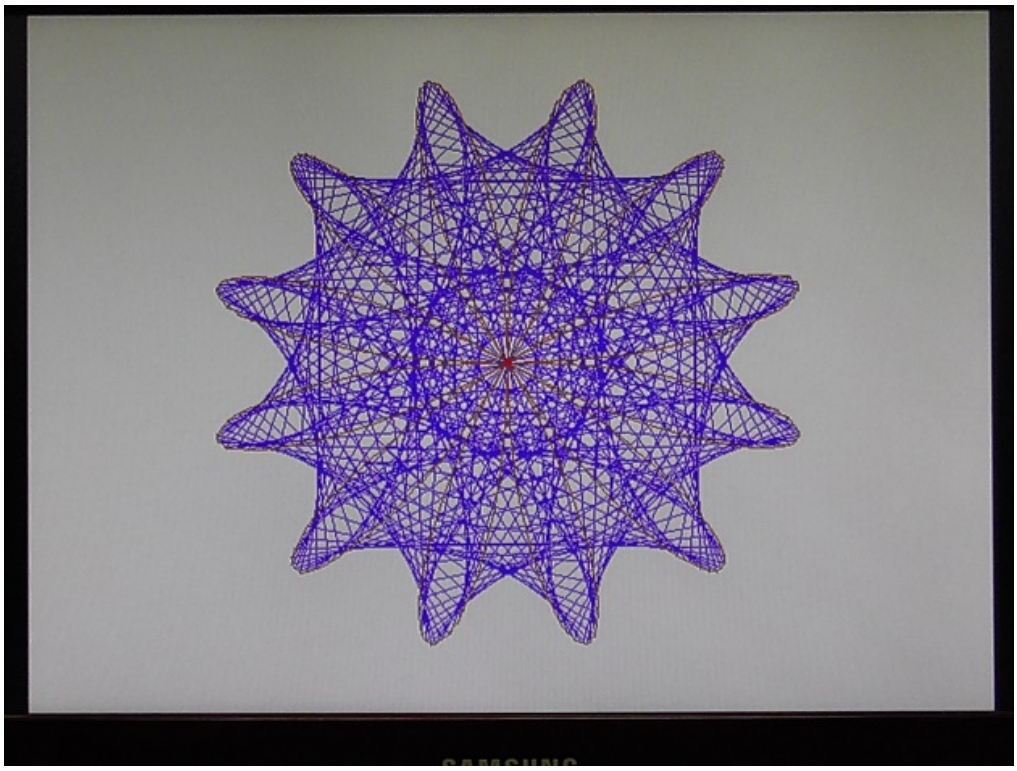
A video graphics card adaptor for the TRS-80 Model 1 microcomputer



Glen Kleinschmidt

http://www.glensstuff.com/

July 2021

Introduction

This project consists of two parts:

Hardware - a very simple interface in 74HC(T) logic designed to permit any of my three "VGA" video graphics cards (http://www.glensstuff.com/vgavideocards/vgavideocards.htm) to be fully addressed and controlled via the Expansion Interface port of a vintage TRS-80 Model 1 microcomputer. Add dual-monitor support and high-resolution graphics to your 1970s era hardware!

Software - fast machine-code driver and video graphics drawing routines callable by BASIC. Specifically programmed for 16 KB Level II machines and packaged as a single object file, the routines load into the top 4 KB of RAM with the SYSTEM command and provide BASIC with a host of powerful drawing functions. These functions include: setting pixels, drawing lines between specified points, drawing circles/arcs and rectangles (outlined or filled), bucket-pour filling, pixel reading for block copying (amongst other things) and a basic 80 column by 60 row text mode with 128 (completely redefinable) ASCII characters to choose from.

I have dubbed these my "Stage 1" routines and they were specifically written for the B variant of my three video card designs – that is the 64-colour card with a resolution of 640 by 480 pixels.

The complete machine-code Assembly listing is provided in this document, along with comprehensive instructions on how to program display graphics with the routines. Several demonstration BASIC program listings are also included, along with screen photos of the graphical displays that they produce.

The Stage 1 routines, along with all of the BASIC programs detailed herein, can be downloaded from the page dedicated to this project on my website:

http://www.glensstuff.com/trs80vga/trs80vga.htm

They are provided in the format of audio files to be played back for "cassette" loading. PCB Gerber files for all hardware described herein is also made available.

Some important considerations

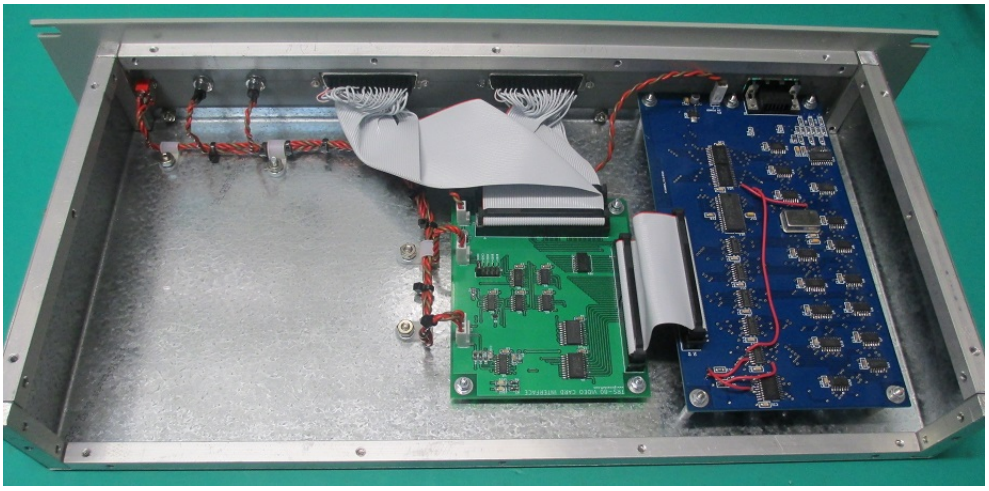
I constructed my unit into a slightly agricultural, home-fabricated, 19" rack case, specifically to have a form-factor to suit my TRS-80 Model 1 clone (http://www.glensstuff.com/trs80/trs80.htm). Note that the video card interface is designed with a through-port connector so that the Expansion Interface port still remains available with the VGA adaptor plugged in.

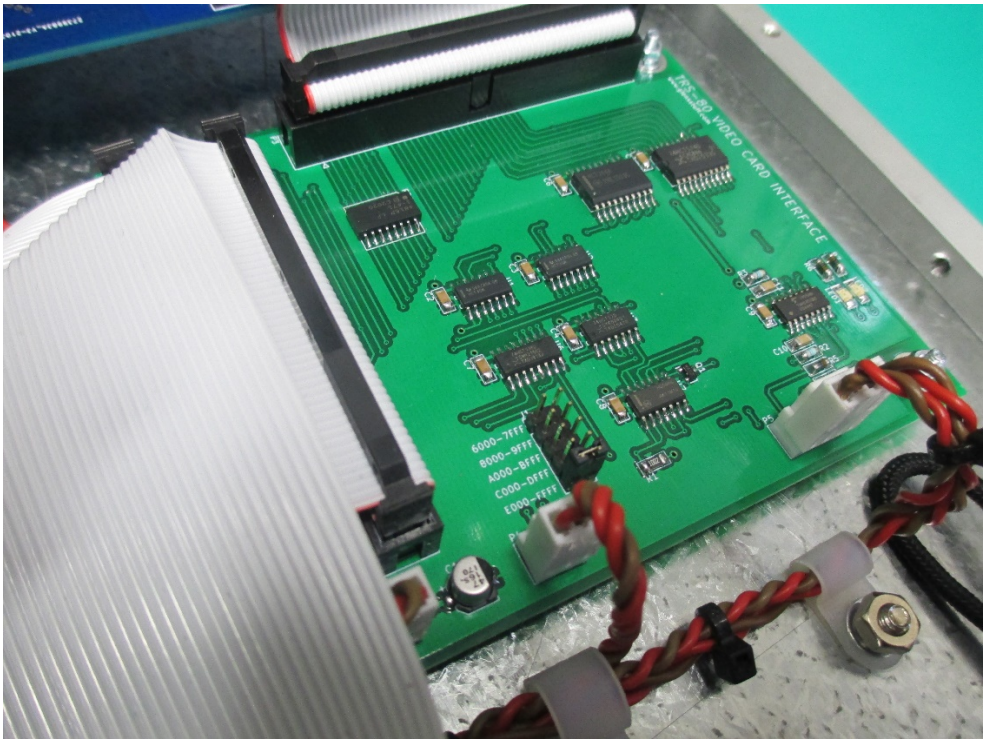
Anyone contemplating duplicating this design for connection to an original TRS-80 Model 1 computer will have to come up with their own practical packaging solution, utilising the through-port connector or not. I think a box that sits immediately behind the keyboard unit, that one of the monitors can sit atop would probably be the most practical solution, in most cases.

Also, as per my TRS-80 Model 1 clone design, the Expansion Interface port is broken out on the video card interface PCB via a 50-pin IDC header. In order to connect the unit to an original TRS-80 Model 1 computer, an adaptor cable would need to be soldered up as the Expansion Interface port on the original machine took the form of a 40-pin edge connector with a different pin-out. I'd recommend against making the interconnecting ribbon cable excessively long too. I also believe that, depending the original hardware's PCB revision number, +5V may or may not be available at the Expansion Interface port connector. An alternative source of power for the adaptor therefore might have to be found.

Keep in mind here that if embarking upon such an adventure with the original hardware, you're on your own. My hardware design is theoretically compatible, but I don't own an original machine and have never plugged anything into one.

Some pictures of the completed adaptor:





Installed into my retro computer rack and plugged into my TRS-80 Model 1 clone:

R.I.P. Fluffy Bum.

Hardware description

Interfacing to the TRS-80

A full hardware description of my "B"-variant video card design is not part of this document and technical details of the video card are only described, hereon, to the extent necessary to explain the operation of the TRS-80 interface circuit. A hyperlink to the webpage dedicated to the hardware details of the video card has already been provided in the introductory text.

VGA video card B has 2^19 bytes of memory, which equates to 512 kilobytes in vintage terms, where 1 KB = 1024 bytes. 307,200 (640 x 480) of these 524,288 bytes control pixels on the screen. The unused memory locations can be used, if desired, as system RAM for storing whatever the enterprising programmer might see fit. Indeed, one need not even bother to plug a display monitor into the video card at all and just utilise the hardware as one massive RAM expansion! Afterall, from the processor's perspective, the video card is nothing more than an external memory.

512 KB, however, is well beyond the limit that can be directly addressed by the Z80 microprocessor. The primary purpose of the interface circuit is to address this issue (pun intended). The interface circuit partly consists of a "page address register", which is used to control the most significant address lines of the video card such that the CPU now has access to the full 512 KB of video memory in manageable 8 KB pages. This means that the video card only occupies an 8-kilobyte chunk of the computer's main memory map.

The page address register is a single byte of memory that can be both written to and read, just like any other RAM location and is formed by combination of octal latch U4 and octal buffer/line-driver U5. The address location of the page address register is at the very top of the 8 KB chunk and its location is decoded, for the most part, by integrated circuits U1, U2 and U6. Note that this is a location of the video memory page that does not correspond to a pixel on the screen.

The video card hardware assigns one kilobyte of memory to each horizontal line of pixels on the video display, of which byte locations 1 through to 640 correspond to the pixels of the respective line; note the offset of one and that the count does not start at 0. This means that each 8 KB page directly programs 8 horizontal video lines. The vertical display resolution is 480 lines/pixels, so the full complement of video memory that maps to the screen is accessed through 480 / 8 = 60 individual pages, numbered 0 to 59.

The jumper header at the decoded outputs of U6, a 3-to-8-line decoder, permits the start address of the 8 KB chunk of address space that controls the video card to be set to any one of five sequential, 8 KB blocks at the top of the Z80's 16-bit address space.

Note that my Stage 1 machine-code routines require that this jumper be set to the pin 9-10 bridging position, so as to occupy the address range (in hexadecimal) E000 – FFFF. This is the top-most area of memory and the page address register, when jumpered as such, then resides at the highest location in memory – FFFF.

My TRS-80 Model 1 clone is equipped with the full 48 KB of system RAM allowable by the Level II operating system. This means that the otherwise unutilised top half (32 KB) of the computer's memory map is filled with on-board RAM and that the video card memory, as it occupies this space, is effectively addressed in parallel.

This leads to one potential issue that needs to be accounted for. Writing to two separate memories in parallel is always fine, but there is a potential for data bus contention when reading if the location being read doesn't have the same data value in both memory devices. Ordinarily, both memories will contain the exact same data, except initially upon the computer being powered-on, in which case the contents of both RAMs will be more or less random. If a read operation is erroneously performed, before at least one write operation to a

select location has been made, then the odds aren't in our favour that the two memories will be holding the same value and each will be trying to put a different data value onto the data bus during the read cycle.

Eight-resistor pack RP1 serves as a current-limiting isolator between the data bus of the video card and that of the computer. At 470 ohms, these resistors are low enough in value by a safe margin so as to not cause any signal integrity issues at the speed that the TRS-80 CPU operates at, but are high enough in value to adequately limit the I/O-pin currents such that nothing at all deleterious will happen in the possible case of contention.

There isn't much else to the video card interface. Dual monostable IC U8 delivers pulse-stretching duties to drive a pair of useful control-interface activity-indicating LEDs. LED2 on the PCB will flash whenever the page address register is written to. LED1 blinks when the video memory is being addressed. Header P5 allows external, front panel LEDs to be connected. They function in parallel to the ones on the PCB and installing the ones on the PCB is optional. In my build I labelled these front-panel LEDs "P.R." and "V.M." for "page register" and "video memory" respectively.

The final aspect of the video card interface to describe is the function of the front-panel "SYNC ON" switch. This SPDT switch connects to header P4 (common/pole to pin 2). When flicked to the "SYNC ON" position, this switch connects the vertical blanking output of the video card to the Z80's interrupt line.

This gives one the potential opportunity to synchronise the operations of the CPU to the display, such that write and read operations to the video memory can be timed to the vertical blanking interval.

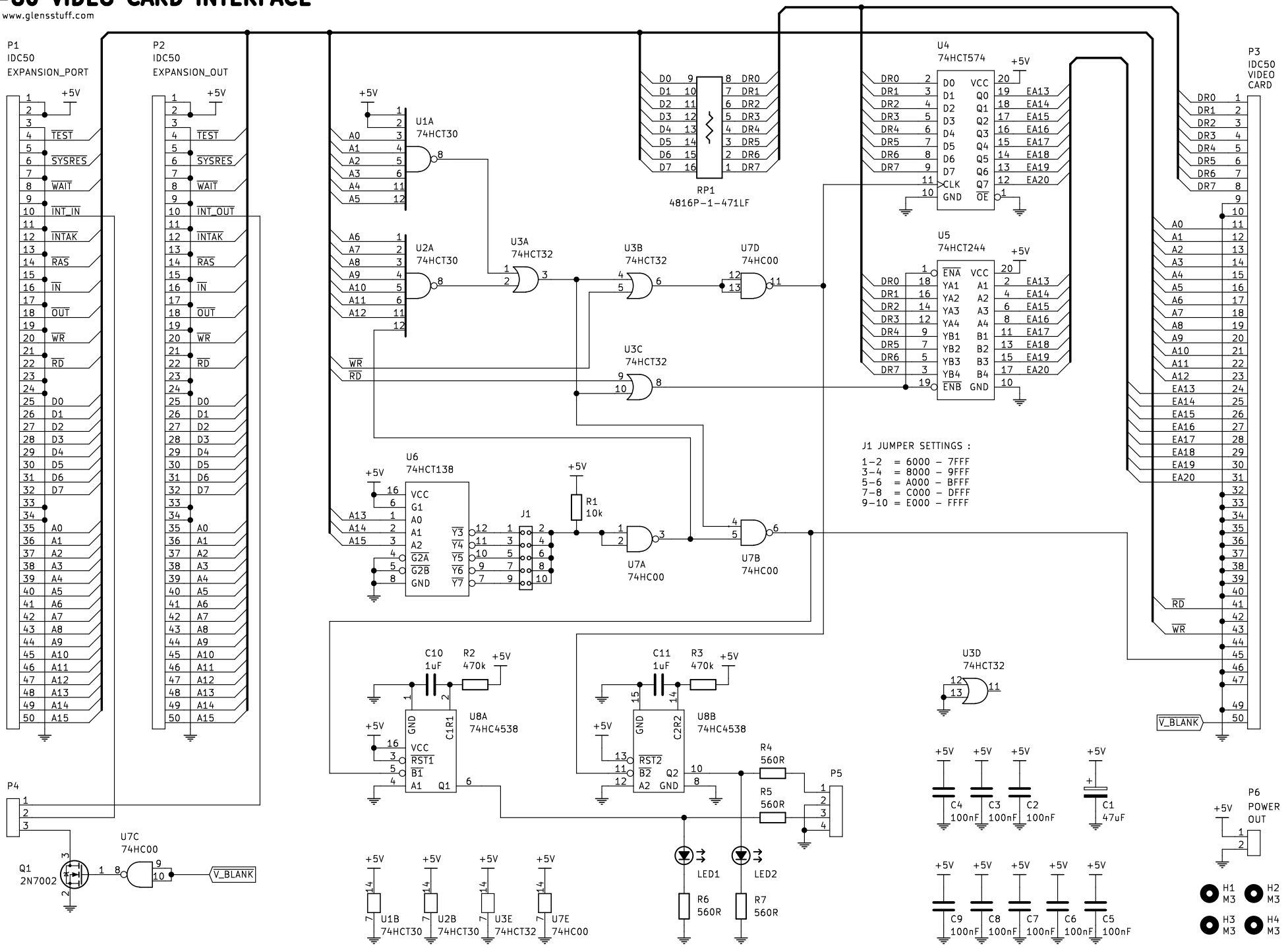
When the video memory is addressed during active screen time, my video card designs "blank the beam". This is in fact also how the TRS-80 handles its own native video display. It's a method that does mitigate the visual impact of the disturbance produced on the display, but you still do get a lot of random-looking black streaks on non-blank areas of the screen when the video memory is being accessed intensively.

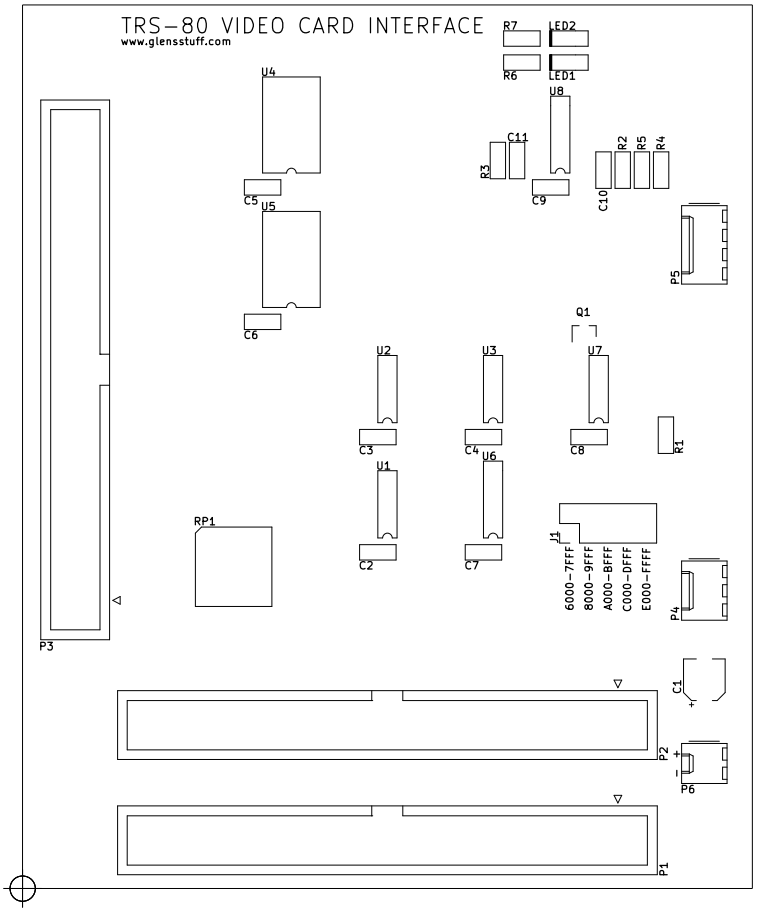
Confining access to the video memory to the vertical blanking period only, however, would greatly increase the amount of time taken to draw a complete image to the screen in a lot of instances. There are exceptions where it wouldn't make much difference at all though. A case in point is the Lorenz Attractor-plotting, BASIC example program presented elsewhere in this document. This program is doing some heavy math (for a 1.8 MHz-clocked Z80!) in BASIC and almost all of the run-time CPU overhead is consumed in calculation, rather than video memory updates. However, display-writes are consequently executed at low duty cycle and the visual disruption caused by the "beam blanking" is in fact now not that great at all.

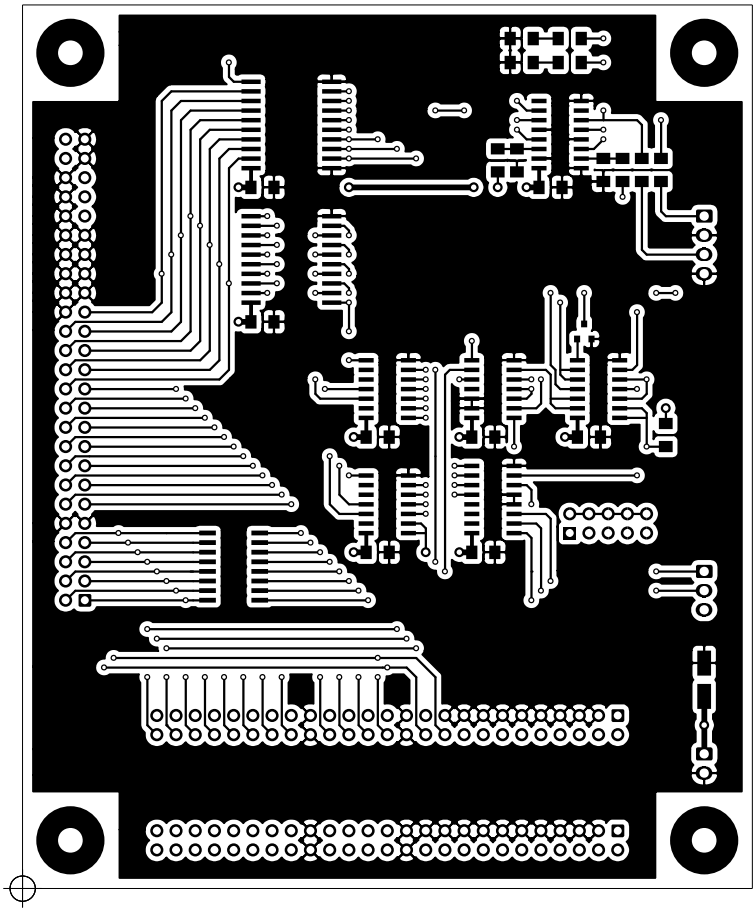
I didn't code support for blanking period synchronisation into my Stage 1 machine-code routines, as the intent with these routines was to draw static graphical displays and to simply plot the results of mathematical problems (geometric stuff, chaotic attractors, fractals, etc) to the display screen as expediently as practical. Making use of the potential for synchronisation is something that I have left open for exploration in the future.

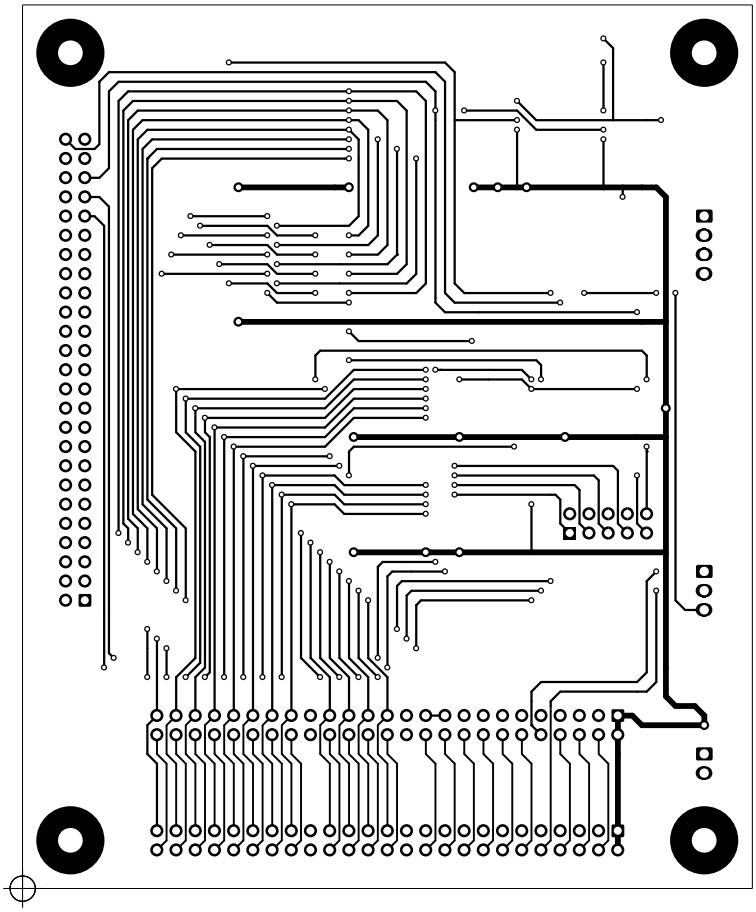
TRS-80 VIDEO CARD INTERFACE

MAY 2021 www.glensstuff.com




TRS-80 VIDEO CARD INTERFACE
www.glensstuff.com
R7
LED2
R6
LED1
U4
U8
C11
R2
R5
R4
R3
C5
C9
C10
U5
P5
C6
Q1
U2
U3
U7
C3
C4
C8
R1
U1
U6
RP1
J1
C2
C7
6000-7FFF
8000-9FFF
A000-BFFF
C000-DFFF
E000-FFFF
P4
P3
C1
P2
P6
P1





Parts for Video Card Interface PCB

Designator	Component type	Value	Part #	Manufacturer	Package	Description	Quantity
R1	Resistor	10k		Various	1206	Chip, thin film	1
R2, R3	Resistor	470k		Various	1206	Chip, thin film	2
R4, R5, R6, R7	Resistor	560R		Various	1206	Chip, thin film	4
RP1	Resistor array	470R x 8	4816P-1-471LF	Bournes	16 pin SMT		1
C1	Capacitor	47uF / 16V	EEE-1CA470WR	Panasonic	SMT	Electrolytic	1
C2, C3, C4, C5, C6, C7 ,C8, C9	Capacitor	100nF		Various	1206	Ceramic	8
C10, C11	Capacitor	1uF		Various	1206	Ceramic	2
U1, U2	I.C.		74HCT30	Various	SOIC	8-Input NAND gate	2
U3	I.C.		74HCT32	Various	SOIC	Quad 2-input OR gate	1
U4	I.C.		74HCT574	Various	SOIC - WIDE, 7.5mm	Octal D flip-flop	1
U5	I.C.		74HCT244	Various	SOIC - WIDE, 7.5mm	Octal buffer / line driver	1
U6	I.C.		74HCT138	Various	SOIC	3 to 8 line decoder / demulitplexer	1
U7	I.C.		74HC00	Various	SOIC	Quad 2-input NAND gate	1
U8	I.C.		74HC4538	Various	SOIC	Dual monostable	1
Q1	MOSFET		2N7002	Various	SOT-23		1
LED1, LED2	LED			Various	1206	Your choice of colour	2
P1, P2, P3	Connector			Various	TH	50-way IDC boxed header	3
P4	Connector		0022232031	Molex	TH	KK-254 vertical header, 3-way	1
P5	Connector		0022272041	Molex	TH	KK-254 vertical header, 4-way	1
P6	Connector		0022272021	Molex	TH	KK-254 vertical header, 2-way	1
J1 - part A	Header, male pins			Various	TH	Dual row, 12-way (2x6), 2.54mm	1
J1 - part B	Jumper, shorting			Various		2.54mm	1

Stage 1 graphics routines

Loading the Stage 1 routines

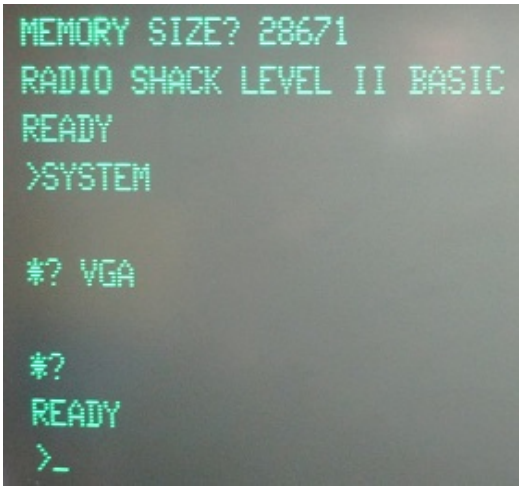
The Stage 1 machine-code routines are designed to load into to the top 4 KB of system RAM on a 16 KB machine. In hexadecimal, this is the address range 7000 to 7FFF, or, in decimal, 28672 to 32767.

