
6852 3C                 INR     A
6853 CA6468             JZ      MSKFND              ;DOES NOT NEED TO BE SHIFTED
6856 3C                 INR     A
6857 C26068             JNZ     SHIFT2              ;MUST BE SHIFTED TWICE
685A 78         SHIFT1: MOV     A,B                 ;LOAD BIT MASK
685B 07                 RLC                         ;SHIFT LEFT ONCE
685C 47                 MOV     B,A                 ;SAVE IN B
685D C36468             JMP     MSKFND
6860 78         SHIFT2: MOV     A,B                 ;LOAD BIT MASK
6861 0707               RLC ! RLC                   ;SHIFT LEFT TWICE
6863 47                 MOV     B,A                 ;SAVE IN B
                ;
                ;THE BIT MASK IS IN B AND THE CHARACTER ADDRESS IS IN HL.
                ;NOW COMPUTE AND PUT PROPER VALUE ON THE SCREEN.
                MSKFND:
6864 3A8B68             LDA     DATA+2              ;GET FUNCTION #
6867 A7                 ANA     A
6868 CA7E68             JZ      SETBLK              ;0 MEANS SET DOT BLACK
686B 3D                 DCR     A
686C CA8568             JZ      SETWHT              ;1 MEANS SET DOT WHITE
                ;
                ;ASSUME CALLER WANTS DOT STATUS
                ;
686F 7E                 MOV     A,M                 ;LOAD CHARACTER CELL
6870 A0                 ANA     B                   ;AND WITH BIT MASK
                        IF      BLK
                        JNZ     RETBLK
                        ENDIF
                        IF      NOT BLK
6871 CA7A68             JZ      RETBLK
                        ENDIF
6874 3E01               MVI     A,1                 ;RETURN A 1 FOR "WHITE"
6876 328B68             STA     DATA+2
6879 C9                 RET
687A 328B68     RETBLK: STA     DATA+2              ;RETURN A 0 FOR "BLACK"
687D C9                 RET


                ;THE FOLLOWING ROUTINES ALL LOAD THE CHARACTER CELL,
                ;MODIFY IT IN THE DESIRED WAY, SAVE IT, AND RETURN.
                ;
                ;SET DOT BLACK
                ;
                SETBLK:
                        IF      BLK                 ;CONDITIONAL ASSEMBLY
                                                    ;WHEN BLACK=1
                        MOV     A,M
                        ORA     B
                        MOV     M,A
                        RET
```

```
                        ENDIF
                ;
                        IF      NOT BLK
687E 78                 MOV     A,B
687F 2F                 CMA
6880 47                 MOV     B,A
6881 7E                 MOV     A,M
6882 A0                 ANA     B
6883 77                 MOV     M,A
6884 C9                 RET
                        ENDIF
                ;
                ;SET DOT WHITE
                ;
                SETWHT:
                        IF      BLK
                        MOV     A,B
                        CMA
                        MOV     B,A
                        MOV     A,M
                        ANA     B
                        MOV     M,A
                        RET
                        ENDIF
                ;
                        IF      NOT BLK
6885 7E                 MOV     A,M
6886 B0                 ORA     B
6887 77                 MOV     M,A
6888 C9                 RET
                        ENDIF

6889            DATA    DS      3               ;STORAGE FOR Y,X,
                                                ;AND FUNCTION NUMBER
688C            YREM    DS      1               ;STORAGE FOR REMAINER OF Y/3
688D                    END     6800H
```