When the computer is first turned on, this area of high memory needs to be reserved. This is achieved by entering, at the MEMORY SIZE? prompt, the address location *preceding* the start address of the memory to be reserved. Answer MEMORY SIZE? with 28671 and then press enter.

Next, to load the routines from cassette, type SYSTEM and press enter. At the *? prompt, type VGA and press enter. The cassette motor will now activate and the routines will start loading into the reserved area of high memory.

When loading is complete the *? prompt will appear again. At this point the computer is awaiting instruction prior to executing the machine-code program just loaded. Normally you would type / and then hit ENTER (to execute from the beginning address) or enter an alternative, higher address location if required.

The Stage 1 routines, however, are not intended to be run independently of BASIC. At this point instead you simply need to press the BREAK key to exit the SYSTEM command mode and return to the BASIC command mode. READY will be displayed, followed by the >_ prompt.



That's it! The Stage 1 routines now occupy high memory and you're ready to rock.

Accessing the Stage 1 routines from BASIC

In LEVEL II BASIC, machine-code subroutines are executed by calling them with the USR(X) function. This usually takes the form:

A=USR(X)

X is an optional variable to be transferred from BASIC to the machine-code routine and A is a variable returned to BASIC from the machine-code routine. Both A and X are signed 16-bit integers with a permissible range of -32768 to 32767.

A machine-code routine, however, need not necessarily transfer variables in either direction. In the case that there is no requirement to pass a variable from BASIC, X is simply substituted with a zero for practicality, although its value doesn't actually matter (the value between the brackets is not allowed to be left blank):

A=USR(0)

If the machine-code routine does not return a value to BASIC, A simply acquires the value entered between the brackets and can be ignored.

Prior to executing the USR function, the start address of the machine-code routine to be called needs to be set. When BASIC executes a USR statement it branches to the 16-bit address stored at locations 16526 and 16527. These two bytes are in the area of low memory reserved by the BASIC operating system. 16526 is the least significant byte of the 16-bit address and 16527 is the most significant byte. The values at 16526 and 16527 are modified with the POKE statement.

Display coordinates and programming overview

VGA video Card B has a display resolution of 640 x 480 pixels. For the horizontal, the pixel coordinate range is 1 to 640 and for the vertical it is 0 to 479. The origin is at the top left-hand corner of the screen – the uppermost and leftmost pixel has the coordinate values 1, 0 and the lowermost and rightmost pixel has the coordinate values 640, 479.

Quick-reference charts **Table 2** and **Table 3** on the following page list all of the individual Stage 1 machine-code routines and their respective start address values. The routines are divided into two separate sets – routines for transferring variables (pixel coordinate data, specifically) from BASIC and the routines that actually act upon the transferred pixel coordinate data and draw something to the VGA display.

Table 1 lists configuration variables. These are variables with data ranges that fall with the range of a single byte (0 – 255). Their values can be altered simply by POKEing the desired values to the corresponding address locations.

Note that the POKE 16527 address value is 112 for all machine-code routines. This corresponds to an address range of 7000 to 70FF (HEX). While most of the routines actually occupy address areas above 70FF, these routines are provided with indirect address jumps, all of which are located below 70FF. This was done to simplify and speed up operations in BASIC as 16527 now only ever needs to be set once at the beginning of your BASIC program, not changed alongside 16526 each and every time you need to call a different machine-code routine.

Variable name	Suggested abbreviation	POKE address	Range	Default value
Plot colour	PC	28673	0 - 63	63 (white)
Segment enable	SEG	28686	0 - 255	255
Row	ROW	28689	0 - 59	0
Column	COL	28687	0 - 79	0

Table 1

Routine	Suggested abbreviation	POKE 16527	POKE 16526	Address
Set Pixel X	PX	112	110	28782
Set Pixel Y	PY	112	117	28789
Set X1	X1	112	124	28796
Set Y1	Y1	112	131	28803
Set X2	X2	112	138	28810
Set Y2	Y2	112	145	28817

Table 2 – variable transfer routines

Routine	Suggested abbreviation	POKE 16527	POKE 16526	Address
Clear screen	CS	112	64	28736
Set pixel	SP	112	206	28878
Draw line (X1,Y1) - (X2,Y2)	L1	112	200	28872
Draw line - (X2,Y2)	L2	112	203	28875
Circle / arc	CA	112	197	28869
Circle, filled	CF	112	194	28866
Rectangle	RE	112	191	28863
Rectangle, filled	RF	112	188	28860
Bucket pour	BP	112	182	28854
Text	TXT	112	179	28851
Read pixel	RP	112	185	28857

Table 3 – drawing routines

Plot colour

This single-byte variable sets the drawing colour. There are 64 colours to choose from and the bit encoding is as follows:

Bit 0 (1) – Red, least significant bit
Bit 1 (2) – Red, most significant bit
Bit 2 (4) – Green, least significant bit
Bit 3 (8) – Green, most significant bit
Bit 4 (16) – Blue, least significant bit
Bit 5 (32) – Blue, most significant bit

For example, 0 = black and 15 = yellow. The default setting for **plot colour** is 63, which is white.

In BASIC the address location of **plot colour** can be assigned to the suggested variable name PC:

```
PC = 28673
```

The drawing colour can then be altered at any time with the following statement (where X specifies the colour):

```
POKE PC, X
```

Clear screen

This routine clears the video RAM by setting every pixel to a selected colour. The colour value is transferred via the USR statement and complies with the same bit encoding as **plot colour**.

The following BASIC code, for example, clears the screen to fully-saturated red:

```
10 POKE 16527, 112 : I = 16526 : CS = 64
.
.
100 POKE I, CS : A = USR(3)
```

The **clear screen** routine does not return a value to BASIC.

Variable transfer routines

These routines are listed in **Table 2**. These routines all operate in an identical fashion and their purpose is to facilitate the easy transfer (from BASIC) of the pixel coordinate data required by the various drawing routines. Each horizontal (X) coordinate value has a valid range of 1 to 640 and each vertical (V) coordinate value has a valid range of 0 to 479.

The following BASIC code, for example, sets variables **pixel X** and p**ixel Y** to 320 and 240 (roughly the centre of the screen) respectively:

```
10 POKE 16527, 112 : I = 16526 : PX = 110 : PY = 117
.
.
100 POKE I, PX : A = USR(320)
110 POKE I, PY : A = USR(240)
```

The desired values for **pixel X** and **pixel Y** are now stored in locations reserved for them within the area of high memory reserved for the Stage 1 machine-code routines.

None of the **variable transfer routines** return a value to BASIC.

Set pixel

This is the most elementary and least useful drawing routine. It sets any desired pixel to the colour currently defined by the variable **plot colour.** The pixel to be set is selected by specifying the coordinate values **pixel X** and **pixel Y**.

The **set pixel** routine neither receives from nor returns a variable to BASIC and is executed with the statement:

A = USR(0)

A will just acquire the value of zero.

Draw line (X1, Y1) – (X2, Y2)

This routine draws a line between two coordinates. Drawing starts at the coordinate specified by variables **X1** and **Y1** and terminates at the coordinate specified by variables **X2** and **Y2**. Either coordinate location can be anywhere on the screen.

This routine neither receives from nor returns a variable to BASIC and is executed with the statement:

A = USR(0)

A will just acquire the value of zero.

Draw line – (X2, Y2)

This routine works exactly as per **Draw line (X1, Y1) – (X2, Y2)** the first time it is executed, but once the line is drawn the coordinate variables **X1** and **Y1** are automatically reprogrammed to the end-point coordinate of the line just drawn – i.e., **X1** will now = **X2** and **Y1** will now = **Y2**. To draw the next line, with the starting point being the end of the line previously plotted, end-point coordinate variables **X2** and **Y2** need only be updated as required.

This routine is useful for joining sequentially computed points and is probably more versatile, overall, than **Draw line (X1, Y1) – (X2, Y2)**. At any time, the chain can be broken by redefining variables **X1** and **Y1**, prior to the next call, to define a new origin to start drawing from.

This routine neither receives from nor returns a variable to BASIC and is executed with the statement:

A = USR(0)

A will just acquire the value of zero.

Circle / arc

This routine draws a circle or an arc anywhere on the screen, about a centre point coordinate specified by variables **pixel X** and **pixel Y,** to a desired radius.

This routine receives the radius value via the USR statement and is executed with:

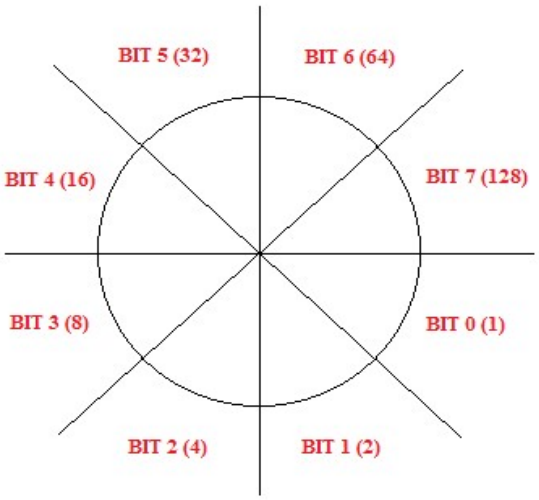
A = USR(RAD)

RAD is the radius value. This routine does not return a value to BASIC and A will acquire the value of RAD.

The result drawn to the screen is dependent upon the variable **segment enable** (see **Table 1**).

The **circle / arc** routine is based on Bresenham's mid-point circle algorithm, which rapidly computes one-eighth of a circle's circumference using integer-only arithmetic. The rest of the circle can then be drawn, as required, simply by mirroring and inverting this single computed octant as necessary. This makes it easy to add, with minimal extra code, a limited, but still useful capability for drawing arcs.

Each octant of the potential circle is assigned a bit of the variable **segment enable**:



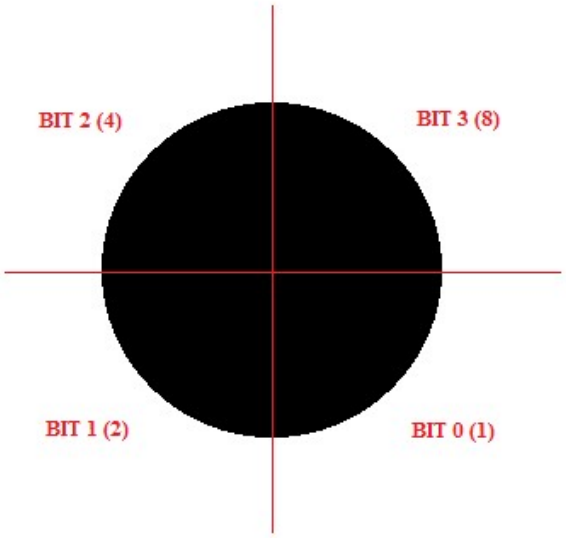
For a segment to be drawn, its assigned bit must be set to 1. This makes it possible to define and draw an arc with start and end points selectable to a resolution of 360 / 8 = 45 degrees.
Segmented circles can also be drawn. For example, setting **segment enable** to BIT0 + BIT3 + BIT4 + BIT7 = 153 will draw ( ). The default setting for **segment enable** is 255 (the sum of all bits), which draws a complete circle.

When more complex arcs are required, use the trigonometric functions SIN and COS in BASIC and join the computed points with **draw line – (X2, Y2).** This is of course a much slower operation.

Circle, filled

This routine works as per **circle / arc,** but instead of drawing only the circumference of the circle, the circle is drawn solid and the segmenting function is simplified from selectable octants to selectable quadrants.

Each quadrant of the potential solid circle is assigned a bit from the variable **segment enable** as follows:



The top four bits of **segment enable** are ignored. A value of 15 will result in a complete solid circle being plotted. Alternatively, a value of 1, for example, will draw the lower right-hand pie section only and a value of 12 will draw the complete top half of the circle.

Rectangle

This routine draws the outline of a rectangle. Variables **X1** and **Y1** set the upper, left-hand corner coordinate of the rectangle and variables **X2** and **Y2** set the lower, right-hand coordinate of the rectangle. These coordinate locations can be anywhere on the screen, but **X2** must be > **X1** and **Y2** must be > **Y1**. These conditions are tested at the beginning of the routine and it will exit back to BASIC without executing if they are not met.

This routine neither receives from nor returns a variable to BASIC and is executed with the statement:

A = USR(0)

A will just acquire the value of zero.

Rectangle, filled

Operates exactly as per **rectangle** but, as suggested by the name, the rectangle is drawn filled solid.

Bucket pour

This routine allows complex shapes to be easily filled with colour. Prior to execution, a "pour origin" must be set. This is a coordinate position located within the desired boundary to be filled and is specified by the variables **pixel X** and **pixel Y**. The fill colour is set by **plot colour**.

This routine does not return a value to BASIC and is executed with the statement:

A = USR(D)

D is a value specifying the direction of the pour, of which there are four permissible values:

1 = pour down
2 = pour up
4 = pour left
8 = pour right

The **bucket pour** routine works by firstly reading and then storing the original colour value of the pixel located at the pour origin, before changing it to the fill colour. In the case of a vertical-direction pour (D = either 1 or 2) it will then test and, provided that the colour tests the same as the stored value, set each pixel contiguous to the pour origin to the fill colour, both to the left and to the right of the pour origin. Encountering a different colour means that the boundary of the shape to be filled has been reached.

Once the boundary has been drawn to, both to the left and to the right of the pour origin, the vertical coordinate of the pour origin will be either incremented or decremented, depending on the direction of the pour. Unless the pour origin hits upon a colour different from the stored value the process described above will repeat all over again. The routine will terminate when the pour origin hits upon a different colour in the specified direction of the pour.

Operation for horizontal-direction pouring is as per the vertical-direction description; just substitute left and right with above and below and the vertical coordinate of the pour origin with the horizontal.

The **bucket pour** algorithm is a very simple one, built for speed with minimal condition testing. This simplicity comes with some limitations though. The algorithm only has an orthogonal, direct line-of-sight vision from the pour origin and it cannot see nor hunt around corners. Also, if the pour is to continue right down to the extremities of the boundary of the shape to be filled, in the specified direction of the pour, then the pour origin must be positioned in direct line-of-sight of that extremity.

None of this, however, means that there is a limit to the complexity of shapes that can be filled with the application of the **bucket pour** routine. It just means that more than one application of the routine, each with a new pour origin of course, might be required to completely fill a complex shape.

Text

This routine permits text and graphical characters to be easily printed to the screen. This is useful for, amongst other things, giving charts and graphs axis names and values. The text mode is fixed at 60 horizontal rows of 80 characters.

Before text is written to the screen, the starting location is specified by setting the **row** and **column** variables (see **Table 1**). The topmost and leftmost character position is the origin (**row** and **column** both set to 0). For example, setting **column** to 1 and **row** to 4 sets the position of the first text character to the second position across and to the fifth position down.

This routine does not return a value to BASIC and is executed with the statement:

A = USR(CHAR)

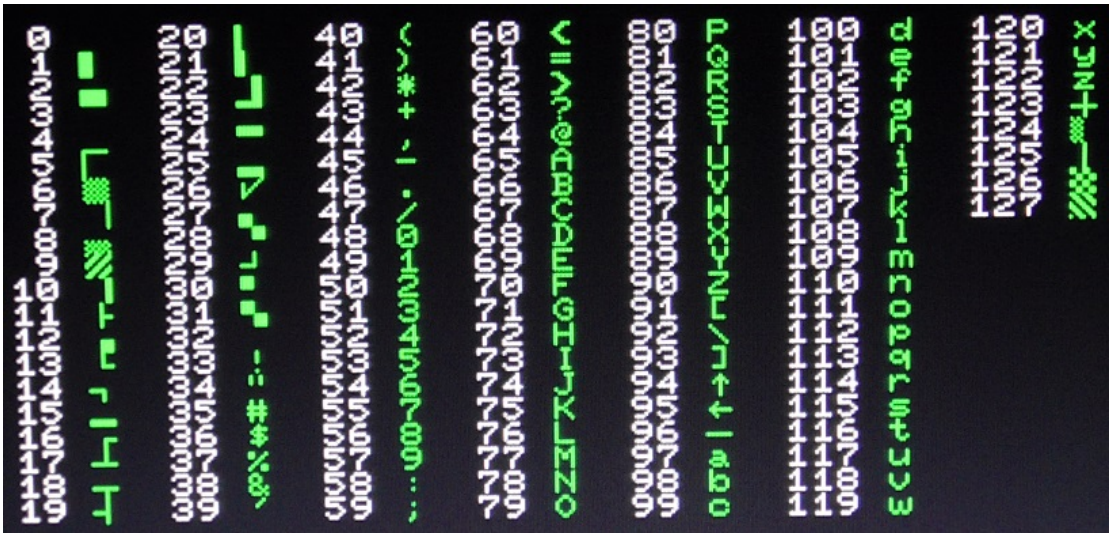
CHAR is the ASCII code for the character to be displayed.

The **column** variable is automatically incremented each time a character is printed to the screen and when the end of the line has been reached, **row** is incremented and **column** is reset to zero. When the very last character position of the screen has been written to – that is a character has been printed to **column** / **row** position 79, 59 - **column** and **row** are both reset to the origin (0, 0).

There are 128 pre-defined text/graphical characters to choose from. I actually used Commodore's "PETSCII" character set here, as per the original PET 2001 computer. Each character is defined by 8 bytes and the complete character set occupies the top 1 KB of my Stage 1 routines (address locations 0x7C00 to 0x7FFF) – see the Assembly Code listing further into this document.

Note that as these characters are defined in RAM, they are totally re-definable. You can re-define any selected character simply by POKEing alternative data values to the memory locations specific to that character.

Here are all of the "PETSCII" characters with their respective ASCII codes, printed to the screen using the **text** routine itself:



Read pixel

The very last routine and the only one that returns a value to BASIC. This routine permits the value of any video memory address location to be read. Variables **pixel X** and **pixel Y** specify the address to be read and the permissible range is not limited to the memory locations that map to pixels on the screen.

This routine is useful for block copying part of the video image to another part of the screen, or for reading back image data for saving.

Read pixel does not receive a variable from BASIC and is executed with the statement:

A = USR(0)

A will acquire the value of the selected memory location. An alternative statement is:

PRINT USR(0)

This will print the value of the selected memory location to the TRS-80's native video screen.

Alternatively, the PEEK statement can be used in BASIC to read the contents of the VGA card video memory, but you will need to POKE to the page register at 65,535, as the video card memory is addressed in 8 KB pages. Note that to PEEK or POKE memory locations above 32,767, Level II BASIC requires that the address value used in a PEEK or POKE statement takes the form: –(65536 – desired address). For example, POKE -1, VALUE writes VALUE to the page register of the video card interface.

The reason for writing **read pixel** into the repertoire of the Stage 1 routines is that it is required internally by the **bucket pour** routine.

The complete assembly listing

The “Stage 1” drawing routines were my first attempt at writing anything in machine-code in nearly twenty years; and anything at all for the Z80 microprocessor, besides a handful of instructions keyed into a Micro-Professor MPF-I once upon a time at TAFE. The MPF-1 was a very basic training computer for the Z80 microprocessor, hugely popular with educational institutions. But even back then, 8-bit microprocessors such as the Z80 were already a bit of an anachronism. Apart from legacy hardware, microcontrollers with flash memory already dominated the embedded landscape and I skipped pass machine-code and assembly language programming the moment I got my hands on my first C compiler.

Anyway, now that I’ve decided to dive back into the past, I think it turned out to be somewhat serendipitous that I initially picked the TRS-80 for this video card interfacing endeavour. To explain, I think that I might have fallen slightly in love with the Z80 microprocessor’s 16-bit register pairing and 16-bit arithmetic operations. The (integer-only) arithmetic required of Bresenham’s algorithms for plotting lines and circles with a pixel coordinate range of 1 through 640 for the horizontal and 0 through 479 for the vertical go beyond the computational range of eight measly bits, but fall safely within the limits that bound 16-bit operations.

Consequently, my code is predominately composed of commands from the Z80’s 16-bit Load and 16-bit Arithmetic instruction groups. As a bonus, TRS-80 Level II BASIC transfers variables to and from machine-code routines via the USR instruction in the form of 16-bit (two-byte) signed integers. These traits all amounted to making the writing of these routines a great deal more straightforward than they might otherwise have been. Now that I’ve become acquainted with the Z80 to this degree, I think that I would find it difficult to warm comparably to something like the register-deficient 6502.

While writing these routines was a fairly straightforward task, it was also a rather tedious one. This is without doubt the most laborious 4096 bytes of code that I have ever written and debugged. As presented on the following pages, I used Microsoft Excel on my workshop PC as a quasi-assembler that can be usefully read and commented. Data was entered into the TRS-80 by a combination of POKEs from DATA arrays in BASIC program listings and via the machine-code monitor program T-Bug. T-Bug was also used to save the completed routines residing in high memory as a SYSTEM-mode object file (saving to “cassette” with the PUNCH command).

With my machine-code / assembly experience now outlined, I’d now like to point out that I make no claims at all to the effect that my initial code might represent any kind of established protocol or state of the art, or that upon revision there won’t be scope for improvement. I only propose that my code works as claimed and demonstrated. I’ve called these routines “Stage 1” for a reason.

There is a lot of repetition in my routines. Lots of sections of code are repeats of other sections of code with only few variations, but there is good reason for this, and that reason is speed. When you’ve got a vintage microprocessor clocked at just under 1.8 MHz controlling a relatively high-resolution video display, clock cycles are at a premium. The Z80 isn’t exactly as clock cycle efficient as some modern processors. CALL/RET and JUMP instructions for bouncing between subroutines, not to mention excessive decision making, all burn clock cycles and can make a big difference to the time it takes a machine-code routine called by BASIC to complete its task. This is especially so if executed within a tight iterative loop computing the coordinates of hundreds or possibly thousands of pixels.

7000	28672	0	0		COLOUR BYTE, BACKGROUND	VARIABLES
7001	28673	3F	63		COLOUR BYTE, PIXEL	
7002	28674	1	1		PIXEL X, LSB	
7003	28675	0	0		PIXEL X, MSB	
7004	28676	0	0		PIXEL Y, LSB	
7005	28677	0	0		PIXEL Y, MSB	
7006	28678	1	1		X1, LSB	
7007	28679	0	0		X1, MSB	
7008	28680	0	0		Y1, LSB	
7009	28681	0	0		Y1, MSB	
700A	28682	1	1		X2, LSB	
700B	28683	0	0		X2, MSB	
700C	28684	0	0		Y2, LSB	
700D	28685	0	0		Y2, MSB	
700E	28686	FF	255		SEGMENT ENABLE	
700F	28687	0	0		COLUMN, LSB	
7010	28688	0	0		COLUMN, MSB (FOR MATH ONLY - LSB RANGE 0-79)	
7011	28689	0	0		ROW, LSB	
7012	28690	0	0		ROW, MSB (FOR MATH ONLY - LSB RANGE 0-59)	
7013	28691	0	0			
7014	28692	0	0			
7015	28693	0	0			
7016	28694	0	0			
7017	28695	0	0			
7018	28696	0	0			
7019	28697	0	0			
701A	28698	0	0			
701B	28699	0	0			
701C	28700	0	0			
701D	28701	0	0			
701E	28702	0	0			
701F	28703	0	0		CHAR LINE COUNT	FOR TEXT ROUTINE
7020	28704	0	0		CHARACTER CELL X ORIGIN, LSB	
7021	28705	0	0		CHARACTER CELL X ORIGIN, MSB	
7022	28706	0	0		CHAR ROM POINTER, LSB	
7023	28707	0	0		CHAR ROM POINTER, MSB	
7024	28708	1	1		POUR MODE	FOR BUCKET POUR ROUTINE
7025	28709	0	0		RADIUS, LSB	FOR DRAW CIRCLE/ARC FUNCTION
7026	28710	0	0		RADIUS, MSB	
7027	28711	0	0		OCTANT X, LSB	
7028	28712	0	0		OCTANT X, MSB	
7029	28713	0	0		OCTANT Y, LSB	
702A	28714	0	0		OCTANT Y, MSB	
702B	28715	0	0		DDX, LSB	
702C	28716	0	0		DDX, MSB	
702D	28717	0	0		DDY, LSB	
702E	28718	0	0		DDY, MSB	
702F	28719	0	0		F, LSB	
7030	28720	0	0		F, MSB	
7031	28721	0	0		CENTER X, LSB	
7032	28722	0	0		CENTER X, MSB	
7033	28723	0	0		CENTER Y, LSB	
7034	28724	0	0		CENTER Y, MSB	
7035	28725	0	0		F, LSB	FOR DRAW LINE FUNCTION
7036	28726	0	0		F, MSB	
7037	28727	0	0		DDX, LSB	
7038	28728	0	0		DDX, MSB	
7039	28729	0	0		DDY, LSB	
703A	28730	0	0		DDY, MSB	
703B	28731	0	0		DELTA X, LSB	
703C	28732	0	0		DELTA X, MSB	
703D	28733	0	0		DELTA Y, LSB	
703E	28734	0	0		DELTA Y, MSB	
703F	28735	0	0		PAGE ADDRESS	FOR VIDEO MEMORY CLEAR
7040	28736	CD	205	VIDEO MEMORY CLEAR ROUTINE		RETRIEVE BACKGROUND COLOUR VALUE FROM BASIC AND STORE
7041	28737	7F	127			
7042	28738	A	10	CALL 2687	GET COLOUR VALUE FROM BASIC	
7043	28739	45	69	LOAD B, L	LOAD COLOUR VALUE INTO REGISTER B	
7044	28740	21	33			
7045	28741	0	0			
7046	28742	70	112	LOAD HL, 28672		
7047	28743	70	112	LOAD (HL), B	SAVE COLOUR VALUE TO 28672	
7048	28744	21	33			UPDATE PAGE ADDRESS REGISTER
7049	28745	3F	63			
704A	28746	70	112	LOAD HL, 28735		
704B	28747	36	54			
704C	28748	3C	60	LOAD (HL), 60	SAVE VALUE 60 TO 28735	
704D	28749	46	70	LOAD B, (HL)	LOAD REGISTER B FROM 28735	
704E	28750	5	5	DEC B	B NOW = 59 (UPPER MEMORY PAGE ADDRESS). LOOP BEGINS HERE	
704F	28751	21	33			
7050	28752	FF	255			
7051	28753	FF	255	LOAD HL, 65535	ADDRESS OF PAGE REGISTER	
7052	28754	70	112	LOAD (HL), B	WRITE B TO PAGE REGISTER	
7053	28755	21	33			
7054	28756	0	0			
7055	28757	70	112	LOAD HL, 28672	ADDRESS OF BACKGROUND COLOUR VALUE	WRITE BACKGROUND COLOUR VALUE TO ALL LOCATIONS OF CURRENT PAGE
7056	28758	46	70	LOAD B, (HL)	LOAD BACKGROUND COLOUR VALUE INTO REGISTER B	
7057	28759	21	33			
7058	28760	0	0			
7059	28761	E0	224	LOAD HL, 57344	START ADDRESS FOR VIDEO RAM	
705A	28762	70	112	LOAD (HL), B	WRITE COLOUR VALUE TO VIDEO RAM START ADDRESS	
705B	28763	11	17			
705C	28764	1	1			
705D	28765	E0	224	LOAD DE, 57345	LOAD DE REGISTER WITH VIDEO RAM START ADDRESS+1	
705E	28766	1	1			
705F	28767	80	128			
7060	28768	1E	30	LOAD BC, 7808	LOAD BC REGISTER WITH NUMBER OF ITERATIONS	
7061	28769	ED	237			
7062	28770	B0	176	LIDR	COPY COLOUR VALUE TO REMAINING VIDEO RAM PAGE LOCATIONS	
7063	28771	21	33			

7064	28772	3F	63			REPEAT UNTIL ALL PAGES (59 TO 0) ARE WRITTEN
7065	28773	70	112	LOAD HL, 28735	ADDRESS OF CURRENT PAGE ADDRESS	
7066	28774	46	70	LOAD B, (HL)	RETRIEVE PAGE ADDRESS TO REGISTER B	
7067	28775	10	16			
7068	28776	1	1	DJNZ, 1	DECREMENT PAGE ADDRESS VALUE (IN REGISTER B) AND SKIP RETURN TO BASIC IF NOT ZERO	
7069	28777	C9	201	RET	RETURN TO BASIC	
706A	28778	70	112	LOAD (HL), B	SAVE PAGE ADDRESS VALUE TO 28735	
706B	28779	C3	195			
706C	28780	4E	78			
706D	28781	70	112	JP, 28750	UNCONDITIONAL JUMP TO 28750	
706E	28782	CD	205	SET PIXEL X VALUE FROM BASIC ROUTINE		CALL BASIC SUBROUTINE TO RECEIVE VARIABLE FROM USR(VAR)
706F	28783	7F	127			
7070	28784	A	10	CALL 2687	GET PIXEL X VALUE FROM BASIC - USR(PIXEL X) - POKE 16527,112 - POKE 16526, 110	
7071	28785	22	34			SAVE VARIABLE AND RETURN TO BASIC
7072	28786	2	2			
7073	28787	70	112	LOAD (28674), HL	SAVE PIXEL X TO 28674 - 28675	
7074	28788	C9	201	RET	RETURN TO BASIC	
7075	28789	CD	205	SET PIXEL Y VALUE FROM BASIC ROUTINE		CALL BASIC SUBROUTINE TO RECEIVE VARIABLE FROM USR(VAR)
7076	28790	7F	127			
7077	28791	A	10	CALL 2687	GET PIXEL Y VALUE FROM BASIC - USR(PIXEL Y) - POKE 16527,112 - POKE 16526, 117	
7078	28792	22	34			SAVE VARIABLE AND RETURN TO BASIC
7079	28793	4	4			
707A	28794	70	112	LOAD (28676), HL	SAVE PIXEL Y TO 28676 - 28677	
707B	28795	C9	201	RET	RETURN TO BASIC	
707C	28796	CD	205	SET X1 VALUE FROM BASIC ROUTINE		CALL BASIC SUBROUTINE TO RECEIVE VARIABLE FROM USR(VAR)
707D	28797	7F	127			
707E	28798	A	10	CALL 2687	GET X1 VALUE FROM BASIC - USR(X1) - POKE 16527,112 - POKE 16526, 124	
707F	28799	22	34			SAVE VARIABLE AND RETURN TO BASIC
7080	28800	6	6			
7081	28801	70	112	LOAD (28678), HL	SAVE X1 TO 28678 - 28679	
7082	28802	C9	201	RET	RETURN TO BASIC	
7083	28803	CD	205	SET Y1 VALUE FROM BASIC ROUTINE		CALL BASIC SUBROUTINE TO RECEIVE VARIABLE FROM USR(VAR)
7084	28804	7F	127			
7085	28805	A	10	CALL 2687	GET Y1 VALUE FROM BASIC - USR(Y1) - POKE 16527,112 - POKE 16526, 131	
7086	28806	22	34			SAVE VARIABLE AND RETURN TO BASIC
7087	28807	8	8			
7088	28808	70	112	LOAD (28680), HL	SAVE Y1 TO 28680 - 28681	
7089	28809	C9	201	RET	RETURN TO BASIC	
708A	28810	CD	205	SET X2 VALUE FROM BASIC ROUTINE		CALL BASIC SUBROUTINE TO RECEIVE VARIABLE FROM USR(VAR)
708B	28811	7F	127			
708C	28812	A	10	CALL 2687	GET X2 VALUE FROM BASIC - USR(X2) - POKE 16527,112 - POKE 16526, 138	
708D	28813	22	34			SAVE VARIABLE AND RETURN TO BASIC
708E	28814	A	10			
708F	28815	70	112	LOAD (28682), HL	SAVE X2 TO 28682 - 28683	
7090	28816	C9	201	RET	RETURN TO BASIC	
7091	28817	CD	205	SET Y2 VALUE FROM BASIC ROUTINE		CALL BASIC SUBROUTINE TO RECEIVE VARIABLE FROM USR(VAR)
7092	28818	7F	127			
7093	28819	A	10	CALL 2687	GET Y2 VALUE FROM BASIC - USR(Y2) - POKE 16527,112 - POKE 16526, 145	
7094	28820	22	34			SAVE VARIABLE AND RETURN TO BASIC
7095	28821	C	12			
7096	28822	70	112	LOAD (28684), HL	SAVE Y2 TO 28684 - 28685	
7097	28823	C9	201	RET	RETURN TO BASIC	
7098	28824	0	0			
7099	28825	0	0			
709A	28826	0	0			
709B	28827	0	0			
709C	28828	0	0			
709D	28829	0	0			
709E	28830	0	0			
709F	28831	0	0			
70A0	28832	0	0			
70A1	28833	0	0			
70A2	28834	0	0			
70A3	28835	0	0			
70A4	28836	0	0			
70A5	28837	0	0			
70A6	28838	0	0			
70A7	28839	0	0			
70A8	28840	0	0			
70A9	28841	0	0			
70AA	28842	0	0			
70AB	28843	0	0			
70AC	28844	0	0			
70AD	28845	0	0			
70AE	28846	0	0			
70AF	28847	0	0			
70B0	28848	0	0			
70B1	28849	0	0			
70B2	28850	0	0			
70B3	28851	C3	195			THESE ARE INDIRECT JUMPS FOR CALLING ROUTINES LOCATED ABOVE 0x70FF. THIS SPEEDS UP BASIC AS 16527 NOW ONLY NEEDS TO BE POKED ONCE (TO 112)
70B4	28852	E4	228			
70B5	28853	7A	122	JP, 31460	INDIRECT JUMP TO **TEXT** ROUTINE	
70B6	28854	C3	195			
70B7	28855	F4	244			
70B8	28856	79	121	JP, 31220	INDIRECT JUMP TO **BUCKET POUR** ROUTINE	
70B9	28857	C3	195			
70BA	28858	BC	188			
70BB	28859	79	121	JP, 31062	INDIRECT JUMP TO **READ PIXEL** ROUTINE	
70BC	28860	C3	195			
70BD	28861	56	86			
70BE	28862	79	121	JP, 31062	INDIRECT JUMP TO **DRAW FILLED RECTANGLE**	
70BF	28863	C3	195			
70C0	28864	BD	189			
70C1	28865	78	120	JP, 30909	INDIRECT JUMP TO **DRAW OUTLINE RECTANGLE**	
70C2	28866	C3	195			
70C3	28867	96	150			
70C4	28868	78	120	JP, 31043	INDIRECT JUMP TO **DRAW SOLID CIRCLE**	
70C5	28869	C3	195			
70C6	28870	43	67			
70C7	28871	75	117	JP, 30019	INDIRECT JUMP TO **DRAW CIRCLE/ARC**	
70C8	28872	C3	195			
70C9	28873	9F	159			
70CA	28874	74	116	JP, 29855	INDIRECT JUMP TO **DRAW LINE((X1, Y1) - (X2, Y2))**	

70CB	28875	C3	195			
70CC	28876	D3	211			
70CD	28877	74	116	JP, 29907	INDIRECT JUMP TO **DRAW LINE(- (X2, Y2))**	
70CE	28878	ED	237	SET PIXEL ROUTINE		
70CF	28879	5B	91			
70D0	28880	4	4			
70D1	28881	70	112	LOAD DE, (28676)	LOAD PIXEL Y INTO DE	
70D2	28882	CB	203			
70D3	28883	3B	59	SRL, E		
70D4	28884	CB	203			
70D5	28885	3B	59	SRL, E		
70D6	28886	CB	203			
70D7	28887	3B	59	SRL, E		
70D8	28888	CB	203			
70D9	28889	42	66	BIT 0, D	TEST BIT 0, D	
70DA	28890	28	40			
70DB	28891	2	2	JR Z, 2	SKIP NEXT INSTRUCTION IF BIT 0 = 0	
70DC	28892	CB	203			
70DD	28893	EB	235	SET 5, E	SET BIT 5, E	
70DE	28894	21	33			
70DF	28895	FF	255			
70E0	28896	FF	255	LOAD HL, 65535	ADDRESS OF PAGE REGISTER	
70E1	28897	73	115	LOAD (HL), E	WRITE E TO PAGE REGISTER	
70E2	28898	21	33			
70E3	28899	4	4			
70E4	28900	70	112	LOAD HL, 28676		
70E5	28901	56	86	LOAD D, (HL)	COPY PIXEL Y LSB INTO D	
70E6	28902	CB	203			
70E7	28903	22	34	SLA, D		
70E8	28904	CB	203			
70E9	28905	22	34	SLA, D		
70EA	28906	CB	203			
70EB	28907	FA	250	SET 7, D	E000 VIDEO MEMORY OFFSET	
70EC	28908	CB	203			
70ED	28909	F2	242	SET 6, D	E000 VIDEO MEMORY OFFSET	
70EE	28910	CB	203			
70EF	28911	EA	234	SET 5, D	E000 VIDEO MEMORY OFFSET	
70F0	28912	1E	30			
70F1	28913	0	0	LOAD E, 0	DE NOW CONTAINS LINE NUMBER (0-7) * 1024 + 0xE000 OFFSET	
70F2	28914	21	33			
70F3	28915	1	1			
70F4	28916	70	112	LOAD HL, 28673		
70F5	28917	4E	78	LOAD C, (HL)	FETCH PIXEL COLOUR BYTE INTO C	
70F6	28918	2A	42			
70F7	28919	2	2			
70F8	28920	70	112	LOAD HL, (28674)	FETCH PIXEL X INTO HL	
70F9	28921	19	25	ADD HL, DE	16-BIT ADDITION HL+DE. HL NOW CONTAINS PIXEL ADDRESS	
70FA	28922	71	113	LOAD (HL), C	WRITE COLOUR VALUE TO VIDEO RAM	
70FB	28923	C9	201	RET	RETURN	
70FC	28924	0	0			
70FD	28925	0	0			
70FE	28926	0	0			
70FF	28927	0	0			
7100	28928	0	0			
7101	28929	0	0			
7102	28930	0	0			
7103	28931	0	0			
7104	28932	0	0			
7105	28933	0	0			
7106	28934	0	0			
7107	28935	2A	42	BRESENHAM'S LINE ALGORITHM FOR X2=>X1 AND Y2=>Y1 (QUADRANT A)		
7108	28936	A	10			
7109	28937	70	112	LOAD HL, (28682)	LOAD X2 INTO HL	
710A	28938	ED	237			
710B	28939	5B	91			
710C	28940	6	6			
710D	28941	70	112	LOAD DE, (28678)	LOAD X1 INTO DE	
710E	28942	37	55	SCF		
710F	28943	3F	63	CCF	CARRY FLAG TO ZERO	
7110	28944	ED	237			
7111	28945	52	82	SBC HL, DE	SUBTRACT X1 FROM X2. HL NOW CONTAINS DELTA X	
7112	28946	22	34			
7113	28947	3B	59			
7114	28948	70	112	LOAD (28731), HL	SAVE DELTA X TO 28731 - 28732	
7115	28949	CB	203			
7116	28950	25	37	SLA, L	MULTIPLY DELTA X BY TWO, STEP 1 OF 2	
7117	28951	CB	203			
7118	28952	14	20	RL, H	MULTIPLY DELTA X BY TWO, STEP 2 OF 2	
7119	28953	22	34			
711A	28954	37	55			
711B	28955	70	112	LOAD (28727), HL	SAVE DDX TO 28727 - 28728	
711C	28956	2A	42			
711D	28957	C	12			
711E	28958	70	112	LOAD HL, (28684)	LOAD Y2 INTO HL	
711F	28959	ED	237			
7120	28960	5B	91			
7121	28961	8	8			
7122	28962	70	112	LOAD DE, (28680)	LOAD Y1 INTO DE	
7123	28963	37	55	SCF		
7124	28964	3F	63	CCF	CARRY FLAG TO ZERO	
7125	28965	ED	237			
7126	28966	52	82	SBC HL, DE	SUBTRACT Y1 FROM Y2. HL NOW CONTAINS DELTA Y	
7127	28967	22	34			
7128	28968	3D	61			
7129	28969	70	112	LOAD (28733), HL	SAVE DELTA Y TO 28733 - 28734	
712A	28970	CB	203			
712B	28971	25	37	SLA, L	MULTIPLY DELTA Y BY TWO, STEP 1 OF 2	
712C	28972	CB	203			
712D	28973	14	20	RL, H	MULTIPLY DELTA Y BY TWO, STEP 2 OF 2	
712E	28974	22	34			
712F	28975	39	57			
7130	28976	70	112	LOAD (28729), HL	SAVE DDY TO 28729 - 28730	
7131	28977	2A	42			

COMPUTE PAGE FROM Y COORDINATE AND WRITE TO PAGE ADDRESS REGISTER. THERE ARE 60 PAGES OF 8 LINES EACH FOR THE VERTICAL RESOLUTION OF 480 PIXELS.

EXTRACT LINE NUMBER OF PAGE AND ADD BASE ADDRESS VALUE (0xE000) FOR THE VIDEO RAM

ADD X COORDINATE VALUE TO COMPLETE THE PIXEL ADDRESS AND WRITE COLOUR VALUE

RETURN TO BASIC

DELTA X = X2 - X1

DDX = 2 * DELTA X

DELTA Y = Y2 - Y1

DDY = 2 * DELTA Y

7132	28978	3B	59			IF DELTA Y > DELTA X THEN GO TO THE OPPOSITE OCTANT OF THIS QUADRANT
7133	28979	70	112	LOAD HL, (28731)	LOAD DELTA X INTO HL	
7134	28980	ED	237			
7135	28981	5B	91			
7136	28982	3D	61			
7137	28983	70	112	LOAD DE, (28733)	LOAD DELTA Y INTO DE	
7138	28984	37	55	SCF		
7139	28985	3F	63	CCF	CARRY FLAG TO ZERO	
713A	28986	ED	237			
713B	28987	52	82	SBC HL, DE	SUBTRACT DELTA Y FROM DELTA X. SIGN NEGATIVE IF DELTA Y > DELTA X	
713C	28988	FA	250			
713D	28989	97	151			
713E	28990	71	113	JP M, 29079	JUMP TO NEXT OCTANT IF DELTA Y > DELTA X (SIGN NEGATIVE)	
713F	28991	2A	42			F = DDY - DELTA X
7140	28992	39	57			
7141	28993	70	112	LOAD HL, (28729)	LOAD DDY INTO HL	
7142	28994	ED	237			
7143	28995	5B	91			
7144	28996	3B	59			
7145	28997	70	112	LOAD DE, (28731)	LOAD DELTA X INTO DE	
7146	28998	37	55	SCF		
7147	28999	3F	63	CCF	CARRY FLAG TO ZERO	
7148	29000	ED	237			
7149	29001	52	82	SBC HL, DE	SUBTRACT DELTA X FROM DDY	
714A	29002	22	34			
714B	29003	35	53			
714C	29004	70	112	LOAD (28725), HL	SAVE F TO 28725 - 28726	
714D	29005	CD	205		****** LOOP RETURNS TO HERE ******	PLOT POINT & ITERATIVE LOOP RETURN POINT
714E	29006	CE	206			
714F	29007	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
7150	29008	21	33			IF F > 0 THEN DO NEXT TWO OPERATIONS, ELSE SKIP
7151	29009	0	0			
7152	29010	0	0	LOAD HL, 0	HL TO ZERO	
7153	29011	ED	237			
7154	29012	5B	91			
7155	29013	35	53			
7156	29014	70	112	LOAD DE, (28725)	LOAD F INTO DE	
7157	29015	37	55	SCF		
7158	29016	3F	63	CCF	CARRY FLAG TO ZERO	
7159	29017	ED	237			
715A	29018	52	82	SBC HL, DE	SUBTRACT F FROM ZERO. SIGN NEGATIVE IF F > 0	
715B	29019	F2	242			
715C	29020	73	115			
715D	29021	71	113	JP P, 29043	JUMP Y-INCREMENT IF F <= 0 (SIGN POSITIVE)	
715E	29022	2A	42			PIXEL Y = PIXEL Y + 1
715F	29023	4	4			
7160	29024	70	112	LOAD HL, (28676)	LOAD PIXEL Y INTO HL	
7161	29025	23	35	INC HL	PIXEL Y = PIXEL Y + 1	
7162	29026	22	34			
7163	29027	4	4			
7164	29028	70	112	LOAD (28676), HL	SAVE UPDATED PIXEL Y VALUE	
7165	29029	2A	42			F = F - DDX
7166	29030	35	53			
7167	29031	70	112	LOAD HL, (28725)	LOAD F INTO HL	
7168	29032	ED	237			
7169	29033	5B	91			
716A	29034	37	55			
716B	29035	70	112	LOAD DE, (28727)	LOAD DDX INTO DE	
716C	29036	37	55	SCF		
716D	29037	3F	63	CCF	CARRY FLAG TO ZERO	
716E	29038	ED	237			
716F	29039	52	82	SBC HL, DE	SUBTRACT DDX FROM F	
7170	29040	22	34			
7171	29041	35	53			
7172	29042	70	112	LOAD (28725),HL	SAVE F TO 28725 - 28726	
7173	29043	2A	42		****** Y-INCREMENT JUMP GOES TO HERE ******	F = F + DDY
7174	29044	35	53			
7175	29045	70	112	LOAD HL, (28725)	LOAD F INTO HL.	
7176	29046	ED	237			
7177	29047	5B	91			
7178	29048	39	57			
7179	29049	70	112	LOAD DE, (28729)	LOAD DDY INTO DE	
717A	29050	19	25	ADD HL, DE	F = F + DDY	
717B	29051	22	34			
717C	29052	35	53			
717D	29053	70	112	LOAD (28725), HL	SAVE F TO 28725 - 28726	
717E	29054	ED	237			PIXEL X = PIXEL X + 1
717F	29055	5B	91			
7180	29056	2	2			
7181	29057	70	112	LOAD DE, (28674)	LOAD PIXEL X INTO DE	
7182	29058	13	19	INC DE	PIXEL X = PIXEL X + 1	
7183	29059	ED	237			
7184	29060	53	83			
7185	29061	2	2			
7186	29062	70	112	LOAD (28674), DE	SAVE UPDATED PIXEL X VALUE	
7187	29063	2A	42			IF PIXEL X <= X2 THEN DO NEXT ITERATION, ELSE END
7188	29064	A	10			
7189	29065	70	112	LOAD HL, (28682)	LOAD X2 INTO HL	
718A	29066	37	55	SCF		
718B	29067	3F	63	CCF	CARRY FLAG TO ZERO	
718C	29068	ED	237			
718D	29069	52	82	SBC HL, DE	SUBTRACT PIXEL X FROM X2. SIGN NEGATIVE IF PIXEL X > X2	
718E	29070	F2	242			
718F	29071	4D	77			
7190	29072	71	113	JP P, 29005	IF PIXEL X <= X2 (SIGN POSITIVE) THEN DO NEXT ITERATION	
7191	29073	1B	27	DEC DE	PIXEL X = PIXEL X - 1 = X2	PIXEL X = X2
7192	29074	ED	237			
7193	29075	53	83			
7194	29076	2	2			
7195	29077	70	112	LOAD (28674), DE	SAVE UPDATED PIXEL X VALUE	
7196	29078	C9	201	RET		END OCTANT
7197	29079	2A	42		****** SECOND OCTANT STARTS HERE ********	
7198	29080	37	55			

7199	29081	70	112	LOAD HL, (28727)	LOAD DDX INTO HL	F = DDX - DELTA Y
719A	29082	ED	237			
719B	29083	5B	91			
719C	29084	3D	61			
719D	29085	70	112	LOAD DE, (28733)	LOAD DELTA Y INTO DE	
719E	29086	37	55	SCF		
719F	29087	3F	63	CCF	CARRY FLAG TO ZERO	
71A0	29088	ED	237			
71A1	29089	52	82	SBC HL, DE	SUBTRACT DELTA Y FROM DDX	
71A2	29090	22	34			
71A3	29091	35	53			
71A4	29092	70	112	LOAD (28725), HL	SAVE F TO 0x7035 - 0x7036	
71A5	29093	CD	205		****** LOOP RETURNS TO HERE ******	PLOT POINT & ITERATIVE LOOP RETURN POINT
71A6	29094	CE	206			
71A7	29095	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
71A8	29096	21	33			IF F > 0 THEN DO NEXT TWO OPERATIONS, ELSE SKIP
71A9	29097	0	0			
71AA	29098	0	0	LOAD HL, 0	HL TO ZERO	
71AB	29099	ED	237			
71AC	29100	5B	91			
71AD	29101	35	53			
71AE	29102	70	112	LOAD DE, (28725)	LOAD F INTO DE	
71AF	29103	37	55	SCF		
71B0	29104	3F	63	CCF	CARRY FLAG TO ZERO	
71B1	29105	ED	237			
71B2	29106	52	82	SBC HL, DE	SUBTRACT F FROM ZERO. SIGN NEGATIVE IF F > 0	
71B3	29107	F2	242			
71B4	29108	CB	203			
71B5	29109	71	113	JP P, 29131	JUMP X-INCREMENT IF F <= 0 (SIGN POSITIVE)	
71B6	29110	2A	42			PIXEL X = PIXEL X + 1
71B7	29111	2	2			
71B8	29112	70	112	LOAD HL, (28674)	LOAD PIXEL X INTO HL	
71B9	29113	23	35	INC HL	PIXEL X = PIXEL X + 1	
71BA	29114	22	34			
71BB	29115	2	2			
71BC	29116	70	112	LOAD (28674), HL	SAVE UPDATED PIXEL X VALUE	
71BD	29117	2A	42			F = F - DDY
71BE	29118	35	53			
71BF	29119	70	112	LOAD HL, 28725	LOAD F INTO HL	
71C0	29120	ED	237			
71C1	29121	5B	91			
71C2	29122	39	57			
71C3	29123	70	112	LOAD DE, 28729	LOAD DDY INTO DE	
71C4	29124	37	55	SCF		
71C5	29125	3F	63	CCF	CARRY FLAG TO ZERO	
71C6	29126	ED	237			
71C7	29127	52	82	SBC HL, DE	SUBTRACT DDY FROM F	
71C8	29128	22	34			
71C9	29129	35	53			
71CA	29130	70	112	LOAD (28725), HL	SAVE F TO 28725 - 28726	
71CB	29131	2A	42		****** X-INCREMENT JUMP GOES TO HERE ******	F = F + DDX
71CC	29132	35	53			
71CD	29133	70	112	LOAD HL, 28725	LOAD F INTO HL	
71CE	29134	ED	237			
71CF	29135	5B	91			
71D0	29136	37	55			
71D1	29137	70	112	LOAD DE, 28727	LOAD DDX INTO DE	
71D2	29138	19	25	ADD HL, DE	F = F + DDX	
71D3	29139	22	34			
71D4	29140	35	53			
71D5	29141	70	112	LOAD (28725), HL	SAVE F TO 28725 - 28726	PIXEL Y = PIXEL Y + 1
71D6	29142	ED	237			
71D7	29143	5B	91			
71D8	29144	4	4			
71D9	29145	70	112	LOAD DE, 28676	LOAD PIXEL Y INTO DE	
71DA	29146	13	19	INC DE	PIXEL Y = PIXEL Y + 1	
71DB	29147	ED	237			
71DC	29148	53	83			
71DD	29149	4	4			
71DE	29150	70	112	LOAD (28676), DE	SAVE UPDATED PIXEL Y VALUE	
71DF	29151	2A	42			IF PIXEL Y2 <= PIXEL Y THEN DO NEXT ITERATION, ELSE END
71E0	29152	C	12			
71E1	29153	70	112	LOAD HL, (28684)	LOAD Y2 INTO HL	
71E2	29154	37	55	SCF		
71E3	29155	3F	63	CCF	CARRY FLAG TO ZERO	
71E4	29156	ED	237			
71E5	29157	52	82	SBC HL, DE	SUBTRACT PIXEL Y FROM Y2. SIGN NEGATIVE IF PIXEL Y > Y2	
71E6	29158	F2	242			
71E7	29159	A5	165			
71E8	29160	71	113	JP P, 29093	IF PIXEL Y <= Y2 (SIGN POSITIVE) THEN DO NEXT ITERATION	
71E9	29161	1B	27	DEC DE	PIXEL Y = PIXEL Y - 1 = Y2	PIXEL Y = Y2
71EA	29162	ED	237			
71EB	29163	53	83			
71EC	29164	4	4			
71ED	29165	70	112	LOAD (28676), DE	SAVE UPDATED PIXEL Y VALUE	
71EE	29166	C9	201	RET		END OCTANT
71EF	29167	2A	42	BRESENHAM'S LINE ALGORITHM FOR X2<X1 AND Y2=>Y1 (QUADRANT B)		DELTA X = X1 - X2
71F0	29168	6	6			
71F1	29169	70	112	LOAD HL, (28678)	LOAD X1 INTO HL	
71F2	29170	ED	237			
71F3	29171	5B	91			
71F4	29172	A	10			
71F5	29173	70	112	LOAD DE, (286768)	LOAD X2 INTO DE	
71F6	29174	37	55	SCF		
71F7	29175	3F	63	CCF	CARRY FLAG TO ZERO	
71F8	29176	ED	237			
71F9	29177	52	82	SBC HL, DE	SUBTRACT X2 FROM X1. HL NOW CONTAINS DELTA X	
71FA	29178	22	34			
71FB	29179	3B	59			
71FC	29180	70	112	LOAD (28731), HL	SAVE DELTA X TO 28731 - 28732	
71FD	29181	CB	203			
71FE	29182	25	37	SLA, L	MULTIPLY DELTA X BY TWO, STEP 1 OF 2	
71FF	29183	CB	203			

7200	29184	14	20	RL, H	MULTIPLY DELTA X BY TWO, STEP 2 OF 2	DDX = 2 * DELTA X
7201	29185	22	34			
7202	29186	37	55			
7203	29187	70	112	LOAD (28727), HL	SAVE DDX TO 28727 - 28728	
7204	29188	2A	42			DELTA Y = Y2 - Y1
7205	29189	C	12			
7206	29190	70	112	LOAD HL, (28684)	LOAD Y2 INTO HL	
7207	29191	ED	237			
7208	29192	5B	91			
7209	29193	8	8			
720A	29194	70	112	LOAD DE, (28680)	LOAD Y1 INTO DE	
720B	29195	37	55	SCF		
720C	29196	3F	63	CCF	CARRY FLAG TO ZERO	
720D	29197	ED	237			
720E	29198	52	82	SBC HL, DE	SUBTRACT Y1 FROM Y2. HL NOW CONTAINS DELTA Y	
720F	29199	22	34			
7210	29200	3D	61			
7211	29201	70	112	LOAD (28733), HL	SAVE DELTA Y TO 28733 - 28734	
7212	29202	CB	203			DDY = 2 * DELTA Y
7213	29203	25	37	SLA, L	MULTIPLY DELTA Y BY TWO, STEP 1 OF 2	
7214	29204	CB	203			
7215	29205	14	20	RL, H	MULTIPLY DELTA Y BY TWO, STEP 2 OF 2	
7216	29206	22	34			
7217	29207	39	57			
7218	29208	70	112	LOAD (28729), HL	SAVE DDY TO 28729 - 28730	
7219	29209	2A	42			IF DELTA Y > DELTA X THEN GO TO THE OPPOSITE OCTANT OF THIS QUADRANT
721A	29210	3B	59			
721B	29211	70	112	LOAD HL, (28731)	LOAD DELTA X INTO HL	
721C	29212	ED	237			
721D	29213	5B	91			
721E	29214	3D	61			
721F	29215	70	112	LOAD DE, (28733)	LOAD DELTA Y INTO DE	
7220	29216	37	55	SCF		
7221	29217	3F	63	CCF	CARRY FLAG TO ZERO	
7222	29218	ED	237			
7223	29219	52	82	SBC HL, DE	SUBTRACT DELTA Y FROM DELTA X. SIGN NEGATIVE IF DELTA Y > DELTA X	
7224	29220	FA	250			
7225	29221	7D	125			
7226	29222	72	114	JP M, 29309	JUMP TO NEXT OCTANT IF DELTA Y > DELTA X (SIGN NEGATIVE)	
7227	29223	2A	42			F = DDY - DELTA X
7228	29224	39	57			
7229	29225	70	112	LOAD HL, (28729)	LOAD DDY INTO HL	
722A	29226	ED	237			
722B	29227	5B	91			
722C	29228	3B	59			
722D	29229	70	112	LOAD DE, (28731)	LOAD DELTA X INTO DE	
722E	29230	37	55	SCF		
722F	29231	3F	63	CCF	CARRY FLAG TO ZERO	
7230	29232	ED	237			
7231	29233	52	82	SBC HL, DE	SUBTRACT DELTA X FROM DDY	
7232	29234	22	34			
7233	29235	35	53			
7234	29236	70	112	LOAD (28725), HL	SAVE F TO 28725 - 28726	
7235	29237	CD	205		****** LOOP RETURNS TO HERE ******	PLOT POINT & ITERATIVE LOOP RETURN POINT
7236	29238	CE	206			
7237	29239	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
7238	29240	21	33			IF F > 0 THEN DO NEXT TWO OPERATIONS, ELSE SKIP
7239	29241	0	0			
723A	29242	0	0	LOAD HL, 0	HL TO ZERO	
723B	29243	ED	237			
723C	29244	5B	91			
723D	29245	35	53			
723E	29246	70	112	LOAD DE, (28725)	LOAD F INTO DE	
723F	29247	37	55	SCF		
7240	29248	3F	63	CCF	CARRY FLAG TO ZERO	
7241	29249	ED	237			
7242	29250	52	82	SBC HL, DE	SUBTRACT F FROM ZERO. SIGN NEGATIVE IF F > 0	
7243	29251	F2	242			
7244	29252	5B	91			
7245	29253	72	114	JP P, 29275	JUMP Y-INCREMENT IF F <= 0 (SIGN POSITIVE)	
7246	29254	2A	42			PIXEL Y = PIXEL Y + 1
7247	29255	4	4			
7248	29256	70	112	LOAD HL, (28676)	LOAD PIXEL Y INTO HL	
7249	29257	23	35	INC HL	PIXEL Y = PIXEL Y + 1	
724A	29258	22	34			
724B	29259	4	4			
724C	29260	70	112	LOAD (28676), HL	SAVE UPDATED PIXEL Y VALUE	
724D	29261	2A	42			F = F - DDX
724E	29262	35	53			
724F	29263	70	112	LOAD HL, (28725)	LOAD F INTO HL	
7250	29264	ED	237			
7251	29265	5B	91			
7252	29266	37	55			
7253	29267	70	112	LOAD DE, (28727)	LOAD DDX INTO DE	
7254	29268	37	55	SCF		
7255	29269	3F	63	CCF	CARRY FLAG TO ZERO	
7256	29270	ED	237			
7257	29271	52	82	SBC HL, DE	SUBTRACT DDX FROM F	
7258	29272	22	34			
7259	29273	35	53			
725A	29274	70	112	LOAD (28725),HL	SAVE F TO 28725 - 28726	
725B	29275	2A	42		****** Y-INCREMENT JUMP GOES TO HERE ******	F = F + DDY
725C	29276	35	53			
725D	29277	70	112	LOAD HL, (28725)	LOAD F INTO HL.	
725E	29278	ED	237			
725F	29279	5B	91			
7260	29280	39	57			
7261	29281	70	112	LOAD DE, (28729)	LOAD DDY INTO DE	
7262	29282	19	25	ADD HL, DE	F = F + DDY	
7263	29283	22	34			
7264	29284	35	53			
7265	29285	70	112	LOAD (28725), HL	SAVE F TO 28725 - 28726	
7266	29286	2A	42			

7267	29287	2	2			PIXEL X = PIXEL X - 1
7268	29288	70	112	LOAD HL, (28674)	LOAD PIXEL X INTO HL	
7269	29289	2B	43	DEC HL	PIXEL X = PIXEL X - 1	
726A	29290	22	34			
726B	29291	2	2			
726C	29292	70	112	LOAD (28674), HL	SAVE UPDATED PIXEL X VALUE	
726D	29293	ED	237			IF PIXEL X2 <= PIXEL X THEN DO NEXT ITERATION
726E	29294	5B	91			
726F	29295	A	10			
7270	29296	70	112	LOAD DE, (28682)	LOAD X2 INTO DE	
7271	29297	37	55	SCF		
7272	29298	3F	63	CCF	CARRY FLAG TO ZERO	
7273	29299	ED	237			
7274	29300	52	82	SBC HL, DE	SUBTRACT X2 FROM PIXEL X. SIGN NEGATIVE IF X2 > PIXEL X	
7275	29301	F2	242			
7276	29302	35	53			
7277	29303	72	114	JP P, 29237	IF X2 <= PIXEL X (SIGN POSITIVE) THEN DO NEXT ITERATION	
7278	29304	ED	237			PIXEL X = X2
7279	29305	53	83			
727A	29306	2	2			
727B	29307	70	112	LOAD (28674), DE	SAVE X2 TO PIXEL X	
727C	29308	C9	201	RET		END OCTANT
727D	29309	2A	42		****** SECOND OCTANT STARTS HERE ********	F = DDX - DELTA Y
727E	29310	37	55			
727F	29311	70	112	LOAD HL, (28727)	LOAD DDX INTO HL	
7280	29312	ED	237			
7281	29313	5B	91			
7282	29314	3D	61			
7283	29315	70	112	LOAD DE, (28733)	LOAD DELTA Y INTO DE	
7284	29316	37	55	SCF		
7285	29317	3F	63	CCF	CARRY FLAG TO ZERO	
7286	29318	ED	237			
7287	29319	52	82	SBC HL, DE	SUBTRACT DELTA Y FROM DDX	
7288	29320	22	34			
7289	29321	35	53			
728A	29322	70	112	LOAD (28725), HL	SAVE F TO 28725 - 28726	
728B	29323	CD	205		****** LOOP RETURNS TO HERE ******	PLOT POINT & ITERATIVE LOOP RETURN POINT
728C	29324	CE	206			
728D	29325	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
728E	29326	21	33			IF F > 0 THEN DO NEXT TWO OPERATIONS, ELSE SKIP
728F	29327	0	0			
7290	29328	0	0	LOAD HL, 0	HL TO ZERO	
7291	29329	ED	237			
7292	29330	5B	91			
7293	29331	35	53			
7294	29332	70	112	LOAD DE, (28725)	LOAD F INTO DE	
7295	29333	37	55	SCF		
7296	29334	3F	63	CCF	CARRY FLAG TO ZERO	
7297	29335	ED	237			
7298	29336	52	82	SBC HL, DE	SUBTRACT F FROM ZERO. SIGN NEGATIVE IF F > 0	
7299	29337	F2	242			
729A	29338	B1	177			
729B	29339	72	114	JP P, 29361	JUMP X-INCREMENT IF F <= 0 (SIGN POSITIVE)	
729C	29340	2A	42			PIXEL X = PIXEL X - 1
729D	29341	2	2			
729E	29342	70	112	LOAD HL, (28674)	LOAD PIXEL X INTO HL	
729F	29343	2B	43	DEC HL	PIXEL X = PIXEL X - 1	
72A0	29344	22	34			
72A1	29345	2	2			
72A2	29346	70	112	LOAD (28674), HL	SAVE UPDATED PIXEL X VALUE	
72A3	29347	2A	42			F = F - DDY
72A4	29348	35	53			
72A5	29349	70	112	LOAD HL, 28725	LOAD F INTO HL	
72A6	29350	ED	237			
72A7	29351	5B	91			
72A8	29352	39	57			
72A9	29353	70	112	LOAD DE, 28729	LOAD DDY INTO DE	
72AA	29354	37	55	SCF		
72AB	29355	3F	63	CCF	CARRY FLAG TO ZERO	
72AC	29356	ED	237			
72AD	29357	52	82	SBC HL, DE	SUBTRACT DDY FROM F	
72AE	29358	22	34			
72AF	29359	35	53			
72B0	29360	70	112	LOAD (28725), HL	SAVE F TO 28725 - 28726	
72B1	29361	2A	42		****** X-INCREMENT JUMP GOES TO HERE ******	F = F + DDX
72B2	29362	35	53			
72B3	29363	70	112	LOAD HL, 28725	LOAD F INTO HL	
72B4	29364	ED	237			
72B5	29365	5B	91			
72B6	29366	37	55			
72B7	29367	70	112	LOAD DE, 28727	LOAD DDX INTO DE	
72B8	29368	19	25	ADD HL, DE	F = F + DDX	
72B9	29369	22	34			
72BA	29370	35	53			
72BB	29371	70	112	LOAD (28725), HL	SAVE F TO 28725 - 28726	
72BC	29372	ED	237			PIXEL Y = PIXEL Y + 1
72BD	29373	5B	91			
72BE	29374	4	4			
72BF	29375	70	112	LOAD DE, 28676	LOAD PIXEL Y INTO DE	
72C0	29376	13	19	INC DE	PIXEL Y = PIXEL Y + 1	
72C1	29377	ED	237			
72C2	29378	53	83			
72C3	29379	4	4			
72C4	29380	70	112	LOAD (28676), DE	SAVE UPDATED PIXEL Y VALUE	
72C5	29381	2A	42			IF PIXEL Y2 <= PIXEL Y THEN DO NEXT ITERATION, ELSE END
72C6	29382	C	12			
72C7	29383	70	112	LOAD HL, (28684)	LOAD Y2 INTO HL	
72C8	29384	37	55	SCF		
72C9	29385	3F	63	CCF	CARRY FLAG TO ZERO	
72CA	29386	ED	237			
72CB	29387	52	82	SBC HL, DE	SUBTRACT PIXEL Y FROM Y2. SIGN NEGATIVE IF PIXEL Y > Y2	
72CC	29388	F2	242			
72CD	29389	8B	139			

72CE	29390	72	114	JP P, 29323	IF PIXEL Y <= Y2 (SIGN POSITIVE) THEN DO NEXT ITERATION. ELSE END IF PIXEL Y > Y2	
72CF	29391	1B	27	DEC DE	PIXEL Y = PIXEL Y - 1 = Y2	
72D0	29392	ED	237			PIXEL Y = Y2
72D1	29393	53	83			
72D2	29394	4	4			
72D3	29395	70	112	LOAD (28676), DE	SAVE UPDATED PIXEL Y VALUE	
72D4	29396	C9	201	RET		END OCTANT
72D5	29397	2A	42	BRESENHAM'S LINE ALGORITHM FOR X2<X1 AND Y2<Y1 (QUADRANT C)		
72D6	29398	6	6			
72D7	29399	70	112	LOAD HL, (28678)	LOAD X1 INTO HL	
72D8	29400	ED	237			
72D9	29401	5B	91			
72DA	29402	A	10			
72DB	29403	70	112	LOAD DE, (286768)	LOAD X2 INTO DE	
72DC	29404	37	55	SCF		DELTA X = X1 - X2
72DD	29405	3F	63	CCF	CARRY FLAG TO ZERO	
72DE	29406	ED	237			
72DF	29407	52	82	SBC HL, DE	SUBTRACT X2 FROM X1. HL NOW CONTAINS DELTA X	
72E0	29408	22	34			
72E1	29409	3B	59			
72E2	29410	70	112	LOAD (28731), HL	SAVE DELTA X TO 28731 - 28732	
72E3	29411	CB	203			
72E4	29412	25	37	SLA, L	MULTIPLY DELTA X BY TWO, STEP 1 OF 2	
72E5	29413	CB	203			
72E6	29414	14	20	RL, H	MULTIPLY DELTA X BY TWO, STEP 2 OF 2	DDX = 2 * DELTA X
72E7	29415	22	34			
72E8	29416	37	55			
72E9	29417	70	112	LOAD (28727), HL	SAVE DDX TO 28727 - 28728	
72EA	29418	2A	42			
72EB	29419	8	8			
72EC	29420	70	112	LOAD HL, (28680)	LOAD Y1 INTO HL	
72ED	29421	ED	237			
72EE	29422	5B	91			
72EF	29423	C	12			
72F0	29424	70	112	LOAD DE, (28684)	LOAD Y2 INTO DE	DELTA Y = Y1 - Y2
72F1	29425	37	55	SCF		
72F2	29426	3F	63	CCF	CARRY FLAG TO ZERO	
72F3	29427	ED	237			
72F4	29428	52	82	SBC HL, DE	SUBTRACT Y2 FROM Y1. HL NOW CONTAINS DELTA Y	
72F5	29429	22	34			
72F6	29430	3D	61			
72F7	29431	70	112	LOAD (28733), HL	SAVE DELTA Y TO 28733 - 28734	
72F8	29432	CB	203			
72F9	29433	25	37	SLA, L	MULTIPLY DELTA Y BY TWO, STEP 1 OF 2	
72FA	29434	CB	203			
72FB	29435	14	20	RL, H	MULTIPLY DELTA Y BY TWO, STEP 2 OF 2	DDY = 2 * DELTA Y
72FC	29436	22	34			
72FD	29437	39	57			
72FE	29438	70	112	LOAD (28729), HL	SAVE DDY TO 28729 - 28730	
72FF	29439	2A	42			
7300	29440	3B	59			
7301	29441	70	112	LOAD HL, (28731)	LOAD DELTA X INTO HL	
7302	29442	ED	237			
7303	29443	5B	91			
7304	29444	3D	61			
7305	29445	70	112	LOAD DE, (28733)	LOAD DELTA Y INTO DE	
7306	29446	37	55	SCF		IF DELTA Y > DELTA X THEN GO TO THE OPPOSITE OCTANT OF THIS QUADRANT
7307	29447	3F	63	CCF	CARRY FLAG TO ZERO	
7308	29448	ED	237			
7309	29449	52	82	SBC HL, DE	SUBTRACT DELTA Y FROM DELTA X. SIGN NEGATIVE IF DELTA Y > DELTA X	
730A	29450	FA	250			
730B	29451	63	99			
730C	29452	73	115	JP M, 29539	JUMP TO NEXT OCTANT IF DELTA Y > DELTA X (SIGN NEGATIVE)	
730D	29453	2A	42			
730E	29454	39	57			
730F	29455	70	112	LOAD HL, (28729)	LOAD DDY INTO HL	
7310	29456	ED	237			
7311	29457	5B	91			
7312	29458	3B	59			
7313	29459	70	112	LOAD DE, (28731)	LOAD DELTA X INTO DE	
7314	29460	37	55	SCF		F = DDY - DELTA X
7315	29461	3F	63	CCF	CARRY FLAG TO ZERO	
7316	29462	ED	237			
7317	29463	52	82	SBC HL, DE	SUBTRACT DELTA X FROM DDY	
7318	29464	22	34			
7319	29465	35	53			
731A	29466	70	112	LOAD (28725), HL	SAVE F TO 28725 - 28726	
731B	29467	CD	205		****** LOOP RETURNS TO HERE ******	PLOT POINT & ITERATIVE LOOP RETURN POINT
731C	29468	CE	206			
731D	29469	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
731E	29470	21	33			
731F	29471	0	0			
7320	29472	0	0	LOAD HL, 0	HL TO ZERO	
7321	29473	ED	237			
7322	29474	5B	91			
7323	29475	35	53			
7324	29476	70	112	LOAD DE, (28725)	LOAD F INTO DE	IF F > 0 THEN DO NEXT TWO OPERATIONS, ELSE SKIP
7325	29477	37	55	SCF		
7326	29478	3F	63	CCF	CARRY FLAG TO ZERO	
7327	29479	ED	237			
7328	29480	52	82	SBC HL, DE	SUBTRACT F FROM ZERO. SIGN NEGATIVE IF F > 0	
7329	29481	F2	242			
732A	29482	41	65			
732B	29483	73	115	JP P, 29505	JUMP IF F <= 0 (SIGN POSITIVE)	
732C	29484	2A	42			
732D	29485	4	4			
732E	29486	70	112	LOAD HL, (28676)	LOAD PIXEL Y INTO HL	
732F	29487	2B	43	DEC HL	PIXEL Y = PIXEL Y - 1	PIXEL Y = PIXEL Y - 1
7330	29488	22	34			
7331	29489	4	4			
7332	29490	70	112	LOAD (28676), HL	SAVE UPDATED PIXEL Y VALUE	
7333	29491	2A	42			
7334	29492	35	53			

7335	29493	70	112	LOAD HL, (28725)	LOAD F INTO HL	F = F - DDX
7336	29494	ED	237			
7337	29495	5B	91			
7338	29496	37	55			
7339	29497	70	112	LOAD DE, (28727)	LOAD DDX INTO DE	
733A	29498	37	55	SCF		
733B	29499	3F	63	CCF	CARRY FLAG TO ZERO	
733C	29500	ED	237			
733D	29501	52	82	SBC HL, DE	SUBTRACT DDX FROM F	
733E	29502	22	34			
733F	29503	35	53			
7340	29504	70	112	LOAD (28725),HL	SAVE F TO 28725 - 28726	
7341	29505	2A	42			F = F + DDY
7342	29506	35	53			
7343	29507	70	112	LOAD HL, (28725)	LOAD F INTO HL.	
7344	29508	ED	237			
7345	29509	5B	91			
7346	29510	39	57			
7347	29511	70	112	LOAD DE, (28729)	LOAD DDY INTO DE	
7348	29512	19	25	ADD HL, DE	F = F + DDY	
7349	29513	22	34			
734A	29514	35	53			
734B	29515	70	112	LOAD (28725), HL	SAVE F TO 28725 - 28726	
734C	29516	2A	42			PIXEL X = PIXEL X - 1
734D	29517	2	2			
734E	29518	70	112	LOAD HL, (28674)	LOAD PIXEL X INTO HL	
734F	29519	2B	43	DEC HL	PIXEL X = PIXEL X - 1	
7350	29520	22	34			
7351	29521	2	2			
7352	29522	70	112	LOAD (28674), HL	SAVE UPDATED PIXEL X VALUE	
7353	29523	ED	237			IF PIXEL X2 <= PIXEL X THEN DO NEXT ITERATION, ELSE END
7354	29524	5B	91			
7355	29525	A	10			
7356	29526	70	112	LOAD DE, (28682)	LOAD X2 INTO DE	
7357	29527	37	55	SCF		
7358	29528	3F	63	CCF	CARRY FLAG TO ZERO	
7359	29529	ED	237			
735A	29530	52	82	SBC HL, DE	SUBTRACT X2 FROM PIXEL X. SIGN NEGATIVE IF X2 > PIXEL X	
735B	29531	F2	242			
735C	29532	1B	27			
735D	29533	73	115	JP P, 29467	IF X2 <= PIXEL X (SIGN POSITIVE) THEN DO NEXT ITERATION.	
735E	29534	ED	237			PIXEL X = X2
735F	29535	53	83			
7360	29536	2	2			
7361	29537	70	112	LOAD (28674), DE	SAVE X2 TO PIXEL X	
7362	29538	C9	201	RET		END OCTANT
7363	29539	2A	42		****** SECOND OCTANT STARTS HERE ********	F = DDX - DELTA Y
7364	29540	37	55			
7365	29541	70	112	LOAD HL, (28727)	LOAD DDX INTO HL	
7366	29542	ED	237			
7367	29543	5B	91			
7368	29544	3D	61			
7369	29545	70	112	LOAD DE, (28733)	LOAD DELTA Y INTO DE	
736A	29546	37	55	SCF		
736B	29547	3F	63	CCF	CARRY FLAG TO ZERO	
736C	29548	ED	237			
736D	29549	52	82	SBC HL, DE	SUBTRACT DELTA Y FROM DDX	
736E	29550	22	34			
736F	29551	35	53			
7370	29552	70	112	LOAD (28725), HL	SAVE F TO 28725 - 28726	
7371	29553	CD	205		****** LOOP RETURNS TO HERE ******	PLOT POINT & ITERATIVE LOOP RETURN POINT
7372	29554	CE	206			
7373	29555	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
7374	29556	21	33			IF F > 0 THEN DO NEXT TWO OPERATIONS, ELSE SKIP
7375	29557	0	0			
7376	29558	0	0	LOAD HL, 0	HL TO ZERO	
7377	29559	ED	237			
7378	29560	5B	91			
7379	29561	35	53			
737A	29562	70	112	LOAD DE, (28725)	LOAD F INTO DE	
737B	29563	37	55	SCF		
737C	29564	3F	63	CCF	CARRY FLAG TO ZERO	
737D	29565	ED	237			
737E	29566	52	82	SBC HL, DE	SUBTRACT F FROM ZERO. SIGN NEGATIVE IF F > 0	
737F	29567	F2	242			
7380	29568	97	151			
7381	29569	73	115	JP P, 29591	JUMP X-INCREMENT IF F <= 0 (SIGN POSITIVE)	
7382	29570	2A	42			PIXEL X = PIXEL X - 1
7383	29571	2	2			
7384	29572	70	112	LOAD HL, (28674)	LOAD PIXEL X INTO HL	
7385	29573	2B	43	DEC HL	PIXEL X = PIXEL X - 1	
7386	29574	22	34			
7387	29575	2	2			
7388	29576	70	112	LOAD (28674), HL	SAVE UPDATED PIXEL X VALUE	
7389	29577	2A	42			F = F - DDY
738A	29578	35	53			
738B	29579	70	112	LOAD HL, 28725	LOAD F INTO HL	
738C	29580	ED	237			
738D	29581	5B	91			
738E	29582	39	57			
738F	29583	70	112	LOAD DE, 28729	LOAD DDY INTO DE	
7390	29584	37	55	SCF		
7391	29585	3F	63	CCF	CARRY FLAG TO ZERO	
7392	29586	ED	237			
7393	29587	52	82	SBC HL, DE	SUBTRACT DDY FROM F	
7394	29588	22	34			
7395	29589	35	53			
7396	29590	70	112	LOAD (28725), HL	SAVE F TO 28725 - 28726	
7397	29591	2A	42		****** X-INCREMENT JUMP GOES TO HERE ******	
7398	29592	35	53			
7399	29593	70	112	LOAD HL, 28725	LOAD F INTO HL	
739A	29594	ED	237			
739B	29595	5B	91			

739C	29596	37	55			F = F + DDX
739D	29597	70	112	LOAD DE, 28727	LOAD DDX INTO DE	
739E	29598	19	25	ADD HL, DE	F = F + DDX	
739F	29599	22	34			
73A0	29600	35	53			
73A1	29601	70	112	LOAD (28725), HL	SAVE F TO 28725 - 28726	
73A2	29602	2A	42			PIXEL Y = PIXEL Y - 1
73A3	29603	4	4			
73A4	29604	70	112	LOAD HL, 28676	LOAD PIXEL Y INTO HL	
73A5	29605	2B	43	DEC HL	PIXEL Y = PIXEL Y - 1	
73A6	29606	22	34			
73A7	29607	4	4			
73A8	29608	70	112	LOAD (28676), HL	SAVE UPDATED PIXEL Y VALUE	
73A9	29609	ED	237			IF PIXEL Y2 <= PIXEL Y THEN DO NEXT ITERATION, ELSE END
73AA	29610	5B	91			
73AB	29611	C	12			
73AC	29612	70	112	LOAD DE, (28684)	LOAD Y2 INTO DE	
73AD	29613	37	55	SCF		
73AE	29614	3F	63	CCF	CARRY FLAG TO ZERO	
73AF	29615	ED	237			
73B0	29616	52	82	SBC HL, DE	SUBTRACT Y2 FROM PIXEL Y. SIGN NEGATIVE IF PIXEL Y2 > PIXEL Y	
73B1	29617	F2	242			
73B2	29618	71	113			
73B3	29619	73	115	JP P, 29553	IF Y2 <= PIXEL Y (SIGN POSITIVE) THEN DO NEXT ITERATION. ELSE END IF PIXEL Y2 > PIXEL Y	
73B4	29620	ED	237			PIXEL Y = Y2
73B5	29621	53	83			
73B6	29622	4	4			
73B7	29623	70	112	LOAD (28676), DE	SAVE X2 TO PIXEL Y	
73B8	29624	C9	201	RET		END OCTANT
73B9	29625	2A	42	BRESENHAM'S LINE ALGORITHM FOR X2=>X1 AND Y2<Y1 (QUADRANT D)		
73BA	29626	A	10			
73BB	29627	70	112	LOAD HL, (28682)	LOAD X2 INTO HL	DELTA X = X2 - X1
73BC	29628	ED	237			
73BD	29629	5B	91			
73BE	29630	6	6			
73BF	29631	70	112	LOAD DE, (28678)	LOAD X1 INTO DE	
73C0	29632	37	55	SCF		
73C1	29633	3F	63	CCF	CARRY FLAG TO ZERO	
73C2	29634	ED	237			
73C3	29635	52	82	SBC HL, DE	SUBTRACT X1 FROM X2. HL NOW CONTAINS DELTA X.	
73C4	29636	22	34			
73C5	29637	3B	59			
73C6	29638	70	112	LOAD (28731), HL	SAVE DELTA X TO 28731 - 28732	
73C7	29639	CB	203			DDX = 2 * DELTA X
73C8	29640	25	37	SLA, L	MULTIPLY DELTA X BY TWO, STEP 1 OF 2	
73C9	29641	CB	203			
73CA	29642	14	20	RL, H	MULTIPLY DELTA X BY TWO, STEP 2 OF 2	
73CB	29643	22	34			
73CC	29644	37	55			
73CD	29645	70	112	LOAD (28727), HL	SAVE DDX TO 28727 - 28728	
73CE	29646	2A	42			DELTA Y = Y2 - Y1
73CF	29647	8	8			
73D0	29648	70	112	LOAD HL, (28680)	LOAD Y1 INTO HL	
73D1	29649	ED	237			
73D2	29650	5B	91			
73D3	29651	C	12			
73D4	29652	70	112	LOAD DE, (28684)	LOAD Y2 INTO DE	
73D5	29653	37	55	SCF		
73D6	29654	3F	63	CCF	CARRY FLAG TO ZERO	
73D7	29655	ED	237			
73D8	29656	52	82	SBC HL, DE	SUBTRACT Y2 FROM Y1. HL NOW CONTAINS DELTA Y	
73D9	29657	22	34			
73DA	29658	3D	61			
73DB	29659	70	112	LOAD (28733), HL	SAVE DELTA Y TO 28733 - 28734	
73DC	29660	CB	203			DDY = 2 * DELTA Y
73DD	29661	25	37	SLA, L	MULTIPLY DELTA Y BY TWO, STEP 1 OF 2	
73DE	29662	CB	203			
73DF	29663	14	20	RL, H	MULTIPLY DELTA Y BY TWO, STEP 2 OF 2	
73E0	29664	22	34			
73E1	29665	39	57			
73E2	29666	70	112	LOAD (28729), HL	SAVE DDY TO 28729 - 28730	
73E3	29667	2A	42			IF DELTA Y > DELTA X THEN GO TO THE OPPOSITE OCTANT OF THIS QUADRANT
73E4	29668	3B	59			
73E5	29669	70	112	LOAD HL, (28731)	LOAD DELTA X INTO HL	
73E6	29670	ED	237			
73E7	29671	5B	91			
73E8	29672	3D	61			
73E9	29673	70	112	LOAD DE, (28733)	LOAD DELTA Y INTO DE	
73EA	29674	37	55	SCF		
73EB	29675	3F	63	CCF	CARRY FLAG TO ZERO	
73EC	29676	ED	237			
73ED	29677	52	82	SBC HL, DE	SUBTRACT DELTA Y FROM DELTA X. SIGN NEGATIVE IF DELTA Y > DELTA X	
73EE	29678	FA	250			
73EF	29679	49	73			
73F0	29680	74	116	JP M, 29769	JUMP TO NEXT OCTANT IF DELTA Y > DELTA X (SIGN NEGATIVE)	
73F1	29681	2A	42			F = DDY - DELTA X
73F2	29682	39	57			
73F3	29683	70	112	LOAD HL, (28729)	LOAD DDY INTO HL	
73F4	29684	ED	237			
73F5	29685	5B	91			
73F6	29686	3B	59			
73F7	29687	70	112	LOAD DE, (28731)	LOAD DELTA X INTO DE	
73F8	29688	37	55	SCF		
73F9	29689	3F	63	CCF	CARRY FLAG TO ZERO	
73FA	29690	ED	237			
73FB	29691	52	82	SBC HL, DE	SUBTRACT DELTA X FROM DDY	
73FC	29692	22	34			
73FD	29693	35	53			
73FE	29694	70	112	LOAD (28725), HL	SAVE F TO 28725 - 28726	
73FF	29695	CD	205		****** LOOP RETURNS TO HERE ******	PLOT POINT & ITERATIVE LOOP RETURN POINT
7400	29696	CE	206			
7401	29697	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
7402	29698	21	33			

7403	29699	0	0			
7404	29700	0	0	LOAD HL, 0	HL TO ZERO	
7405	29701	ED	237			
7406	29702	5B	91			
7407	29703	35	53			
7408	29704	70	112	LOAD DE, (28725)	LOAD F INTO DE	IF F > 0 THEN DO NEXT TWO OPERATIONS, ELSE SKIP
7409	29705	37	55	SCF		
740A	29706	3F	63	CCF	CARRY FLAG TO ZERO	
740B	29707	ED	237			
740C	29708	52	82	SBC HL, DE	SUBTRACT F FROM ZERO. SIGN NEGATIVE IF F > 0	
740D	29709	F2	242			
740E	29710	25	37			
740F	29711	74	116	JP P, 29733	JUMP Y-DECREMENT IF F <= 0 (SIGN POSITIVE)	
7410	29712	2A	42			
7411	29713	4	4			
7412	29714	70	112	LOAD HL, (28676)	LOAD PIXEL Y INTO HL	PIXEL Y = PIXEL Y - 1
7413	29715	2B	43	DEC HL	PIXEL Y = PIXEL Y - 1	
7414	29716	22	34			
7415	29717	4	4			
7416	29718	70	112	LOAD (28676), HL	SAVE UPDATED PIXEL Y VALUE	
7417	29719	2A	42			
7418	29720	35	53			
7419	29721	70	112	LOAD HL, (28725)	LOAD F INTO HL	
741A	29722	ED	237			
741B	29723	5B	91			
741C	29724	37	55			
741D	29725	70	112	LOAD DE, (28727)	LOAD DDX INTO DE	F = F - DDX
741E	29726	37	55	SCF		
741F	29727	3F	63	CCF	CARRY FLAG TO ZERO	
7420	29728	ED	237			
7421	29729	52	82	SBC HL, DE	SUBTRACT DDX FROM F	
7422	29730	22	34			
7423	29731	35	53			
7424	29732	70	112	LOAD (28725),HL	SAVE F TO 28725 - 28726	
7425	29733	2A	42			
7426	29734	35	53			
7427	29735	70	112	LOAD HL, (28725)	LOAD F INTO HL.	
7428	29736	ED	237			
7429	29737	5B	91			
742A	29738	39	57			F = F + DDY
742B	29739	70	112	LOAD DE, (28729)	LOAD DDY INTO DE	
742C	29740	19	25	ADD HL, DE	ADD DDY TO F	
742D	29741	22	34			
742E	29742	35	53			
742F	29743	70	112	LOAD (28725), HL	SAVE F TO 28725 - 28726	
7430	29744	ED	237			
7431	29745	5B	91			
7432	29746	2	2			
7433	29747	70	112	LOAD DE, (28674)	LOAD PIXEL X INTO DE	
7434	29748	13	19	INC DE	PIXEL X = PIXEL X + 1	PIXEL X = PIXEL X + 1
7435	29749	ED	237			
7436	29750	53	83			
7437	29751	2	2			
7438	29752	70	112	LOAD (28674), DE	SAVE UPDATED PIXEL X VALUE	
7439	29753	2A	42			
743A	29754	A	10			
743B	29755	70	112	LOAD HL, (28682)	LOAD X2 INTO HL	
743C	29756	37	55	SCF		
743D	29757	3F	63	CCF	CARRY FLAG TO ZERO	IF PIXEL X <= X2 THEN DO NEXT ITERATION, ELSE END
743E	29758	ED	237			
743F	29759	52	82	SBC HL, DE	SUBTRACT PIXEL X FROM X2. SIGN NEGATIVE IF PIXEL X > X2	
7440	29760	F2	242			
7441	29761	FF	255			
7442	29762	73	115	JP P, 29695	JUMP IF PIXEL X <= X2 (SIGN POSITIVE)	
7443	29763	1B	27	DEC DE		
7444	29764	ED	237			
7445	29765	53	83			PIXEL X = X2
7446	29766	2	2			
7447	29767	70	112	LOAD (28674), DE	SAVE UPDATED PIXEL X VALUE	
7448	29768	C9	201	RET		END OCTANT
7449	29769	2A	42		****** SECOND OCTANT STARTS HERE ********	
744A	29770	37	55			
744B	29771	70	112	LOAD HL, (28727)	LOAD DDX INTO HL	
744C	29772	ED	237			
744D	29773	5B	91			
744E	29774	3D	61			
744F	29775	70	112	LOAD DE, (28733)	LOAD DELTA Y INTO DE	F = DDX - DELTA Y
7450	29776	37	55	SCF		
7451	29777	3F	63	CCF	CARRY FLAG TO ZERO	
7452	29778	ED	237			
7453	29779	52	82	SBC HL, DE	SUBTRACT DELTA Y FROM DDX	
7454	29780	22	34			
7455	29781	35	53			
7456	29782	70	112	LOAD (28725), HL	SAVE F TO 28725 - 28726	
7457	29783	CD	205		****** LOOP RETURNS TO HERE ******	PLOT POINT & ITERATIVE LOOP RETURN POINT
7458	29784	CE	206			
7459	29785	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
745A	29786	21	33			
745B	29787	0	0			
745C	29788	0	0	LOAD HL, 0	HL TO ZERO	
745D	29789	ED	237			
745E	29790	5B	91			
745F	29791	35	53			
7460	29792	70	112	LOAD DE, (28725)	LOAD F INTO DE	IF F > 0 THEN DO NEXT TWO OPERATIONS, ELSE SKIP
7461	29793	37	55	SCF		
7462	29794	3F	63	CCF	CARRY FLAG TO ZERO	
7463	29795	ED	237			
7464	29796	52	82	SBC HL, DE	SUBTRACT F FROM ZERO. SIGN NEGATIVE IF F > 0	
7465	29797	F2	242			
7466	29798	7D	125			
7467	29799	74	116	JP P, 29821	JUMP IF F <= 0 (SIGN POSITIVE)	
7468	29800	2A	42			
7469	29801	2	2			

746A	29802	70	112	LOAD HL, (28674)	LOAD PIXEL X INTO HL	PIXEL X = PIXEL X + 1
746B	29803	23	35	INC HL	PIXEL X = PIXEL X + 1	
746C	29804	22	34			
746D	29805	2	2			
746E	29806	70	112	LOAD (28674), HL	SAVE UPDATED PIXEL X VALUE	
746F	29807	2A	42			F = F - DDY
7470	29808	35	53			
7471	29809	70	112	LOAD HL, 28725	LOAD F INTO HL	
7472	29810	ED	237			
7473	29811	5B	91			
7474	29812	39	57			
7475	29813	70	112	LOAD DE, 28729	LOAD DDY INTO DE	
7476	29814	37	55	SCF		
7477	29815	3F	63	CCF	CARRY FLAG TO ZERO	
7478	29816	ED	237			
7479	29817	52	82	SBC HL, DE	SUBTRACT DDY FROM F	
747A	29818	22	34			
747B	29819	35	53			
747C	29820	70	112	LOAD (28725), HL	SAVE F TO 28725 - 28726	
747D	29821	2A	42		****** X-INCREMENT JUMP GOES TO HERE ******	F = F + DDX
747E	29822	35	53			
747F	29823	70	112	LOAD HL, 28725	LOAD F INTO HL	
7480	29824	ED	237			
7481	29825	5B	91			
7482	29826	37	55			
7483	29827	70	112	LOAD DE, 28727	LOAD DDX INTO DE	
7484	29828	19	25	ADD HL, DE	ADD DDX TO F	
7485	29829	22	34			
7486	29830	35	53			
7487	29831	70	112	LOAD (28725), HL	SAVE F TO 28725 - 28726	
7488	29832	2A	42			PIXEL Y = PIXEL Y - 1
7489	29833	4	4			
748A	29834	70	112	LOAD HL, 28676	LOAD PIXEL Y INTO HL	
748B	29835	2B	43	DEC HL	PIXEL Y = PIXEL Y - 1	
748C	29836	22	34			
748D	29837	4	4			
748E	29838	70	112	LOAD (28676), HL	SAVE UPDATED PIXEL Y VALUE	
748F	29839	ED	237			IF PIXEL Y <= Y2 THEN DO NEXT ITERATION, ELSE END
7490	29840	5B	91			
7491	29841	C	12			
7492	29842	70	112	LOAD DE, (28684)	LOAD Y2 INTO DE	
7493	29843	37	55	SCF		
7494	29844	3F	63	CCF	CARRY FLAG TO ZERO	
7495	29845	ED	237			
7496	29846	52	82	SBC HL, DE	SUBTRACT Y2 FROM PIXEL Y. SIGN NEGATIVE IF PIXEL Y2 > PIXEL Y	
7497	29847	F2	242			
7498	29848	57	87			
7499	29849	74	116	JP P, 29783	JUMP IF Y2 <= PIXEL Y (SIGN POSITIVE)	
749A	29850	ED	237			PIXEL Y = Y2
749B	29851	53	83			
749C	29852	4	4			
749D	29853	70	112	LOAD (28676), DE	SAVE Y2 VALUE TO PIXEL Y	
749E	29854	C9	201	RET		END OCTANT
749F	29855	ED	237	DRAW LINE (X1, Y1) - (X2, Y2)		MOVE PIXEL POINTER TO START OF LINE PIXEL X = X1, PIXEL Y = Y1
74A0	29856	5B	91			
74A1	29857	6	6			
74A2	29858	70	112	LOAD DE,(28678)	LOAD X1 INTO DE	
74A3	29859	ED	237			
74A4	29860	53	83			
74A5	29861	2	2			
74A6	29862	70	112	LOAD (0x28674),DE	SET PIXEL X VALUE TO X1	
74A7	29863	ED	237			
74A8	29864	4B	75			
74A9	29865	8	8			
74AA	29866	70	112	LOAD BC,(28680)	LOAD Y1 INTO BC	
74AB	29867	ED	237			
74AC	29868	43	67			
74AD	29869	4	4			
74AE	29870	70	112	LOAD (28676),BC	SET PIXEL Y VALUE TO Y1	
74AF	29871	2A	42			IF X1 <= X2 THEN SELECT EITHER QUADRANT A OR D, ELSE SELECT EITHER QUADRANT B OR C.
74B0	29872	A	10			
74B1	29873	70	112	LOAD HL,(28682)	LOAD X2 INTO HL	
74B2	29874	37	55	SCF		
74B3	29875	3F	63	CCF	CARRY FLAG TO ZERO	
74B4	29876	ED	237			
74B5	29877	52	82	SBC HL, DE	SUBTRACT X1 FROM X2	
74B6	29878	F2	242			
74B7	29879	C6	198			
74B8	29880	74	116	JP P, 29894	IF X1 <= X2 (SIGN POSITIVE) THEN JUMP	
74B9	29881	2A	42			IF Y1 <= Y2 THEN GO TO QUADRANT B, ELSE GO TO QUADRANT C
74BA	29882	C	12			
74BB	29883	70	112	LOAD HL,(28684)	LOAD Y2 INTO HL	
74BC	29884	37	55	SCF		
74BD	29885	3F	63	CCF	CARRY FLAG TO ZERO	
74BE	29886	ED	237			
74BF	29887	42	66	SBC HL, BC	SUBTRACT Y1 FROM Y2	
74C0	29888	F2	242			
74C1	29889	EF	239			
74C2	29890	71	113	JP P, 29167	IF Y1 <= Y2 (SIGN POSITIVE) THEN JUMP TO QUADRANT B.	
74C3	29891	C3	195			
74C4	29892	D5	213			
74C5	29893	72	114	JP, 29379	JUMP TO QUADRANT C	
74C6	29894	2A	42			IF Y1 <= Y2 THEN GO TO QUADRANT A, ELSE GO TO QUADRANT D
74C7	29895	C	12			
74C8	29896	70	112	LOAD HL,(28684)	LOAD Y2 INTO HL	
74C9	29897	37	55	SCF		
74CA	29898	3F	63	CCF	CARRY FLAG TO ZERO	
74CB	29899	ED	237			
74CC	29900	42	66	SBC HL, BC	SUBTRACT Y1 FROM Y2	
74CD	29901	F2	242			
74CE	29902	7	7			
74CF	29903	71	113	JP P, 28935	IF Y1 <= Y2 (SIGN POSITIVE) THEN JUMP TO QUADRANT A	
74D0	29904	C3	195			

74D1	29905	B9	185			
74D2	29906	73	115	JP, 29625	JUMP TO QUADRANT D	
74D3	29907	ED	237	DRAW LINE (- (X2, Y2))		
74D4	29908	5B	91			
74D5	29909	6	6			
74D6	29910	70	112	LOAD DE,(28678)	LOAD X1 INTO DE	
74D7	29911	ED	237			
74D8	29912	53	83			
74D9	29913	2	2			
74DA	29914	70	112	LOAD (0x28674),DE	SET PIXEL X VALUE TO X1	
74DB	29915	ED	237			
74DC	29916	4B	75			
74DD	29917	8	8			
74DE	29918	70	112	LOAD BC,(28680)	LOAD Y1 INTO BC	
74DF	29919	ED	237			
74E0	29920	43	67			
74E1	29921	4	4			
74E2	29922	70	112	LOAD (28676),BC	SET PIXEL Y VALUE TO Y1	
74E3	29923	2A	42			
74E4	29924	A	10			
74E5	29925	70	112	LOAD HL,(28682)	LOAD X2 INTO HL	
74E6	29926	37	55	SCF		
74E7	29927	3F	63	CCF	CARRY FLAG TO ZERO	
74E8	29928	ED	237			
74E9	29929	52	82	SBC HL, DE	SUBTRACT X1 FROM X2	
74EA	29930	F2	242			
74EB	29931	3	3			
74EC	29932	75	117	JP P, 29955	IF X1 <= X2 (SIGN POSITIVE) THEN JUMP	
74ED	29933	2A	42			
74EE	29934	C	12			
74EF	29935	70	112	LOAD HL,(28684)	LOAD Y2 INTO HL	
74F0	29936	37	55	SCF		
74F1	29937	3F	63	CCF	CARRY FLAG TO ZERO	
74F2	29938	ED	237			
74F3	29939	42	66	SBC HL, BC	SUBTRACT Y1 FROM Y2	
74F4	29940	F2	242			
74F5	29941	FD	253			
74F6	29942	74	116	JP P, 29949	IF Y1 <= Y2 (SIGN POSITIVE) THEN JUMP TO QUADRANT B CALL.	
74F7	29943	CD	205			
74F8	29944	D5	213			
74F9	29945	72	114	CALL, 29379	QUADRANT C	
74FA	29946	C3	195			
74FB	29947	16	22			
74FC	29948	75	117	JP, 29974		
74FD	29949	CD	205			
74FE	29950	EF	239			
74FF	29951	71	113	CALL, 29167	QUADRANT B	
7500	29952	C3	195			
7501	29953	16	22			
7502	29954	75	117	JP, 29974		
7503	29955	2A	42			
7504	29956	C	12			
7505	29957	70	112	LOAD HL,(28684)	LOAD Y2 INTO HL	
7506	29958	37	55	SCF		
7507	29959	3F	63	CCF	CARRY FLAG TO ZERO	
7508	29960	ED	237			
7509	29961	42	66	SBC HL, BC	SUBTRACT Y1 FROM Y2	
750A	29962	F2	242			
750B	29963	13	19			
750C	29964	75	117	JP P, 29971	IF Y1 <= Y2 (SIGN POSITIVE) THEN JUMP TO QUADRANT A CALL	
750D	29965	CD	205			
750E	29966	B9	185			
750F	29967	73	115	CALL, 29625	QUADRANT D	
7510	29968	C3	195			
7511	29969	16	22			
7512	29970	75	117	JP, 29974		
7513	29971	CD	205			
7514	29972	7	7			
7515	29973	71	113	CALL, 28935	QUADRANT A	
7516	29974	2A	42			
7517	29975	2	2			
7518	29976	70	112	LOAD HL, (28674)	LOAD PIXEL X INTO HL	
7519	29977	22	34			
751A	29978	6	6			
751B	29979	70	112	LOAD (28678), HL	X1 = PIXEL X	
751C	29980	2A	42			
751D	29981	4	4			
751E	29982	70	112	LOAD HL, (28676)	LOAD PIXEL Y INTO HL	
751F	29983	22	34			
7520	29984	8	8			
7521	29985	70	112	LOAD (28680), HL	Y1 = PIXEL Y	
7522	29986	C9	201	RET		
7523	29987	0	0			
7524	29988	0	0			
7525	29989	0	0			
7526	29990	0	0			
7527	29991	0	0			
7528	29992	0	0			
7529	29993	0	0			
752A	29994	0	0			
752B	29995	0	0			
752C	29996	0	0			
752D	29997	0	0			
752E	29998	0	0			
752F	29999	0	0			
7530	30000	0	0			
7531	30001	0	0			
7532	30002	0	0			
7533	30003	0	0			
7534	30004	0	0			
7535	30005	0	0			
7536	30006	0	0			
7537	30007	0	0			

MOVE PIXEL POINTER TO START OF LINE PIXEL X = X1, PIXEL Y = Y1

IF X1 <= X2 THEN SELECT EITHER QUADRANT A OR D, ELSE SELECT EITHER QUADRANT B OR C.

IF Y1 <= Y2 THEN CALL QUADRANT B, ELSE CALL QUADRANT C

IF Y1<= Y2 THEN CALL QUADRANT A, ELSE CALL QUADRANT D

COORDINATES FOR START OF NEXT LINE SET TO END POINT COORDINATES OF LINE JUST PLOTTED

RETURN TO BASIC

7538	30008	0	0			
7539	30009	0	0			
753A	30010	0	0			
753B	30011	0	0			
753C	30012	0	0			
753D	30013	0	0			
753E	30014	0	0			
753F	30015	0	0			
7540	30016	0	0			
7541	30017	0	0			
7542	30018	0	0			
7543	30019	CD	205	DRAW CIRCLE ROUTINE - IMPLEMENTS BRESENHAM'S MID-POINT CIRCLE ALGORITHM		
7544	30020	7F	127			
7545	30021	A	10	CALL, 2687		GET RADIUS VALUE FROM BASIC USR(RADIUS) AND LOAD INTO DE
7546	30022	22	34			
7547	30023	25	37			
7548	30024	70	112	LOAD (28709), HL		
7549	30025	5D	93	LOAD E, L		
754A	30026	54	84	LOAD D, H	SHIFT RADIUS INTO DE	
754B	30027	2A	42			
754C	30028	2	2			SAVE CENTER X
754D	30029	70	112	LOAD HL, (28674)	LOAD PIXEL X INTO HL	
754E	30030	22	34			
754F	30031	31	49			
7550	30032	70	112	LOAD (28721), HL	SET CENTER X TO PIXEL X	
7551	30033	2A	42			
7552	30034	4	4			SAVE CENTER Y
7553	30035	70	112	LOAD HL, (28676)	LOAD PIXEL Y INTO HL	
7554	30036	22	34			
7555	30037	33	51			
7556	30038	70	112	LOAD (28723), HL	SET CENTER Y TO PIXEL Y	
7557	30039	21	33			
7558	30040	0	0			OCTANT X = 0
7559	30041	0	0	LOAD HL, 0x0000	HL TO ZERO	
755A	30042	22	34			
755B	30043	27	39			
755C	30044	70	112	LOAD (28711), HL	SET OCTANT X TO ZERO	
755D	30045	22	34			DDX = 0
755E	30046	2B	43			
755F	30047	70	112	LOAD (28715), HL	SET DDX TO ZERO	
7560	30048	ED	237			OCTANT Y = RADIUS
7561	30049	53	83			
7562	30050	29	41			
7563	30051	70	112	LOAD (28713), DE	SET OCTANT Y TO RADIUS	
7564	30052	CB	203			
7565	30053	23	35	SLA, E	MULTIPLY RADIUS BY TWO, STEP 1 OF 2	
7566	30054	CB	203			
7567	30055	12	18	RL, D	MULTIPLY RADIUS BY TWO, STEP 2 OF 2	
7568	30056	37	55	SCF		
7569	30057	3F	63	CCF	CARRY FLAG TO ZERO	DDY = -2 * RADIUS
756A	30058	ED	237			
756B	30059	52	82	SBC HL, DE	SUBTRACT 2 * RADIUS FROM ZERO	
756C	30060	22	34			
756D	30061	2D	45			
756E	30062	70	112	LOAD (28717), HL	SET DDY TO -2 * RADIUS	
756F	30063	21	33			
7570	30064	1	1			
7571	30065	0	0	LOAD HL, 0x0001	HL TO ONE	
7572	30066	ED	237			
7573	30067	5B	91			
7574	30068	25	37			
7575	30069	70	112	LOAD DE, (28709)	LOAD RADIUS INTO DE	F = 1 - RADIUS
7576	30070	37	55	SCF		
7577	30071	3F	63	CCF	CARRY FLAG TO ZERO	
7578	30072	ED	237			
7579	30073	52	82	SBC HL, DE	SUBTRACT RADIUS FROM ONE	
757A	30074	22	34			
757B	30075	2F	47			
757C	30076	70	112	LOAD (28719), HL	SET F TO 1 - RADIUS	
757D	30077	CD	205		CALL EITHER THE OCTANT PLOTTING ROUTINE OR THE QUADRANT PLOTING ROUTINE.	DRAW TO SCREEN ITERATIONS START HERE
757E	30078	D8	216		THESE ADDRESS VALUE LOCATIONS ARE MODIFIED TO THE VALUES REQUIRED TO CALL THE	
757F	30079	75	117	CALL, 0xNNNN	QUADRANT PLOTTING ROUTINE BY THE DRAW SOLID CIRCLE ROUTINE WHEN RUN.	
7580	30080	ED	237			
7581	30081	5B	91			
7582	30082	2F	47			
7583	30083	70	112	LOAD DE, (28719)	LOAD F INTO DE	
7584	30084	21	33			
7585	30085	0	0			IF F < 0 THEN SKIP THE NEXT THREE OPERATIONS
7586	30086	0	0	LOAD HL, 0x0000	HL TO ZERO	
7587	30087	37	55	SCF		
7588	30088	3F	63	CCF	CARRY FLAG TO ZERO	
7589	30089	ED	237			
758A	30090	5A	90	ADC HL, DE	ADD F TO ZERO WITH CARRY. SIGN FLAG NEGATIVE IF F < 0	
758B	30091	FA	250			
758C	30092	A5	165			
758D	30093	75	117	JP M, 30117		
758E	30094	2A	42			
758F	30095	29	41			
7590	30096	70	112	LOAD HL, (28713)	LOAD OCTANT Y INTO HL	OCTANT Y = OCTANT Y - 1
7591	30097	2B	43	DEC HL	OCTANT Y = OCTANT Y - 1	
7592	30098	22	34			
7593	30099	29	41			
7594	30100	70	112	LOAD (28713), HL	SAVE OCTANT Y - 1	
7595	30101	2A	42			
7596	30102	2D	45			
7597	30103	70	112	LOAD HL, (28717)	LOAD DDY INTO HL	DDY = DDY + 2
7598	30104	23	35	INC HL		
7599	30105	23	35	INC HL	DDY = DDY + 2	
759A	30106	22	34			
759B	30107	2D	45			
759C	30108	70	112	LOAD (28717), HL	SAVE DDY + 2	
759D	30109	ED	237			
759E	30110	5B	91			

759F	30111	2F	47			F = F + DDY
75A0	30112	70	112	LOAD DE, (28719)	LOAD F INTO DE	
75A1	30113	19	25	ADD HL, DE	ADD F TO DDY	
75A2	30114	22	34			
75A3	30115	2F	47			
75A4	30116	70	112	LOAD (28719), HL	SAVE F = F + DDY	
75A5	30117	2A	42			OCTANT X = OCTANT X + 1
75A6	30118	27	39			
75A7	30119	70	112	LOAD HL, (28711)	LOAD OCTANT X INTO HL	
75A8	30120	23	35	INC HL	OCTANT X = OCTANT X + 1	
75A9	30121	22	34			
75AA	30122	27	39			
75AB	30123	70	112	LOAD (28711), HL	SAVE OCTANT X + 1	
75AC	30124	2A	42			DDX = DDX + 2
75AD	30125	2B	43			
75AE	30126	70	112	LOAD HL, (28715)	LOAD DDX INTO HL	
75AF	30127	23	35	INC HL		
75B0	30128	23	35	INC HL	DDX = DDX + 2	
75B1	30129	22	34			
75B2	30130	2B	43			
75B3	30131	70	112	LOAD (28715), HL	SAVE DDX + 2	
75B4	30132	ED	237			F = F + DDX + 1
75B5	30133	5B	91			
75B6	30134	2F	47			
75B7	30135	70	112	LOAD DE, (28719)	LOAD F INTO DE	
75B8	30136	19	25	ADD HL, DE	ADD F TO DDX. CARRY FLAG RESET	
75B9	30137	23	35	INC HL	F = F + DDX + 1	
75BA	30138	22	34			
75BB	30139	2F	47			
75BC	30140	70	112	LOAD (28719), HL	SAVE F = F + DDX + 1	
75BD	30141	ED	237			IF OCTANT X <= OCTANT Y THEN NEXT ITERATION
75BE	30142	5B	91			
75BF	30143	27	39			
75C0	30144	70	112	LOAD DE, (28713)	LOAD OCTANT X INTO DE	
75C1	30145	2A	42			
75C2	30146	29	41			
75C3	30147	70	112	LOAD HL, (28711)	LOAD OCTANT Y INTO HL	
75C4	30148	37	55	SCF		
75C5	30149	3F	63	CCF	CARRY FLAG TO ZERO	
75C6	30150	ED	237			
75C7	30151	52	82	SBC HL, DE	SUBTRACT OCTANT Y FROM OCTANT X. SIGN NEGATIVE IF OCTANT X < OCTANT Y	
75C8	30152	F2	242			
75C9	30153	7D	125			
75CA	30154	75	117	JP P, 30077	IF OCTANT X <= OCTANT Y THEN NEXT ITERATION	
75CB	30155	2A	42			ELSE RESET PIXEL X TO CENTER X AND PIXELY TO CENTER Y AND RETURN TO BASIC
75CC	30156	31	49			
75CD	30157	70	112	LOAD HL, (28721)	LOAD CENTER X INTO HL	
75CE	30158	22	34			
75CF	30159	2	2			
75D0	30160	70	112	LOAD (28674), HL	SET PIXEL X TO CENTER X	
75D1	30161	2A	42			
75D2	30162	33	51			
75D3	30163	70	112	LOAD HL, (28723)	LOAD CENTER Y INTO HL	
75D4	30164	22	34			
75D5	30165	4	4			
75D6	30166	70	112	LOAD (28676), HL	SET PIXEL Y TO CENTER Y	
75D7	30167	C9	201	RET	RETURN TO BASIC	
75D8	30168	21	33	OCTANT PLOTTING ROUTINE FOR CIRCLE AND ARC		FETCH "SEGMENT ENABLE" VARIABLE INTO E
75D9	30169	E	14			
75DA	30170	70	112	LOAD HL, 28686		
75DB	30171	5E	94	LOAD E, (HL)		
75DC	30172	CB	203			SET PIXEL CX + OX, CY + OY IF BIT 1 OF SEGMENT = 1
75DD	30173	4B	75	BIT 1, E		
75DE	30174	28	40			
75DF	30175	19	25	JR Z, 25	SKIP	
75E0	30176	ED	237			
75E1	30177	5B	91			
75E2	30178	27	39			
75E3	30179	70	112	LOAD DE, (28711)	LOAD OCTANT X INTO DE	
75E4	30180	2A	42			
75E5	30181	31	49			
75E6	30182	70	112	LOAD HL, (28721)	LOAD CENTER X INTO HL	
75E7	30183	19	25	ADD HL, DE	ADD OCTANT X TO CENTER X	
75E8	30184	22	34			
75E9	30185	2	2			
75EA	30186	70	112	LOAD (28674), HL	SET PIXEL X TO CENTER X + OCTANT X	
75EB	30187	ED	237			
75EC	30188	5B	91			
75ED	30189	29	41			
75EE	30190	70	112	LOAD DE, (28713)	LOAD OCTANT Y INTO DE	
75EF	30191	2A	42			
75F0	30192	33	51			
75F1	30193	70	112	LOAD HL, (28723)	LOAD CENTER Y INTO HL	
75F2	30194	19	25	ADD HL, DE	ADD OCTANT Y TO CENTER Y	
75F3	30195	22	34			
75F4	30196	4	4			
75F5	30197	70	112	LOAD (28676), HL	SET PIXEL Y TO CENTER Y + OCTANT Y	
75F6	30198	CD	205			
75F7	30199	CE	206			
75F8	30200	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
75F9	30201	21	33			FETCH "SEGMENT ENABLE" VARIABLE INTO E
75FA	30202	E	14			
75FB	30203	70	112	LOAD HL, 28686		
75FC	30204	5E	94	LOAD E, (HL)		
75FD	30205	CB	203			
75FE	30206	73	115	BIT 6, E		
75FF	30207	28	40			
7600	30208	1A	26	JR Z, 26	SKIP	
7601	30209	ED	237			
7602	30210	5B	91			
7603	30211	27	39			
7604	30212	70	112	LOAD DE, (28711)	LOAD OCTANT X INTO DE	
7605	30213	2A	42			

7606	30214	31	49			
7607	30215	70	112	LOAD HL, (28721)	LOAD CENTER X INTO HL	
7608	30216	19	25	ADD HL, DE	ADD OCTANT X TO CENTER X	
7609	30217	22	34			
760A	30218	2	2			
760B	30219	70	112	LOAD (28674), HL	SET PIXEL X TO CENTER X + OCTANT X	SET PIXEL CX + OX, CY - OY IF BIT 6 OF SEGMENT = 1
760C	30220	ED	237			
760D	30221	5B	91			
760E	30222	29	41			
760F	30223	70	112	LOAD DE, (28713)	LOAD OCTANT Y INTO DE	
7610	30224	2A	42			
7611	30225	33	51			
7612	30226	70	112	LOAD HL, (28723)	LOAD CENTER Y INTO HL	
7613	30227	ED	237			
7614	30228	52	82	SBC HL, DE	SUBTRACT OCTANT Y FROM CENTER Y. CARRY RESET AT LAST ADD	
7615	30229	22	34			
7616	30230	4	4			
7617	30231	70	112	LOAD (28676), HL	SET PIXEL Y TO CENTER Y - OCTANT Y	
7618	30232	CD	205			
7619	30233	CE	206			
761A	30234	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
761B	30235	21	33			
761C	30236	E	14			FETCH "SEGMENT ENABLE" VARIABLE INTO E
761D	30237	70	112	LOAD HL, 28686		
761E	30238	5E	94	LOAD E, (HL)		
761F	30239	CB	203			
7620	30240	7B	123	BIT 7, E		
7621	30241	28	40			
7622	30242	1A	26	JR Z, 26	SKIP	
7623	30243	ED	237			
7624	30244	5B	91			
7625	30245	29	41			
7626	30246	70	112	LOAD DE, (28713)	LOAD OCTANT Y INTO DE	
7627	30247	2A	42			
7628	30248	31	49			
7629	30249	70	112	LOAD HL, (28721)	LOAD CENTER X INTO HL	
762A	30250	19	25	ADD HL, DE	ADD OCTANT Y TO CENTER X	
762B	30251	22	34			
762C	30252	2	2			
762D	30253	70	112	LOAD (28674), HL	SET PIXEL X TO CENTER X + OCTANT X	SET PIXEL CX + OY, CY - OX IF BIT 7 OF SEGMENT = 1
762E	30254	ED	237			
762F	30255	5B	91			
7630	30256	27	39			
7631	30257	70	112	LOAD DE, (28711)	LOAD OCTANT X INTO DE	
7632	30258	2A	42			
7633	30259	33	51			
7634	30260	70	112	LOAD HL, (28723)	LOAD CENTER Y INTO HL	
7635	30261	ED	237			
7636	30262	52	82	SBC HL, DE	SUBTRACT OCTANT X FROM CENTER Y. CARRY RESET AT LAST ADD	
7637	30263	22	34			
7638	30264	4	4			
7639	30265	70	112	LOAD (28676), HL	SET PIXEL Y TO CENTER Y - OCTANT X	
763A	30266	CD	205			
763B	30267	CE	206			
763C	30268	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
763D	30269	21	33			
763E	30270	E	14			FETCH "SEGMENT ENABLE" VARIABLE INTO E
763F	30271	70	112	LOAD HL, 28686		
7640	30272	5E	94	LOAD E, (HL)		
7641	30273	CB	203			
7642	30274	43	67	BIT 0, E		
7643	30275	28	40			
7644	30276	19	25	JR Z, 25	SKIP	
7645	30277	ED	237			
7646	30278	5B	91			
7647	30279	29	41			
7648	30280	70	112	LOAD DE, (28713)	LOAD OCTANT Y INTO DE	
7649	30281	2A	42			
764A	30282	31	49			
764B	30283	70	112	LOAD HL, (28721)	LOAD CENTER X INTO HL	
764C	30284	19	25	ADD HL, DE	ADD OCTANT Y TO CENTER X	
764D	30285	22	34			
764E	30286	2	2			
764F	30287	70	112	LOAD (28674), HL	SET PIXEL X TO CENTER X + OCTANT Y	SET PIXEL CX + OY, CY + OX IF BIT 0 OF SEGMENT = 1
7650	30288	ED	237			
7651	30289	5B	91			
7652	30290	27	39			
7653	30291	70	112	LOAD DE, (28711)	LOAD OCTANT X INTO DE	
7654	30292	2A	42			
7655	30293	33	51			
7656	30294	70	112	LOAD HL, (28723)	LOAD CENTER Y INTO HL	
7657	30295	19	25	ADD HL, DE	ADD OCTANT X TO CENTER Y	
7658	30296	22	34			
7659	30297	4	4			
765A	30298	70	112	LOAD (28676), HL	SET PIXEL Y TO CENTER Y + OCTANT X	
765B	30299	CD	205			
765C	30300	CE	206			
765D	30301	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
765E	30302	21	33			
765F	30303	E	14			FETCH "SEGMENT ENABLE" VARIABLE INTO E
7660	30304	70	112	LOAD HL, 28686		
7661	30305	5E	94	LOAD E, (HL)		
7662	30306	CB	203			
7663	30307	53	83	BIT 2, E		
7664	30308	28	40			
7665	30309	1A	26	JR Z, 26	SKIP	
7666	30310	ED	237			
7667	30311	5B	91			
7668	30312	29	41			
7669	30313	70	112	LOAD DE, (28713)	LOAD OCTANT Y INTO DE	
766A	30314	2A	42			
766B	30315	33	51			
766C	30316	70	112	LOAD HL, (28723)	LOAD CENTER Y INTO HL	

766D	30317	19	25	ADD HL, DE	ADD OCTANT Y TO CENTER Y	
766E	30318	22	34			
766F	30319	4	4			
7670	30320	70	112	LOAD (28676), HL	SET PIXEL Y TO CENTER Y + OCTANT Y	SET PIXEL CX - OX, CY + OY IF BIT 2 OF SEGMENT = 1
7671	30321	ED	237			
7672	30322	5B	91			
7673	30323	27	39			
7674	30324	70	112	LOAD DE, (28711)	LOAD OCTANT X INTO DE	
7675	30325	2A	42			
7676	30326	31	49			
7677	30327	70	112	LOAD HL, (28721)	LOAD CENTER X INTO HL	
7678	30328	ED	237			
7679	30329	52	82	SBC HL, DE	SUBTRACT OCTANT X FROM CENTER X. CARRY RESET AT LAST ADD	
767A	30330	22	34			
767B	30331	2	2			
767C	30332	70	112	LOAD (28674), HL	SET PIXEL X TO CENTER X - OCTANT X	
767D	30333	CD	205			
767E	30334	CE	206			
767F	30335	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
7680	30336	21	33			
7681	30337	E	14			FETCH "SEGMENT ENABLE" VARIABLE INTO E
7682	30338	70	112	LOAD HL, 28686		
7683	30339	5E	94	LOAD E, (HL)		
7684	30340	CB	203			
7685	30341	5B	91	BIT 3, E		
7686	30342	28	40			
7687	30343	1A	26	JR Z, 26	SKIP	
7688	30344	ED	237			
7689	30345	5B	91			
768A	30346	27	39			
768B	30347	70	112	LOAD DE, (28711)	LOAD OCTANT X INTO DE	
768C	30348	2A	42			
768D	30349	33	51			
768E	30350	70	112	LOAD HL, (28723)	LOAD CENTER Y INTO HL	
768F	30351	19	25	ADD HL, DE	ADD OCTANT X TO CENTER Y	
7690	30352	22	34			
7691	30353	4	4			
7692	30354	70	112	LOAD (28676), HL	SET PIXEL Y TO CENTER Y + OCTANT X	SET PIXEL CX - OY, CY + OX IF BIT 3 OF SEGMENT = 1
7693	30355	ED	237			
7694	30356	5B	91			
7695	30357	29	41			
7696	30358	70	112	LOAD DE, (28713)	LOAD OCTANT Y INTO DE	
7697	30359	2A	42			
7698	30360	31	49			
7699	30361	70	112	LOAD HL, (28721)	LOAD CENTER X INTO HL	
769A	30362	ED	237			
769B	30363	52	82	SBC HL, DE	SUBTRACT OCTANT Y FROM CENTER X. CARRY RESET AT LAST ADD	
769C	30364	22	34			
769D	30365	2	2			
769E	30366	70	112	LOAD (28674), HL	SET PIXEL X TO CENTER X - OCTANT Y	
769F	30367	CD	205			
76A0	30368	CE	206			
76A1	30369	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
76A2	30370	21	33			
76A3	30371	E	14			FETCH "SEGMENT ENABLE" VARIABLE INTO E
76A4	30372	70	112	LOAD HL, 28686		
76A5	30373	5E	94	LOAD E, (HL)		
76A6	30374	CB	203			
76A7	30375	63	99	BIT 4, E		
76A8	30376	28	40			
76A9	30377	1F	31	JR Z, 31	SKIP	
76AA	30378	ED	237			
76AB	30379	5B	91			
76AC	30380	29	41			
76AD	30381	70	112	LOAD DE, (28713)	LOAD OCTANT Y INTO DE	
76AE	30382	2A	42			
76AF	30383	31	49			
76B0	30384	70	112	LOAD HL, (28721)	LOAD CENTER X INTO HL	
76B1	30385	37	55	SCF		
76B2	30386	3F	63	CCF	CARRY FLAG TO ZERO	
76B3	30387	ED	237			
76B4	30388	52	82	SBC HL, DE	SUBTRACT OCTANT Y FROM CENTER X	
76B5	30389	22	34			
76B6	30390	2	2			
76B7	30391	70	112	LOAD (28674), HL	SET PIXEL X TO CENTER X - OCTANT Y	SET PIXEL CX - OY, CY - OX IF BIT 4 OF SEGMENT = 1
76B8	30392	ED	237			
76B9	30393	5B	91			
76BA	30394	27	39			
76BB	30395	70	112	LOAD DE, (28711)	LOAD OCTANT X INTO DE	
76BC	30396	2A	42			
76BD	30397	33	51			
76BE	30398	70	112	LOAD HL, (28723)	LOAD CENTER Y INTO HL	
76BF	30399	37	55	SCF		
76C0	30400	3F	63	CCF	CARRY FLAG TO ZERO	
76C1	30401	ED	237			
76C2	30402	52	82	SBC HL, DE	SUBTRACT OCTANT X FROM CENTER Y	
76C3	30403	22	34			
76C4	30404	4	4			
76C5	30405	70	112	LOAD (28676), HL	SET PIXEL Y TO CENTER Y - OCTANT X	
76C6	30406	CD	205			
76C7	30407	CE	206			
76C8	30408	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
76C9	30409	21	33			
76CA	30410	E	14			FETCH "SEGMENT ENABLE" VARIABLE INTO E
76CB	30411	70	112	LOAD HL, 28686		
76CC	30412	5E	94	LOAD E, (HL)		
76CD	30413	CB	203			
76CE	30414	6B	107	BIT 5, E		
76CF	30415	28	40			
76D0	30416	1F	31	JR Z, 31	SKIP	
76D1	30417	ED	237			
76D2	30418	5B	91			
76D3	30419	27	39			

76D4	30420	70	112	LOAD DE, (28711)	LOAD OCTANT X INTO DE	SET PIXEL CX - OX, CY - OY IF BIT 5 OF SEGMENT = 1
76D5	30421	2A	42			
76D6	30422	31	49			
76D7	30423	70	112	LOAD HL, (28721)	LOAD CENTER X INTO HL	
76D8	30424	37	55	SCF		
76D9	30425	3F	63	CCF	CARRY FLAG TO ZERO	
76DA	30426	ED	237			
76DB	30427	52	82	SBC HL, DE	SUBTRACT OCTANT X FROM CENTER X	
76DC	30428	22	34			
76DD	30429	2	2			
76DE	30430	70	112	LOAD (28674), HL	SET PIXEL X TO CENTER X - OCTANT X	
76DF	30431	ED	237			
76E0	30432	5B	91			
76E1	30433	29	41			
76E2	30434	70	112	LOAD DE, (28713)	LOAD OCTANT Y INTO DE	
76E3	30435	2A	42			
76E4	30436	33	51			
76E5	30437	70	112	LOAD HL, (28723)	LOAD CENTER Y INTO HL	
76E6	30438	37	55	SCF		
76E7	30439	3F	63	CCF	CARRY FLAG TO ZERO	
76E8	30440	ED	237			
76E9	30441	52	82	SBC HL, DE	SUBTRACT OCTANT Y FROM CENTER Y	
76EA	30442	22	34			
76EB	30443	4	4			
76EC	30444	70	112	LOAD (28676), HL	SET PIXEL Y TO CENTER Y - OCTANT Y	
76ED	30445	CD	205			
76EE	30446	CE	206			
76EF	30447	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
76F0	30448	C9	201	RET	RETURN	END
76F1	30449	21	33	QUADRANT PLOTTING ROUTINE FOR SOLID CIRCLE		FETCH "SEGMENT ENABLE" VARIABLE INTO E
76F2	30450	E	14			
76F3	30451	70	112	LOAD HL, 28686		
76F4	30452	5E	94	LOAD E, (HL)		
76F5	30453	CB	203			DRAW LOWER RIGHT QUADRANT OF CIRCLE IF BIT 0 OF SEGMENT = 1
76F6	30454	43	67	BIT 0, E		
76F7	30455	28	40			
76F8	30456	5C	92	JR Z, 92	SKIP	
76F9	30457	ED	237			
76FA	30458	5B	91			
76FB	30459	27	39			
76FC	30460	70	112	LOAD DE, (28711)	LOAD OCTANT X INTO DE	
76FD	30461	2A	42			
76FE	30462	31	49			
76FF	30463	70	112	LOAD HL, (28721)	LOAD CENTER X INTO HL	
7700	30464	19	25	ADD HL, DE	ADD OCTANT X TO CENTER X	
7701	30465	22	34			
7702	30466	2	2			
7703	30467	70	112	LOAD (28674), HL	SET PIXEL X TO CENTER X + OCTANT X	
7704	30468	ED	237			
7705	30469	5B	91			
7706	30470	29	41			
7707	30471	70	112	LOAD DE, (28713)	LOAD OCTANT Y INTO DE	
7708	30472	2A	42			
7709	30473	33	51			
770A	30474	70	112	LOAD HL, (28723)	LOAD CENTER Y INTO HL	
770B	30475	19	25	ADD HL, DE	ADD OCTANT Y TO CENTER Y	
770C	30476	22	34			
770D	30477	4	4			
770E	30478	70	112	LOAD (28676), HL	SET PIXEL Y TO CENTER Y + OCTANT Y	
770F	30479	CD	205			
7710	30480	CE	206			
7711	30481	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
7712	30482	2A	42			PIXEL Y = PIXEL Y - 1
7713	30483	4	4			
7714	30484	70	112	LOAD HL, (28676)	LOAD PIXEL Y INTO HL	
7715	30485	2B	43	DEC HL	DECREMENT PIXEL Y	
7716	30486	22	34			
7717	30487	4	4			
7718	30488	70	112	LOAD (28676), HL	SAVE PIXEL Y	
7719	30489	ED	237			IF PIXEL Y => CENTER Y THEN DO NEXT ITERATION ELSE EXIT
771A	30490	5B	91			
771B	30491	33	51			
771C	30492	70	112	LOAD DE, (28723)	LOAD CENTER Y INTO DE	
771D	30493	37	55	SCF		
771E	30494	3F	63	CCF	CARRY FLAG TO ZERO	
771F	30495	ED	237			
7720	30496	52	82	SBC HL, DE	SUBTRACT CENTER Y FROM PIXEL Y. SIGN FLAG NEGATIVE IF PIXEL Y < CENTER Y	
7721	30497	FA	250			
7722	30498	27	39			
7723	30499	77	119	JP M, 30503	EXIT	
7724	30500	C3	195			
7725	30501	F	15			
7726	30502	77	119	JP, 30479	LOOP	
7727	30503	ED	237			
7728	30504	5B	91			
7729	30505	29	41			
772A	30506	70	112	LOAD DE, (28713)	LOAD OCTANT Y INTO DE	
772B	30507	2A	42			
772C	30508	31	49			
772D	30509	70	112	LOAD HL, (28721)	LOAD CENTER X INTO HL	
772E	30510	19	25	ADD HL, DE	ADD OCTANT Y TO CENTER X	
772F	30511	22	34			
7730	30512	2	2			
7731	30513	70	112	LOAD (28674), HL	SET PIXEL X TO CENTER X + OCTANT Y	
7732	30514	ED	237			
7733	30515	5B	91			
7734	30516	27	39			
7735	30517	70	112	LOAD DE, (28711)	LOAD OCTANT X INTO DE	
7736	30518	2A	42			
7737	30519	33	51			
7738	30520	70	112	LOAD HL, (28723)	LOAD CENTER Y INTO HL	
7739	30521	19	25	ADD HL, DE	ADD OCTANT X TO CENTER Y	
773A	30522	22	34			

773B	30523	4	4			
773C	30524	70	112	LOAD (28676), HL	SET PIXEL Y TO CENTER Y + OCTANT X	
773D	30525	CD	205			
773E	30526	CE	206			
773F	30527	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
7740	30528	2A	42			
7741	30529	4	4			
7742	30530	70	112	LOAD HL, (28676)	LOAD PIXEL Y INTO HL	
7743	30531	2B	43	DEC HL	DECREMENT PIXEL Y	PIXEL Y = PIXEL Y - 1
7744	30532	22	34			
7745	30533	4	4			
7746	30534	70	112	LOAD (28676), HL	SAVE PIXEL Y	
7747	30535	ED	237			
7748	30536	5B	91			
7749	30537	33	51			
774A	30538	70	112	LOAD DE, (28723)	LOAD CENTER Y INTO DE	
774B	30539	37	55	SCF		
774C	30540	3F	63	CCF	CARRY FLAG TO ZERO	
774D	30541	ED	237			IF PIXEL Y => CENTER Y THEN DO NEXT ITERATION ELSE EXIT
774E	30542	52	82	SBC HL, DE	SUBTRACT CENTER Y FROM PIXEL Y. SIGN FLAG NEGATIVE IF PIXEL Y < CENTER Y	
774F	30543	FA	250			
7750	30544	55	85			
7751	30545	77	119	JP M, 30549	EXIT	
7752	30546	C3	195			
7753	30547	3D	61			
7754	30548	77	119	JP, 30525	LOOP	
7755	30549	21	33			
7756	30550	E	14			FETCH "SEGMENT ENABLE" VARIABLE INTO E
7757	30551	70	112	LOAD HL, 28686		
7758	30552	5E	94	LOAD E, (HL)		
7759	30553	CB	203			
775A	30554	5B	91	BIT 3, E		
775B	30555	28	40			
775C	30556	60	96	JR Z, 96	SKIP	
775D	30557	ED	237			
775E	30558	5B	91			
775F	30559	27	39			
7760	30560	70	112	LOAD DE, (28711)	LOAD OCTANT X INTO DE	
7761	30561	2A	42			
7762	30562	31	49			
7763	30563	70	112	LOAD HL, (28721)	LOAD CENTER X INTO HL	
7764	30564	19	25	ADD HL, DE	ADD OCTANT X TO CENTER X	
7765	30565	22	34			
7766	30566	2	2			
7767	30567	70	112	LOAD (28674), HL	SET PIXEL X TO CENTER X + OCTANT X	
7768	30568	ED	237			
7769	30569	5B	91			
776A	30570	29	41			
776B	30571	70	112	LOAD DE, (28713)	LOAD OCTANT Y INTO DE	
776C	30572	2A	42			
776D	30573	33	51			
776E	30574	70	112	LOAD HL, (28723)	LOAD CENTER Y INTO HL	
776F	30575	ED	237			
7770	30576	52	82	SBC HL, DE	SUBTRACT OCTANT Y FROM CENTER Y. CARRY RESET AT LAST ADD	
7771	30577	22	34			
7772	30578	4	4			
7773	30579	70	112	LOAD (28676), HL	SET PIXEL Y TO CENTER Y - OCTANT Y	
7774	30580	CD	205			
7775	30581	CE	206			
7776	30582	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
7777	30583	ED	237			
7778	30584	5B	91			
7779	30585	4	4			
777A	30586	70	112	LOAD DE, (28676)	LOAD PIXEL Y INTO DE	
777B	30587	13	19	INC DE	INCREMENT PIXEL Y	PIXEL Y = PIXEL Y + 1
777C	30588	ED	237			
777D	30589	53	83			
777E	30590	4	4			
777F	30591	70	112	LOAD (28676), DE	SAVE PIXEL Y	
7780	30592	2A	42			
7781	30593	33	51			
7782	30594	70	112	LOAD HL, (28723)	LOAD CENTER Y INTO HL	
7783	30595	37	55	SCF		
7784	30596	3F	63	CCF	CARRY FLAG TO ZERO	
7785	30597	ED	237			IF CENTER Y => PIXEL Y THEN DO NEXT ITERATION ELSE EXIT
7786	30598	52	82	SBC HL, DE	SUBTRACT PIXEL Y FROM CENTER Y. SIGN FLAG NEGATIVE IF CENTER Y < PIXEL Y	
7787	30599	FA	250			
7788	30600	8D	141			
7789	30601	77	119	JP M, 30605	EXIT	
778A	30602	C3	195			
778B	30603	74	116			
778C	30604	77	119	JP, 30580	LOOP	
778D	30605	ED	237			
778E	30606	5B	91			
778F	30607	29	41			
7790	30608	70	112	LOAD DE, (28713)	LOAD OCTANT Y INTO DE	
7791	30609	2A	42			
7792	30610	31	49			
7793	30611	70	112	LOAD HL, (28721)	LOAD CENTER X INTO HL	
7794	30612	19	25	ADD HL, DE	ADD OCTANT Y TO CENTER X	
7795	30613	22	34			
7796	30614	2	2			
7797	30615	70	112	LOAD (28674), HL	SET PIXEL X TO CENTER X + OCTANT X	
7798	30616	ED	237			
7799	30617	5B	91			
779A	30618	27	39			
779B	30619	70	112	LOAD DE, (28711)	LOAD OCTANT X INTO DE	
779C	30620	2A	42			
779D	30621	33	51			
779E	30622	70	112	LOAD HL, (28723)	LOAD CENTER Y INTO HL	
779F	30623	ED	237			
77A0	30624	52	82	SBC HL, DE	SUBTRACT OCTANT X FROM CENTER Y. CARRY RESET AT LAST ADD	
77A1	30625	22	34			

77A2	30626	4	4			
77A3	30627	70	112	LOAD (28676), HL	SET PIXEL Y TO CENTER Y - OCTANT X	
77A4	30628	CD	205			
77A5	30629	CE	206			
77A6	30630	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
77A7	30631	ED	237			
77A8	30632	5B	91			
77A9	30633	4	4			
77AA	30634	70	112	LOAD DE, (28676)	LOAD PIXEL Y INTO DE	
77AB	30635	13	19	INC DE	INCREMENT PIXEL Y	
77AC	30636	ED	237			
77AD	30637	53	83			
77AE	30638	4	4			
77AF	30639	70	112	LOAD (28676), DE	SAVE PIXEL Y	
77B0	30640	2A	42			
77B1	30641	33	51			
77B2	30642	70	112	LOAD HL, (28723)	LOAD CENTER Y INTO HL	
77B3	30643	37	55	SCF		
77B4	30644	3F	63	CCF	CARRY FLAG TO ZERO	
77B5	30645	ED	237			
77B6	30646	52	82	SBC HL, DE	SUBTRACT PIXEL Y FROM CENTER Y. SIGN FLAG NEGATIVE IF CENTER Y < PIXEL Y	
77B7	30647	FA	250			
77B8	30648	BD	189			
77B9	30649	77	119	JP M, 30653	EXIT	
77BA	30650	C3	195			
77BB	30651	A4	164			
77BC	30652	77	119	JP, 30628	LOOP	
77BD	30653	21	33			
77BE	30654	E	14			
77BF	30655	70	112	LOAD HL, 28686		
77C0	30656	5E	94	LOAD E, (HL)		
77C1	30657	CB	203			
77C2	30658	4B	75	BIT 1, E		
77C3	30659	28	40			
77C4	30660	5E	94	JR Z, 94	SKIP	
77C5	30661	ED	237			
77C6	30662	5B	91			
77C7	30663	29	41			
77C8	30664	70	112	LOAD DE, (28713)	LOAD OCTANT Y INTO DE	
77C9	30665	2A	42			
77CA	30666	33	51			
77CB	30667	70	112	LOAD HL, (28723)	LOAD CENTER Y INTO HL	
77CC	30668	19	25	ADD HL, DE	ADD OCTANT Y TO CENTER Y	
77CD	30669	22	34			
77CE	30670	4	4			
77CF	30671	70	112	LOAD (28676), HL	SET PIXEL Y TO CENTER Y + OCTANT Y	
77D0	30672	ED	237			
77D1	30673	5B	91			
77D2	30674	27	39			
77D3	30675	70	112	LOAD DE, (28711)	LOAD OCTANT X INTO DE	
77D4	30676	2A	42			
77D5	30677	31	49			
77D6	30678	70	112	LOAD HL, (28721)	LOAD CENTER X INTO HL	
77D7	30679	ED	237			
77D8	30680	52	82	SBC HL, DE	SUBTRACT OCTANT X FROM CENTER X. CARRY RESET AT LAST ADD	
77D9	30681	22	34			
77DA	30682	2	2			
77DB	30683	70	112	LOAD (28674), HL	SET PIXEL X TO CENTER X - OCTANT X	
77DC	30684	CD	205			
77DD	30685	CE	206			
77DE	30686	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
77DF	30687	2A	42			
77E0	30688	4	4			
77E1	30689	70	112	LOAD HL, (28676)	LOAD PIXEL Y INTO HL	
77E2	30690	2B	43	DEC HL	DECREMENT PIXEL Y	
77E3	30691	22	34			
77E4	30692	4	4			
77E5	30693	70	112	LOAD (28676), HL	SAVE PIXEL Y	
77E6	30694	ED	237			
77E7	30695	5B	91			
77E8	30696	33	51			
77E9	30697	70	112	LOAD DE, (28723)	LOAD CENTER Y INTO DE	
77EA	30698	37	55	SCF		
77EB	30699	3F	63	CCF	CARRY FLAG TO ZERO	
77EC	30700	ED	237			
77ED	30701	52	82	SBC HL, DE	SUBTRACT CENTER Y FROM PIXEL Y. SIGN FLAG NEGATIVE IF PIXEL Y < CENTER Y	
77EE	30702	FA	250			
77EF	30703	F4	244			
77F0	30704	77	119	JP M, 30708	EXIT	
77F1	30705	C3	195			
77F2	30706	DC	220			
77F3	30707	77	119	JP, 30684	LOOP	
77F4	30708	ED	237			
77F5	30709	5B	91			
77F6	30710	27	39			
77F7	30711	70	112	LOAD DE, (28711)	LOAD OCTANT X INTO DE	
77F8	30712	2A	42			
77F9	30713	33	51			
77FA	30714	70	112	LOAD HL, (28723)	LOAD CENTER Y INTO HL	
77FB	30715	19	25	ADD HL, DE	ADD OCTANT X TO CENTER Y	
77FC	30716	22	34			
77FD	30717	4	4			
77FE	30718	70	112	LOAD (28676), HL	SET PIXEL Y TO CENTER Y + OCTANT X	
77FF	30719	ED	237			
7800	30720	5B	91			
7801	30721	29	41			
7802	30722	70	112	LOAD DE, (28713)	LOAD OCTANT Y INTO DE	
7803	30723	2A	42			
7804	30724	31	49			
7805	30725	70	112	LOAD HL, (28721)	LOAD CENTER X INTO HL	
7806	30726	ED	237			
7807	30727	52	82	SBC HL, DE	SUBTRACT OCTANT Y FROM CENTER X. CARRY RESET AT LAST ADD	
7808	30728	22	34			

PIXEL Y = PIXEL Y + 1

IF CENTER Y => PIXEL Y
THEN DO NEXT ITERATION
ELSE EXIT

FETCH "SEGMENT ENABLE"
VARIABLE INTO E

DRAW LOWER LEFT
QUADRANT OF CIRCLE IF
BIT 1 OF SEGMENT = 1

PIXEL Y = PIXEL Y - 1

IF PIXEL Y => CENTER Y
THEN DO NEXT ITERATION
ELSE EXIT

7809	30729	2	2			
780A	30730	70	112	LOAD (28674), HL	SET PIXEL X TO CENTER X - OCTANT Y	
780B	30731	CD	205			
780C	30732	CE	206			
780D	30733	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
780E	30734	2A	42			
780F	30735	4	4			
7810	30736	70	112	LOAD HL, (28676)	LOAD PIXEL Y INTO HL	
7811	30737	2B	43	DEC HL	DECREMENT PIXEL Y	PIXEL Y = PIXEL Y - 1
7812	30738	22	34			
7813	30739	4	4			
7814	30740	70	112	LOAD (28676), HL	SAVE PIXEL Y	
7815	30741	ED	237			
7816	30742	5B	91			
7817	30743	33	51			
7818	30744	70	112	LOAD DE, (28723)	LOAD CENTER Y INTO DE	
7819	30745	37	55	SCF		
781A	30746	3F	63	CCF	CARRY FLAG TO ZERO	
781B	30747	ED	237			IF PIXEL Y => CENTER Y THEN DO NEXT ITERATION ELSE EXIT
781C	30748	52	82	SBC HL, DE	SUBTRACT CENTER Y FROM PIXEL Y. SIGN FLAG NEGATIVE IF PIXEL Y < CENTER Y	
781D	30749	FA	250			
781E	30750	23	35			
781F	30751	78	120	JP M, 30755	EXIT	
7820	30752	C3	195			
7821	30753	B	11			
7822	30754	78	120	JP, 30731	LOOP	
7823	30755	21	33			
7824	30756	E	14			FETCH "SEGMENT ENABLE" VARIABLE INTO E
7825	30757	70	112	LOAD HL, 28686		
7826	30758	5E	94	LOAD E, (HL)		
7827	30759	CB	203			
7828	30760	53	83	BIT 2, E		
7829	30761	28	40			
782A	30762	6A	106	JR Z, 106	SKIP	
782B	30763	ED	237			
782C	30764	5B	91			
782D	30765	27	39			
782E	30766	70	112	LOAD DE, (28711)	LOAD OCTANT X INTO DE	
782F	30767	2A	42			
7830	30768	31	49			
7831	30769	70	112	LOAD HL, (28721)	LOAD CENTER X INTO HL	
7832	30770	37	55	SCF		
7833	30771	3F	63	CCF	CARRY FLAG TO ZERO	
7834	30772	ED	237			
7835	30773	52	82	SBC HL, DE	SUBTRACT OCTANT X FROM CENTER X	
7836	30774	22	34			
7837	30775	2	2			DRAW UPPER LEFT QUADRANT OF CIRCLE IF BIT 2 OF SEGMENT = 1
7838	30776	70	112	LOAD (28674), HL	SET PIXEL X TO CENTER X - OCTANT X	
7839	30777	ED	237			
783A	30778	5B	91			
783B	30779	29	41			
783C	30780	70	112	LOAD DE, (28713)	LOAD OCTANT Y INTO DE	
783D	30781	2A	42			
783E	30782	33	51			
783F	30783	70	112	LOAD HL, (28723)	LOAD CENTER Y INTO HL	
7840	30784	37	55	SCF		
7841	30785	3F	63	CCF	CARRY FLAG TO ZERO	
7842	30786	ED	237			
7843	30787	52	82	SBC HL, DE	SUBTRACT OCTANT Y FROM CENTER Y	
7844	30788	22	34			
7845	30789	4	4			
7846	30790	70	112	LOAD (28676), HL	SET PIXEL Y TO CENTER Y - OCTANT Y	
7847	30791	CD	205			
7848	30792	CE	206			
7849	30793	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
784A	30794	ED	237			
784B	30795	5B	91			
784C	30796	4	4			
784D	30797	70	112	LOAD DE, (28676)	LOAD PIXEL Y INTO DE	
784E	30798	13	19	INC DE	INCREMENT PIXEL Y	PIXEL Y = PIXEL Y + 1
784F	30799	ED	237			
7850	30800	53	83			
7851	30801	4	4			
7852	30802	70	112	LOAD (28676), DE	SAVE PIXEL Y	
7853	30803	2A	42			
7854	30804	33	51			
7855	30805	70	112	LOAD HL, (28723)	LOAD CENTER Y INTO HL	
7856	30806	37	55	SCF		
7857	30807	3F	63	CCF	CARRY FLAG TO ZERO	
7858	30808	ED	237			IF CENTER Y => PIXEL Y THEN DO NEXT ITERATION ELSE EXIT
7859	30809	52	82	SBC HL, DE	SUBTRACT PIXEL Y FROM CENTER Y. SIGN FLAG NEGATIVE IF CENTER Y < PIXEL Y	
785A	30810	FA	250			
785B	30811	60	96			
785C	30812	78	120	JP M, 30816	EXIT	
785D	30813	C3	195			
785E	30814	47	71			
785F	30815	78	120	JP, 309791	LOOP	
7860	30816	ED	237			
7861	30817	5B	91			
7862	30818	29	41			
7863	30819	70	112	LOAD DE, (28713)	LOAD OCTANT Y INTO DE	
7864	30820	2A	42			
7865	30821	31	49			
7866	30822	70	112	LOAD HL, (28721)	LOAD CENTER X INTO HL	
7867	30823	37	55	SCF		
7868	30824	3F	63	CCF	CARRY FLAG TO ZERO	
7869	30825	ED	237			
786A	30826	52	82	SBC HL, DE	SUBTRACT OCTANT Y FROM CENTER X	
786B	30827	22	34			
786C	30828	2	2			
786D	30829	70	112	LOAD (28674), HL	SET PIXEL X TO CENTER X - OCTANT Y	
786E	30830	ED	237			
786F	30831	5B	91			

7870	30832	27	39			
7871	30833	70	112	LOAD DE, (28711)	LOAD OCTANT X INTO DE	
7872	30834	2A	42			
7873	30835	33	51			
7874	30836	70	112	LOAD HL, (28723)	LOAD CENTER Y INTO HL	
7875	30837	37	55	SCF		
7876	30838	3F	63	CCF	CARRY FLAG TO ZERO	
7877	30839	ED	237			
7878	30840	52	82	SBC HL, DE	SUBTRACT OCTANT X FROM CENTER Y	
7879	30841	22	34			
787A	30842	4	4			
787B	30843	70	112	LOAD (28676), HL	SET PIXEL Y TO CENTER Y - OCTANT X	
787C	30844	CD	205			
787D	30845	CE	206			
787E	30846	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
787F	30847	ED	237			PIXEL Y = PIXEL Y + 1
7880	30848	5B	91			
7881	30849	4	4			
7882	30850	70	112	LOAD DE, (28676)	LOAD PIXEL Y INTO DE	
7883	30851	13	19	INC DE	INCREMENT PIXEL Y	
7884	30852	ED	237			
7885	30853	53	83			
7886	30854	4	4			
7887	30855	70	112	LOAD (28676), DE	SAVE PIXEL Y	
7888	30856	2A	42			IF CENTER Y => PIXEL Y THEN DO NEXT ITERATION ELSE EXIT
7889	30857	33	51			
788A	30858	70	112	LOAD HL, (28723)	LOAD CENTER Y INTO HL	
788B	30859	37	55	SCF		
788C	30860	3F	63	CCF	CARRY FLAG TO ZERO	
788D	30861	ED	237			
788E	30862	52	82	SBC HL, DE	SUBTRACT PIXEL Y FROM CENTER Y. SIGN FLAG NEGATIVE IF CENTER Y < PIXEL Y	
788F	30863	FA	250			
7890	30864	95	149			
7891	30865	78	120	JP M, 30869	EXIT	
7892	30866	C3	195			
7893	30867	7C	124			
7894	30868	78	120	JP, 30844	LOOP	
7895	30869	C9	201	RET		END
7896	30870	21	33	DRAW SOLID CIRCLE ROUTINE		MODIFY ADDRESS POINTER IN CIRCLE DRAWING ROUTINE TO POINT TO QUADRANT PLOTTING ROUTINE
7897	30871	F1	241			
7898	30872	76	118	LOAD HL, 30449		
7899	30873	22	34			
789A	30874	7E	126			
789B	30875	75	117	LOAD (30078), HL		
789C	30876	CD	205			CALL CIRCLE DRAWING ROUTINE
789D	30877	43	67			
789E	30878	75	117	CALL, 30019		
789F	30879	21	33			RESTORE ADDRESS POINTER IN CIRCLE DRAWING ROUTINE TO POINT BACK TO OCTANT PLOTTING ROUTINE
78A0	30880	D8	216			
78A1	30881	75	117	LOAD HL, 30168		
78A2	30882	22	34			
78A3	30883	7E	126			
78A4	30884	75	117	LOAD (30078), HL		
78A5	30885	C9	201	RET		END
78A6	30886	0	0			
78A7	30887	0	0			
78A8	30888	0	0			
78A9	30889	0	0			
78AA	30890	0	0			
78AB	30891	0	0			
78AC	30892	0	0			
78AD	30893	0	0			
78AE	30894	0	0			
78AF	30895	0	0			
78B0	30896	0	0			
78B1	30897	0	0			
78B2	30898	0	0			
78B3	30899	0	0			
78B4	30900	0	0			
78B5	30901	0	0			
78B6	30902	0	0			
78B7	30903	0	0			
78B8	30904	0	0			
78B9	30905	0	0			
78BA	30906	0	0			
78BB	30907	0	0			
78BC	30908	0	0			
78BD	30909	ED	237	DRAW RECTANGE		CHECK THAT X2 > X1 AND ABORT IF FALSE
78BE	30910	5B	91			
78BF	30911	A	10			
78C0	30912	70	112	LOAD DE, (28682)	LOAD X2 INTO DE	
78C1	30913	2A	42			
78C2	30914	6	6			
78C3	30915	70	112	LOAD HL, (28678)	LOAD X1 INTO HL	
78C4	30916	37	55	SCF		
78C5	30917	3F	63	CCF	CARRY FLAG TO ZERO	
78C6	30918	ED	237			
78C7	30919	52	82	SBC HL, DE	SUBTRACT X2 FROM X1. SIGN FLAG NEGATIVE X2 > X1	
78C8	30920	FA	250			
78C9	30921	CC	204			
78CA	30922	78	120	JP M, 30924		
78CB	30923	C9	201	RET		
78CC	30924	ED	237			CHECK THAT Y2 > Y1 AND ABORT IF FALSE
78CD	30925	5B	91			
78CE	30926	C	12			
78CF	30927	70	112	LOAD DE, (28684)	LOAD Y2 INTO DE	
78D0	30928	2A	42			
78D1	30929	8	8			
78D2	30930	70	112	LOAD HL, (28680)	LOAD Y1 INTO HL	
78D3	30931	37	55	SCF		
78D4	30932	3F	63	CCF	CARRY FLAG TO ZERO	
78D5	30933	ED	237			
78D6	30934	52	82	SBC HL, DE	SUBTRACT Y2 FROM Y1. SIGN FLAG NEGATIVE Y2 > Y1	

78D7	30935	FA	250			
78D8	30936	DB	219			
78D9	30937	78	120	JP M, 30939		
78DA	30938	C9	201	RET		
78DB	30939	2A	42			
78DC	30940	6	6			
78DD	30941	70	112	LOAD HL, (28678)	LOAD X1 INTO HL	PIXEL X = X1
78DE	30942	22	34			
78DF	30943	2	2			
78E0	30944	70	112	LOAD (28674), HL	SAVE PIXEL X	
78E1	30945	2A	42			
78E2	30946	8	8			
78E3	30947	70	112	LOAD HL, (28680)	LOAD Y1 INTO HL	PIXEL Y = Y1
78E4	30948	22	34			
78E5	30949	4	4			
78E6	30950	70	112	LOAD (28676), HL	SAVE PIXEL Y	
78E7	30951	CD	205			
78E8	30952	CE	206			SET PIXEL
78E9	30953	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
78EA	30954	ED	237			
78EB	30955	5B	91			
78EC	30956	2	2			
78ED	30957	70	112	LOAD DE, (28674)	LOAD PIXEL X INTO DE	PIXEL X = PIXEL X + 1
78EE	30958	13	19	INC DE	INCREMENT PIXEL X	
78EF	30959	ED	237			
78F0	30960	53	83			
78F1	30961	2	2			
78F2	30962	70	112	LOAD (28674), DE	SAVE PIXEL X	
78F3	30963	2A	42			
78F4	30964	A	10			
78F5	30965	70	112	LOAD HL, (28682)	LOAD X2 INTO HL	
78F6	30966	37	55	SCF		
78F7	30967	3F	63	CCF	CARRY FLAG TO ZERO	
78F8	30968	ED	237			IF PIXEL X <= X2 THEN DO NEXT ITERATION, ELSE EXIT
78F9	30969	52	82	SBC HL, DE	SUBTRACT PIXEL X FROM X2. SIGN FLAG NEGATIVE IF X2 < PIXEL X	
78FA	30970	FA	250			
78FB	30971	0	0			
78FC	30972	79	121	JP M, 30976	EXIT	
78FD	30973	C3	195			
78FE	30974	E7	231			
78FF	30975	78	120	JP, 30951	LOOP	
7900	30976	1B	27	DEC DE		
7901	30977	ED	237			
7902	30978	53	83			PIXEL X = PIXEL X - 1 = X2
7903	30979	2	2			
7904	30980	70	112	LOAD (28674), DE	SAVE PIXEL X	
7905	30981	CD	205			
7906	30982	CE	206			SET PIXEL
7907	30983	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
7908	30984	ED	237			
7909	30985	5B	91			
790A	30986	4	4			
790B	30987	70	112	LOAD DE, (28676)	LOAD PIXEL Y INTO DE	PIXEL Y = PIXEL Y + 1
790C	30988	13	19	INC DE	INCREMENT PIXEL Y	
790D	30989	ED	237			
790E	30990	53	83			
790F	30991	4	4			
7910	30992	70	112	LOAD (28676), DE	SAVE PIXEL Y	
7911	30993	2A	42			
7912	30994	C	12			
7913	30995	70	112	LOAD HL, (28684)	LOAD Y2 INTO HL	
7914	30996	37	55	SCF		
7915	30997	3F	63	CCF	CARRY FLAG TO ZERO	
7916	30998	ED	237			IF PIXEL Y <= Y2 THEN DO NEXT ITERATION, ELSE EXIT
7917	30999	52	82	SBC HL, DE	SUBTRACT PIXEL Y FROM Y2. SIGN FLAG NEGATIVE IF Y2 < PIXEL Y	
7918	31000	FA	250			
7919	31001	1E	30			
791A	31002	79	121	JP M, 31006	EXIT	
791B	31003	C3	195			
791C	31004	5	5			
791D	31005	79	121	JP, 30981	LOOP	
791E	31006	1B	27	DEC DE		
791F	31007	ED	237			
7920	31008	53	83			PIXEL Y = PIXEL Y - 1 = Y2
7921	31009	4	4			
7922	31010	70	112	LOAD (28676), DE	SAVE PIXEL Y	
7923	31011	CD	205			
7924	31012	CE	206			SET PIXEL
7925	31013	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
7926	31014	ED	237			
7927	31015	5B	91			
7928	31016	2	2			
7929	31017	70	112	LOAD DE, (28674)	LOAD PIXEL X INTO DE	
792A	31018	1B	27	DEC DE	DECREMENT PIXEL X	PIXEL X = PIXEL X - 1
792B	31019	ED	237			
792C	31020	53	83			
792D	31021	2	2			
792E	31022	70	112	LOAD (28674), DE	SAVE PIXEL X	
792F	31023	2A	42			
7930	31024	6	6			
7931	31025	70	112	LOAD HL, (28678)	LOAD X1 INTO HL	
7932	31026	37	55	SCF		
7933	31027	3F	63	CCF	CARRY FLAG TO ZERO	
7934	31028	ED	237			IF PIXEL X > X1 THEN DO NEXT ITERATION, ELSE EXIT
7935	31029	52	82	SBC HL, DE	SUBTRACT PIXEL X FROM X1. SIGN FLAG POSITIVE IF X1 => PIXEL X	
7936	31030	F2	242			
7937	31031	3C	60			
7938	31032	79	121	JP P, 31036	EXIT	
7939	31033	C3	195			
793A	31034	23	35			
793B	31035	79	121	JP, 31011	LOOP	
793C	31036	CD	205			
793D	31037	CE	206			SET PIXEL

793E	31038	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
793F	31039	ED	237			PIXEL Y = PIXEL Y - 1
7940	31040	5B	91			
7941	31041	4	4			
7942	31042	70	112	LOAD DE, (28676)	LOAD PIXEL Y INTO DE	
7943	31043	1B	27	DEC DE	DECREMENT PIXEL Y	
7944	31044	ED	237			
7945	31045	53	83			
7946	31046	4	4			
7947	31047	70	112	LOAD (28676), DE	SAVE PIXEL Y	
7948	31048	2A	42			IF PIXEL Y > Y1 THEN DO NEXT ITERATION, ELSE EXIT
7949	31049	8	8			
794A	31050	70	112	LOAD HL, (28680)	LOAD Y1 INTO HL	
794B	31051	37	55	SCF		
794C	31052	3F	63	CCF	CARRY FLAG TO ZERO	
794D	31053	ED	237			
794E	31054	52	82	SBC HL, DE	SUBTRACT PIXEL Y FROM Y1. SIGN FLAG POSITIVE IF Y1 => PIXEL Y	
794F	31055	F2	242			
7950	31056	55	85			
7951	31057	79	121	JP P, 31061	EXIT	
7952	31058	C3	195			
7953	31059	3C	60			
7954	31060	79	121	JP, 31036	LOOP	
7955	31061	C9	201	RET		RETURN TO BASIC
7956	31062	ED	237	DRAW FILLED RECTANGE		CHECK THAT X2 > X1 AND ABORT IF FALSE
7957	31063	5B	91			
7958	31064	A	10			
7959	31065	70	112	LOAD DE, (28682)	LOAD X2 INTO DE	
795A	31066	2A	42			
795B	31067	6	6			
795C	31068	70	112	LOAD HL, (28678)	LOAD X1 INTO HL	
795D	31069	37	55	SCF		
795E	31070	3F	63	CCF	CARRY FLAG TO ZERO	
795F	31071	ED	237			
7960	31072	52	82	SBC HL, DE	SUBTRACT X2 FROM X1. SIGN FLAG NEGATIVE X2 > X1	
7961	31073	FA	250			
7962	31074	65	101			
7963	31075	79	121	JP M, 31077		
7964	31076	C9	201	RET		
7965	31077	ED	237			CHECK THAT Y2 > Y1 AND ABORT IF FALSE
7966	31078	5B	91			
7967	31079	C	12			
7968	31080	70	112	LOAD DE, (28684)	LOAD Y2 INTO DE	
7969	31081	2A	42			
796A	31082	8	8			
796B	31083	70	112	LOAD HL, (28680)	LOAD Y1 INTO HL	
796C	31084	37	55	SCF		
796D	31085	3F	63	CCF	CARRY FLAG TO ZERO	
796E	31086	ED	237			
796F	31087	52	82	SBC HL, DE	SUBTRACT Y2 FROM Y1. SIGN FLAG NEGATIVE Y2 > Y1	
7970	31088	FA	250			
7971	31089	74	116			
7972	31090	79	121	JP M, 31092		
7973	31091	C9	201	RET		
7974	31092	2A	42			PIXEL Y = Y1
7975	31093	8	8			
7976	31094	70	112	LOAD HL, (28680)	LOAD Y1 INTO HL	
7977	31095	22	34			
7978	31096	4	4			
7979	31097	70	112	LOAD (28676), HL	SAVE PIXEL Y	
797A	31098	2A	42		LOOP START POINT	PIXEL X = X1
797B	31099	6	6			
797C	31100	70	112	LOAD HL, (28678)	LOAD X1 INTO HL	
797D	31101	22	34			
797E	31102	2	2			
797F	31103	70	112	LOAD (28674), HL	SAVE PIXEL X	
7980	31104	CD	205			SET PIXEL
7981	31105	CE	206			
7982	31106	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
7983	31107	ED	237			PIXEL X = PIXEL X + 1
7984	31108	5B	91			
7985	31109	2	2			
7986	31110	70	112	LOAD DE, (28674)	LOAD PIXEL X INTO DE	
7987	31111	13	19	INC DE	INCREMENT PIXEL X	
7988	31112	ED	237			
7989	31113	53	83			
798A	31114	2	2			
798B	31115	70	112	LOAD (28674), DE	SAVE PIXEL X	
798C	31116	2A	42			IF PIXEL X <= X2 THEN DO NEXT ITERATION, ELSE EXIT
798D	31117	A	10			
798E	31118	70	112	LOAD HL, (28682)	LOAD X2 INTO HL	
798F	31119	37	55	SCF		
7990	31120	3F	63	CCF	CARRY FLAG TO ZERO	
7991	31121	ED	237			
7992	31122	52	82	SBC HL, DE	SUBTRACT PIXEL X FROM X2. SIGN FLAG NEGATIVE IF X2 < PIXEL X	
7993	31123	FA	250			
7994	31124	99	153			
7995	31125	79	121	JP M, 31129	EXIT	
7996	31126	C3	195			
7997	31127	80	128			
7998	31128	79	121	JP, 31104	LOOP	
7999	31129	ED	237			PIXEL Y = PIXEL Y + 1
799A	31130	5B	91			
799B	31131	4	4			
799C	31132	70	112	LOAD DE, (28676)	LOAD PIXEL Y INTO DE	
799D	31133	13	19	INC DE	INCREMENT PIXEL Y	
799E	31134	ED	237			
799F	31135	53	83			
79A0	31136	4	4			
79A1	31137	70	112	LOAD (28676), DE	SAVE PIXEL Y	
79A2	31138	2A	42			
79A3	31139	C	12			
79A4	31140	70	112	LOAD HL, (28684)	LOAD Y2 INTO HL	

79A5	31141	37	55	SCF		IF PIXEL Y <= Y2 THEN DO NEXT ITERATION, ELSE EXIT
79A6	31142	3F	63	CCF	CARRY FLAG TO ZERO	
79A7	31143	ED	237			
79A8	31144	52	82	SBC HL, DE	SUBTRACT PIXEL Y FROM Y2. SIGN FLAG NEGATIVE IF Y2 < PIXEL Y	
79A9	31145	FA	250			
79AA	31146	AF	175			
79AB	31147	79	121	JP M, 31151	EXIT	
79AC	31148	C3	195			
79AD	31149	7A	122			
79AE	31150	79	121	JP, 31098	LOOP	
79AF	31151	1B	27	DEC DE		PIXEL Y = PIXEL Y - 1 = Y2
79B0	31152	ED	237			
79B1	31153	53	83			
79B2	31154	4	4			
79B3	31155	70	112	LOAD (28676), DE	SAVE PIXEL Y	
79B4	31156	2A	42			PIXEL X = PIXEL X - 1 = X2
79B5	31157	2	2			
79B6	31158	70	112	LOAD HL, (28674)	LOAD PIXEL X INTO HL	
79B7	31159	2B	43	DEC HL		
79B8	31160	22	34			
79B9	31161	2	2			
79BA	31162	70	112	LOAD (28674), HL	SAVE PIXEL X	
79BB	31163	C9	201	RET		RETURN TO BASIC
79BC	31164	ED	237	READ PIXEL ROUTINE		
79BD	31165	5B	91			
79BE	31166	4	4			
79BF	31167	70	112	LOAD DE, (28676)	LOAD PIXEL Y INTO DE	
79C0	31168	CB	203			
79C1	31169	3B	59	SRL, E		
79C2	31170	CB	203			
79C3	31171	3B	59	SRL, E		
79C4	31172	CB	203			
79C5	31173	3B	59	SRL, E		
79C6	31174	CB	203			
79C7	31175	42	66	BIT 0, D	TEST BIT 0, D	
79C8	31176	28	40			
79C9	31177	2	2	JR Z, 2	SKIP NEXT INSTRUCTION IF BIT 0 = 0	
79CA	31178	CB	203			
79CB	31179	EB	235	SET 5, E	SET BIT 5, E	
79CC	31180	21	33			
79CD	31181	FF	255			
79CE	31182	FF	255	LOAD HL, 65535	ADDRESS OF PAGE REGISTER	
79CF	31183	73	115	LOAD (HL), E	WRITE E TO PAGE REGISTER	
79D0	31184	21	33			
79D1	31185	4	4			
79D2	31186	70	112	LOAD HL, 28676		
79D3	31187	56	86	LOAD D, (HL)	COPY PIXEL Y LSB INTO D	
79D4	31188	CB	203			
79D5	31189	22	34	SLA, D		
79D6	31190	CB	203			
79D7	31191	22	34	SLA, D		
79D8	31192	CB	203			
79D9	31193	FA	250	SET 7, D	E000 VIDEO MEMORY OFFSET	
79DA	31194	CB	203			
79DB	31195	F2	242	SET 6, D	E000 VIDEO MEMORY OFFSET	
79DC	31196	CB	203			
79DD	31197	EA	234	SET 5, D	E000 VIDEO MEMORY OFFSET	
79DE	31198	1E	30			
79DF	31199	0	0	LOAD E, 0	DE NOW CONTAINS LINE NUMBER (0-7) * 1024 + 0xE000 OFFSET	
79E0	31200	2A	42			
79E1	31201	2	2			
79E2	31202	70	112	LOAD HL, (28674)	FETCH PIXEL X INTO HL	
79E3	31203	19	25	ADD HL, DE	16-BIT ADDITION HL+DE. HL NOW CONTAINS PIXEL ADDRESS	
79E4	31204	6E	110	LOAD L, (HL)	LOAD THE PIXEL COLOUR VALUE INTO L	
79E5	31205	26	38			
79E6	31206	0	0	LOAD H, 0	ZERO H REGISTER	
79E7	31207	C3	195		THIS INSTRUCTION IS TEMPORARILY CHANGED TO 201 (RETURN) BY THE BUCKET POUR ROUTINE PRIOR TO "READ PIXEL" BEING CALLED TO PREVENT THE RETURN TO BASIC	
79E8	31208	9A	154			
79E9	31209	A	10	JUMP 2714	RETURN TO BASIC WITH PIXEL COLOUR VALUE - COLOUR = USR(0)	
79EA	31210	0	0			
79EB	31211	0	0			
79EC	31212	0	0			
79ED	31213	0	0			
79EE	31214	0	0			
79EF	31215	0	0			
79F0	31216	0	0			
79F1	31217	0	0			
79F2	31218	0	0			
79F3	31219	0	0			
79F4	31220	CD	205	BUCKET POUR ROUTINE		FETCH AND STORE "PLOT MODE" VARIABLE FROM BASIC
79F5	31221	7F	127			
79F6	31222	A	10	CALL, 2687		
79F7	31223	7D	125	LOAD A, L		
79F8	31224	32	50			
79F9	31225	24	36			
79FA	31226	70	112	LOAD (28708), A		
79FB	31227	2A	42			SAVE CENTER X
79FC	31228	2	2			
79FD	31229	70	112	LOAD HL, (28674)	LOAD PIXEL X INTO HL	
79FE	31230	22	34			
79FF	31231	31	49			
7A00	31232	70	112	LOAD (28721), HL	SET CENTER X TO PIXEL X	
7A01	31233	2A	42			SAVE CENTER Y
7A02	31234	4	4			
7A03	31235	70	112	LOAD HL, (28676)	LOAD PIXEL Y INTO HL	
7A04	31236	22	34			
7A05	31237	33	51			
7A06	31238	70	112	LOAD (28723), HL	SET CENTER Y TO PIXEL Y	
7A07	31239	21	33			CHANGE INSTRUCTION AT 0x79E7 TO 201 (RETURN)
7A08	31240	E7	231			
7A09	31241	79	121	LOAD HL, 31207		
7A0A	31242	36	54			

7A0B	31243	C9	201	LOAD (HL), 201		
7A0C	31244	CD	205			
7A0D	31245	BC	188			READ COLOUR VALUE AT POUR ORIGIN INTO REGISTER B
7A0E	31246	79	121	CALL, 31164	CALL "READ PIXEL" ROUTINE	
7A0F	31247	45	69	LOAD B, L		
7A10	31248	21	33			
7A11	31249	24	36			FETCH "POUR MODE" VARIABLE INTO C
7A12	31250	70	112	LOAD HL, 28708		
7A13	31251	4E	78	LOAD C, (HL)		
7A14	31252	CB	203			
7A15	31253	41	65	BIT 0, C		IF BIT 0 OF POUR MODE IS SET THEN RUN POUR DOWN, ELSE SKIP
7A16	31254	28	40			
7A17	31255	4E	78	JR Z, 78	SKIP	
7A18	31256	CD	205		LOOP TO HERE	
7A19	31257	CE	206			SET PIXEL
7A1A	31258	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
7A1B	31259	2A	42			
7A1C	31260	2	2			
7A1D	31261	70	112	LOAD HL, (28674)	LOAD PIXEL X INTO HL	
7A1E	31262	23	35	INC HL	INCREMENT PIXEL X	PIXEL X = PIXEL X + 1
7A1F	31263	22	34			
7A20	31264	2	2			
7A21	31265	70	112	LOAD (28674), HL	SAVE PIXEL X	
7A22	31266	CD	205			
7A23	31267	BC	188			
7A24	31268	79	121	CALL, 31164	CALL "READ PIXEL" ROUTINE	IS THE BACKGROUND COLOUR STILL THE SAME? IF YES THEN NEXT ITERATION, ELSE ABORT
7A25	31269	78	120	LOAD A, B		
7A26	31270	BD	189	CP L		
7A27	31271	28	40			
7A28	31272	EF	239	JR Z, -17	RELATIVE JUMP ON ZERO TO 31256	
7A29	31273	2A	42			
7A2A	31274	31	49			
7A2B	31275	70	112	LOAD HL, (28721)	LOAD CENTER X INTO HL	RESET PIXEL X TO POUR ORIGIN
7A2C	31276	22	34			
7A2D	31277	2	2			
7A2E	31278	70	112	LOAD (28674), HL	SET PIXEL X TO CENTER X	
7A2F	31279	CD	205		LOOP TO HERE	
7A30	31280	CE	206			SET PIXEL
7A31	31281	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
7A32	31282	2A	42			
7A33	31283	2	2			
7A34	31284	70	112	LOAD HL, (28674)	LOAD PIXEL X INTO HL	
7A35	31285	2B	43	DEC HL	DECREMENT PIXEL X	PIXEL X = PIXEL X - 1
7A36	31286	22	34			
7A37	31287	2	2			
7A38	31288	70	112	LOAD (28674), HL	SAVE PIXEL X	
7A39	31289	CD	205			
7A3A	31290	BC	188			
7A3B	31291	79	121	CALL, 31164	CALL "READ PIXEL" ROUTINE	IS THE BACKGROUND COLOUR STILL THE SAME? IF YES THEN NEXT ITERATION, ELSE ABORT
7A3C	31292	78	120	LOAD A, B		
7A3D	31293	BD	189	CP L		
7A3E	31294	28	40			
7A3F	31295	EF	239	JR Z, -17	RELATIVE JUMP ON ZERO TO 31279	
7A40	31296	2A	42			
7A41	31297	31	49			
7A42	31298	70	112	LOAD HL, (28721)	LOAD CENTER X INTO HL	RESET PIXEL X TO POUR ORIGIN
7A43	31299	22	34			
7A44	31300	2	2			
7A45	31301	70	112	LOAD (28674), HL	SET PIXEL X TO CENTER X	
7A46	31302	2A	42			
7A47	31303	4	4			
7A48	31304	70	112	LOAD HL, (28676)	LOAD PIXEL Y INTO HL	PIXEL Y = PIXEL Y + 1 OR PIXEL Y = PIXEL Y - 1
7A49	31305	23	35	INC HL	THIS INSTRUCTION IS TEMPORARILY CHANGED TO 43 (DEC HL) WHEN POUR DIRECTION IS UP	
7A4A	31306	22	34			
7A4B	31307	4	4			
7A4C	31308	70	112	LOAD (28676), HL	SAVE PIXEL Y	
7A4D	31309	CD	205			
7A4E	31310	BC	188			
7A4F	31311	79	121	CALL, 31164	CALL "READ PIXEL" ROUTINE	IS THE BACKGROUND COLOUR STILL THE SAME? IF YES THEN NEXT ITERATION, ELSE ABORT
7A50	31312	78	120	LOAD A, B		
7A51	31313	BD	189	CP L		
7A52	31314	28	40			
7A53	31315	C4	196	JR Z, -60	RELATIVE JUMP ON ZERO TO 31256	
7A54	31316	21	33			
7A55	31317	E7	231			RESTORE INSTRUCTION AT 0x79E7 BACK TO 195 (JUMP)
7A56	31318	79	121	LOAD HL, 31207		
7A57	31319	36	54			
7A58	31320	C3	195	LOAD (HL), 195		
7A59	31321	2A	42			
7A5A	31322	31	49			
7A5B	31323	70	112	LOAD HL, (28721)	LOAD CENTER X INTO HL	
7A5C	31324	22	34			
7A5D	31325	2	2			
7A5E	31326	70	112	LOAD (28674), HL	SET PIXEL X TO CENTER X	RESTORE PIXEL X AND PIXEL Y TO POUR ORIGIN
7A5F	31327	2A	42			
7A60	31328	33	51			
7A61	31329	70	112	LOAD HL, (28723)	LOAD CENTER Y INTO HL	
7A62	31330	22	34			
7A63	31331	4	4			
7A64	31332	70	112	LOAD (28676), HL	SET PIXEL Y TO CENTER Y	
7A65	31333	C9	201	RET		END ROUTINE
7A66	31334	CB	203			
7A67	31335	49	73	BIT 1, C		IF BIT 1 OF POUR MODE IS SET THEN CONFIGURE AND RUN POUR UP, ELSE SKIP
7A68	31336	28	40			
7A69	31337	E	14	JR Z, 14	RELATIVE JUMP ON ZERO TO 31352	
7A6A	31338	21	33			
7A6B	31339	49	73			
7A6C	31340	7A	122	LOAD HL, 31305		
7A6D	31341	36	54			CHANGE INSTRUCTION AT 0x7A49 TO 43 (DEC HL) AND RUN HORIZONTAL POUR
7A6E	31342	2B	43	LOAD (HL), 43		
7A6F	31343	CD	205			
7A70	31344	18	24			
7A71	31345	7A	122	CALL, 31256	CALL HORIZONTAL POUR	

7A72	31346	21	33			RESTORE INSTRUCTION AT 0x7A49 BACK TO 35 (INC HL)
7A73	31347	49	73			
7A74	31348	7A	122	LOAD HL, 31305		
7A75	31349	36	54			
7A76	31350	23	35	LOAD (HL), 35		
7A77	31351	C9	201	RET		END ROUTINE
7A78	31352	CB	203			IF BIT 2 OF POUR MODE IS SET THEN RUN POUR LEFT, ELSE SKIP
7A79	31353	51	81	BIT 2, C		
7A7A	31354	28	40			
7A7B	31355	4E	78	JR Z, 78	SKIP	
7A7C	31356	CD	205		LOOP TO HERE	SET PIXEL
7A7D	31357	CE	206			
7A7E	31358	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
7A7F	31359	2A	42			PIXEL Y = PIXEL Y + 1
7A80	31360	4	4			
7A81	31361	70	112	LOAD HL, (28676)	LOAD PIXEL Y INTO HL	
7A82	31362	23	35	INC HL	INCREMENT PIXEL Y	
7A83	31363	22	34			
7A84	31364	4	4			
7A85	31365	70	112	LOAD (28676), HL	SAVE PIXEL Y	
7A86	31366	CD	205			IS THE BACKGROUND COLOUR STILL THE SAME? IF YES THEN NEXT ITERATION, ELSE ABORT
7A87	31367	BC	188			
7A88	31368	79	121	CALL, 31164	CALL "READ PIXEL" ROUTINE	
7A89	31369	78	120	LOAD A, B		
7A8A	31370	BD	189	CP L		
7A8B	31371	28	40			
7A8C	31372	EF	239	JR Z, -17	RELATIVE JUMP ON ZERO TO 31356	
7A8D	31373	2A	42			RESET PIXEL Y TO POUR ORIGIN
7A8E	31374	33	51			
7A8F	31375	70	112	LOAD HL, (28723)	LOAD CENTER Y INTO HL	
7A90	31376	22	34			
7A91	31377	4	4			
7A92	31378	70	112	LOAD (28676), HL	SET PIXEL Y TO CENTER Y	
7A93	31379	CD	205		LOOP TO HERE	SET PIXEL
7A94	31380	CE	206			
7A95	31381	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
7A96	31382	2A	42			PIXEL Y = PIXEL Y - 1
7A97	31383	4	4			
7A98	31384	70	112	LOAD HL, (28676)	LOAD PIXEL Y INTO HL	
7A99	31385	2B	43	DEC HL	DECREMENT PIXEL Y	
7A9A	31386	22	34			
7A9B	31387	4	4			
7A9C	31388	70	112	LOAD (28676), HL	SAVE PIXEL Y	
7A9D	31389	CD	205			IS THE BACKGROUND COLOUR STILL THE SAME? IF YES THEN NEXT ITERATION, ELSE ABORT
7A9E	31390	BC	188			
7A9F	31391	79	121	CALL, 31164	CALL "READ PIXEL" ROUTINE	
7AA0	31392	78	120	LOAD A, B		
7AA1	31393	BD	189	CP L		
7AA2	31394	28	40			
7AA3	31395	EF	239	JR Z, -17	RELATIVE JUMP ON ZERO TO 31379	
7AA4	31396	2A	42			RESET PIXEL Y TO POUR ORIGIN
7AA5	31397	33	51			
7AA6	31398	70	112	LOAD HL, (28723)	LOAD CENTER Y INTO HL	
7AA7	31399	22	34			
7AA8	31400	4	4			
7AA9	31401	70	112	LOAD (28676), HL	SET PIXEL Y TO CENTER Y	
7AAA	31402	2A	42			PIXEL X = PIXEL X - 1 OR PIXEL X = PIXEL X + 1
7AAB	31403	2	2			
7AAC	31404	70	112	LOAD HL, (28674)	LOAD PIXEL X INTO HL	
7AAD	31405	2B	43	DEC HL	THIS INSTRUCTION IS TEMPORARILY CHANGED TO 35 (INC HL) WHEN POUR DIRECTION IS RIGHT	
7AAE	31406	22	34			
7AAF	31407	2	2			
7AB0	31408	70	112	LOAD (28674), HL	SAVE PIXEL X	
7AB1	31409	CD	205			IS THE BACKGROUND COLOUR STILL THE SAME? IF YES THEN NEXT ITERATION, ELSE ABORT
7AB2	31410	BC	188			
7AB3	31411	79	121	CALL, 31164	CALL "READ PIXEL" ROUTINE	
7AB4	31412	78	120	LOAD A, B		
7AB5	31413	BD	189	CP L		
7AB6	31414	28	40			
7AB7	31415	C4	196	JR Z, -60	RELATIVE JUMP ON ZERO TO 31356	
7AB8	31416	21	33			RESTORE INSTRUCTION AT 0x79E7 BACK TO 195 (JUMP)
7AB9	31417	E7	231			
7ABA	31418	79	121	LOAD HL, 31207		
7ABB	31419	36	54			
7ABC	31420	C3	195	LOAD (HL), 195		
7ABD	31421	2A	42			RESTORE PIXEL X AND PIXEL Y TO POUR ORIGIN
7ABE	31422	31	49			
7ABF	31423	70	112	LOAD HL, (28721)	LOAD CENTER X INTO HL	
7AC0	31424	22	34			
7AC1	31425	2	2			
7AC2	31426	70	112	LOAD (28674), HL	SET PIXEL X TO CENTER X	
7AC3	31427	2A	42			
7AC4	31428	33	51			
7AC5	31429	70	112	LOAD HL, (28723)	LOAD CENTER Y INTO HL	
7AC6	31430	22	34			
7AC7	31431	4	4			
7AC8	31432	70	112	LOAD (28676), HL	SET PIXEL Y TO CENTER Y	
7AC9	31433	C9	201	RET		END ROUTINE
7ACA	31434	CB	203			IF BIT 3 OF POUR MODE IS SET THEN CONFIGURE AND RUN POUR RIGHT, ELSE SKIP
7ACB	31435	59	89	BIT 3, C		
7ACC	31436	28	40			
7ACD	31437	D	13	JR Z, 13	RELATIVE JUMP ON ZERO TO 31451	
7ACE	31438	21	33			CHANGE INSTRUCTION AT 0x7AAD TO 35 (INC HL) AND RUN VERTICAL POUR
7ACF	31439	AD	173			
7AD0	31440	7A	122	LOAD HL, 31405		
7AD1	31441	36	54			
7AD2	31442	23	35	LOAD (HL), 35		
7AD3	31443	CD	205			
7AD4	31444	7C	124			
7AD5	31445	7A	122	CALL, 31356	CALL HORIZONTAL POUR	
7AD6	31446	21	33			RESTORE INSTRUCTION AT 0x7AAD BACK TO 43 (DEC HL)
7AD7	31447	AD	173			
7AD8	31448	7A	122	LOAD HL, 31405		

7AD9	31449	36	54			0x7AAB BACK TO 45 (DEC HL)
7ADA	31450	2B	43	LOAD (HL), 43		
7ADB	31451	C9	201	RET		END ROUTINE
7ADC	31452	0	0			
7ADD	31453	0	0			
7ADE	31454	0	0			
7ADF	31455	0	0			
7AE0	31456	0	0			
7AE1	31457	0	0			
7AE2	31458	0	0			
7AE3	31459	0	0			
7AE4	31460	CD	205	TEXT ROUTINE		FETCH "CHARACTER" VARIABLE FROM BASIC AND SHIFT INTO DE
7AE5	31461	7F	127			
7AE6	31462	A	10	CALL, 2687		
7AE7	31463	5D	93	LOAD E, L		
7AE8	31464	54	84	LOAD D, H		
7AE9	31465	2A	42			PIXEL X = (COLUMN * 8) + 1
7AEA	31466	F	15			
7AEB	31467	70	112	LOAD HL,(28678)	LOAD COLUMN INTO HL	
7AEC	31468	CB	203			
7AED	31469	25	37	SLA, L	MULTIPLY HL BY 8, STEP 1 OF 6	
7AEE	31470	CB	203			
7AEF	31471	14	20	RL, H	MULTIPLY HL BY 8, STEP 2 OF 6	
7AF0	31472	CB	203			
7AF1	31473	25	37	SLA, L	MULTIPLY HL BY 8, STEP 3 OF 6	
7AF2	31474	CB	203			
7AF3	31475	14	20	RL, H	MULTIPLY HL BY 8, STEP 4 OF 6	
7AF4	31476	CB	203			
7AF5	31477	25	37	SLA, L	MULTIPLY HL BY 8, STEP 5 OF 6	
7AF6	31478	CB	203			
7AF7	31479	14	20	RL, H	MULTIPLY HL BY 8, STEP 6 OF 6	
7AF8	31480	23	35	INC HL	HL + 1	
7AF9	31481	22	34			
7AFA	31482	2	2			
7AFB	31483	70	112	LOAD (28674), HL	SET PIXEL X TO (COLUMN * 8) + 1	
7AFC	31484	22	34			
7AFD	31485	20	32			
7AFE	31486	70	112	LOAD (28704), HL	SAVE CHARACTER CELL X ORIGIN	
7AFF	31487	2A	42			PIXEL Y = ROW * 8
7B00	31488	11	17			
7B01	31489	70	112	LOAD HL,(28680)	LOAD ROW INTO HL	
7B02	31490	CB	203			
7B03	31491	25	37	SLA, L	MULTIPLY HL BY 8, STEP 1 OF 6	
7B04	31492	CB	203			
7B05	31493	14	20	RL, H	MULTIPLY HL BY 8, STEP 2 OF 6	
7B06	31494	CB	203			
7B07	31495	25	37	SLA, L	MULTIPLY HL BY 8, STEP 3 OF 6	
7B08	31496	CB	203			
7B09	31497	14	20	RL, H	MULTIPLY HL BY 8, STEP 4 OF 6	
7B0A	31498	CB	203			
7B0B	31499	25	37	SLA, L	MULTIPLY HL BY 8, STEP 5 OF 6	
7B0C	31500	CB	203			
7B0D	31501	14	20	RL, H	MULTIPLY HL BY 8, STEP 6 OF 6	
7B0E	31502	22	34			
7B0F	31503	4	4			
7B10	31504	70	112	LOAD (28676), HL	SET PIXEL Y TO ROW * 8	
7B11	31505	CB	203			CHAR ROM POINTER = (CHARACTER * 8) + 31744
7B12	31506	23	35	SLA, E	MULTIPLY CHARACTER BY 8, STEP 1 OF 6	
7B13	31507	CB	203			
7B14	31508	12	18	RL, D	MULTIPLY CHARACTER BY 8, STEP 2 OF 6	
7B15	31509	CB	203			
7B16	31510	23	35	SLA, E	MULTIPLY CHARACTER BY 8, STEP 3 OF 6	
7B17	31511	CB	203			
7B18	31512	12	18	RL, D	MULTIPLY CHARACTER BY 8, STEP 4 OF 6	
7B19	31513	CB	203			
7B1A	31514	23	35	SLA, E	MULTIPLY CHARACTER BY 8, STEP 5 OF 6	
7B1B	31515	CB	203			
7B1C	31516	12	18	RL, D	MULTIPLY CHARACTER BY 8, STEP 6 OF 6	
7B1D	31517	21	33			
7B1E	31518	0	0			
7B1F	31519	7C	124	LOAD HL, 31744	CHARACTER ROM START ADDRESS INTO HL	
7B20	31520	19	25	ADD HL, DE	ADD CHARACTER * 8 TO CHARACTER ROM START ADDRESS	
7B21	31521	22	34			
7B22	31522	22	34			
7B23	31523	70	112	LOAD (28706), HL	SAVE CHAR ROM POINTER	
7B24	31524	2A	42		LOOP STARTS HERE	TEST BIT 7 AND SET PIXEL IF TRUE
7B25	31525	22	34			
7B26	31526	70	112	LOAD (28706), HL	LOAD CHAR ROM POINTER INTO HL	
7B27	31527	CB	203			
7B28	31528	7E	126	BIT 7, (HL)	TEST BIT 7 OF CHARACTER BYTE	
7B29	31529	28	40			
7B2A	31530	3	3	JR Z, 3	SKIP SET PIXEL IF FALSE	
7B2B	31531	CD	205			
7B2C	31532	CE	206			
7B2D	31533	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
7B2E	31534	2A	42			PIXEL X = PIXEL X + 1
7B2F	31535	2	2			
7B30	31536	70	112	LOAD HL, (28674)	LOAD PIXEL X INTO HL	
7B31	31537	23	35	INC HL	INCREMENT PIXEL X	
7B32	31538	22	34			
7B33	31539	2	2			
7B34	31540	70	112	LOAD (28674), HL	SAVE PIXEL X	
7B35	31541	2A	42			TEST BIT 6 AND SET PIXEL IF TRUE
7B36	31542	22	34			
7B37	31543	70	112	LOAD (28706), HL	LOAD CHAR ROM POINTER INTO HL	
7B38	31544	CB	203			
7B39	31545	76	118	BIT 6, (HL)	TEST BIT 6 OF CHARACTER BYTE	
7B3A	31546	28	40			
7B3B	31547	3	3	JR Z, 3	SKIP SET PIXEL IF FALSE	
7B3C	31548	CD	205			
7B3D	31549	CE	206			
7B3E	31550	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
7B3F	31551	2A	42			

7B40	31552	2	2			PIXEL X = PIXEL X + 1
7B41	31553	70	112	LOAD HL, (28674)	LOAD PIXEL X INTO HL	
7B42	31554	23	35	INC HL	INCREMENT PIXEL X	
7B43	31555	22	34			
7B44	31556	2	2			
7B45	31557	70	112	LOAD (28674), HL	SAVE PIXEL X	
7B46	31558	2A	42			TEST BIT 5 AND SET PIXEL IF TRUE
7B47	31559	22	34			
7B48	31560	70	112	LOAD (28706), HL	LOAD CHAR ROM POINTER INTO HL	
7B49	31561	CB	203			
7B4A	31562	6E	110	BIT 5, (HL)	TEST BIT 5 OF CHARACTER BYTE	
7B4B	31563	28	40			
7B4C	31564	3	3	JR Z, 3	SKIP SET PIXEL IF FALSE	
7B4D	31565	CD	205			
7B4E	31566	CE	206			
7B4F	31567	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
7B50	31568	2A	42			PIXEL X = PIXEL X + 1
7B51	31569	2	2			
7B52	31570	70	112	LOAD HL, (28674)	LOAD PIXEL X INTO HL	
7B53	31571	23	35	INC HL	INCREMENT PIXEL X	
7B54	31572	22	34			
7B55	31573	2	2			
7B56	31574	70	112	LOAD (28674), HL	SAVE PIXEL X	
7B57	31575	2A	42			TEST BIT 4 AND SET PIXEL IF TRUE
7B58	31576	22	34			
7B59	31577	70	112	LOAD (28706), HL	LOAD CHAR ROM POINTER INTO HL	
7B5A	31578	CB	203			
7B5B	31579	66	102	BIT 4, (HL)	TEST BIT 4 OF CHARACTER BYTE	
7B5C	31580	28	40			
7B5D	31581	3	3	JR Z, 3	SKIP SET PIXEL IF FALSE	
7B5E	31582	CD	205			
7B5F	31583	CE	206			
7B60	31584	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
7B61	31585	2A	42			PIXEL X = PIXEL X + 1
7B62	31586	2	2			
7B63	31587	70	112	LOAD HL, (28674)	LOAD PIXEL X INTO HL	
7B64	31588	23	35	INC HL	INCREMENT PIXEL X	
7B65	31589	22	34			
7B66	31590	2	2			
7B67	31591	70	112	LOAD (28674), HL	SAVE PIXEL X	
7B68	31592	2A	42			TEST BIT 3 AND SET PIXEL IF TRUE
7B69	31593	22	34			
7B6A	31594	70	112	LOAD (28706), HL	LOAD CHAR ROM POINTER INTO HL	
7B6B	31595	CB	203			
7B6C	31596	5E	94	BIT 3, (HL)	TEST BIT 3 OF CHARACTER BYTE	
7B6D	31597	28	40			
7B6E	31598	3	3	JR Z, 3	SKIP SET PIXEL IF FALSE	
7B6F	31599	CD	205			
7B70	31600	CE	206			
7B71	31601	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
7B72	31602	2A	42			PIXEL X = PIXEL X + 1
7B73	31603	2	2			
7B74	31604	70	112	LOAD HL, (28674)	LOAD PIXEL X INTO HL	
7B75	31605	23	35	INC HL	INCREMENT PIXEL X	
7B76	31606	22	34			
7B77	31607	2	2			
7B78	31608	70	112	LOAD (28674), HL	SAVE PIXEL X	
7B79	31609	2A	42			TEST BIT 2 AND SET PIXEL IF TRUE
7B7A	31610	22	34			
7B7B	31611	70	112	LOAD (28706), HL	LOAD CHAR ROM POINTER INTO HL	
7B7C	31612	CB	203			
7B7D	31613	56	86	BIT 2, (HL)	TEST BIT 2 OF CHARACTER BYTE	
7B7E	31614	28	40			
7B7F	31615	3	3	JR Z, 3	SKIP SET PIXEL IF FALSE	
7B80	31616	CD	205			
7B81	31617	CE	206			
7B82	31618	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
7B83	31619	2A	42			PIXEL X = PIXEL X + 1
7B84	31620	2	2			
7B85	31621	70	112	LOAD HL, (28674)	LOAD PIXEL X INTO HL	
7B86	31622	23	35	INC HL	INCREMENT PIXEL X	
7B87	31623	22	34			
7B88	31624	2	2			
7B89	31625	70	112	LOAD (28674), HL	SAVE PIXEL X	
7B8A	31626	2A	42			TEST BIT 1 AND SET PIXEL IF TRUE
7B8B	31627	22	34			
7B8C	31628	70	112	LOAD (28706), HL	LOAD CHAR ROM POINTER INTO HL	
7B8D	31629	CB	203			
7B8E	31630	4E	78	BIT 1, (HL)	TEST BIT 1 OF CHARACTER BYTE	
7B8F	31631	28	40			
7B90	31632	3	3	JR Z, 3	SKIP SET PIXEL IF FALSE	
7B91	31633	CD	205			
7B92	31634	CE	206			
7B93	31635	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
7B94	31636	2A	42			PIXEL X = PIXEL X + 1
7B95	31637	2	2			
7B96	31638	70	112	LOAD HL, (28674)	LOAD PIXEL X INTO HL	
7B97	31639	23	35	INC HL	INCREMENT PIXEL X	
7B98	31640	22	34			
7B99	31641	2	2			
7B9A	31642	70	112	LOAD (28674), HL	SAVE PIXEL X	
7B9B	31643	2A	42			TEST BIT 0 AND SET PIXEL IF TRUE
7B9C	31644	22	34			
7B9D	31645	70	112	LOAD (28706), HL	LOAD CHAR ROM POINTER INTO HL	
7B9E	31646	CB	203			
7B9F	31647	46	70	BIT 0, (HL)	TEST BIT 0 OF CHARACTER BYTE	
7BA0	31648	28	40			
7BA1	31649	3	3	JR Z, 3	SKIP SET PIXEL IF FALSE	
7BA2	31650	CD	205			
7BA3	31651	CE	206			
7BA4	31652	70	112	CALL, 28878	CALL "SET PIXEL" ROUTINE	
7BA5	31653	2A	42			
7BA6	31654	20	32			

7BA7	31655	70	112	LOAD HL, (28704)	LOAD CHARACTER CELL X ORIGIN	PIXEL X = BACK TO ORIGIN
7BA8	31656	22	34			
7BA9	31657	2	2			
7BAA	31658	70	112	LOAD (28674), HL	SAVE PIXEL X	
7BAB	31659	2A	42			INCREMENT CHARACTER ROM POINTER TO NEXT BYTE / LINE
7BAC	31660	22	34			
7BAD	31661	70	112	LOAD (28706), HL	LOAD CHAR ROM POINTER INTO HL	
7BAE	31662	23	35	INC HL		
7BAF	31663	22	34			
7BB0	31664	22	34			
7BB1	31665	70	112	LOAD (28706), HL	SAVE UPDATED CHAR ROM POINTER	
7BB2	31666	2A	42			PIXEL Y = PIXEL Y + 1
7BB3	31667	4	4			
7BB4	31668	70	112	LOAD HL, (28676)	LOAD PIXEL Y INTO HL	
7BB5	31669	23	35	INC HL	INCREMENT PIXEL Y	
7BB6	31670	22	34			
7BB7	31671	4	4			
7BB8	31672	70	112	LOAD (28676), HL	SAVE PIXEL Y	
7BB9	31673	21	33			INCREMENT CHAR PLOTTING LINE COUNT, TEST IF = 8 AND TERMINATE IF TRUE. ELSE PLOT NEXT LINE OF CHARACTER
7BBA	31674	1F	31			
7BBB	31675	70	112	LOAD HL, 28703		
7BBC	31676	34	52	INC, (HL)		
7BBD	31677	CB	203			
7BBE	31678	5E	94	BIT 3, (HL)		
7BBF	31679	28	40			
7BC0	31680	1B	27	JR Z, 27	RELATIVE JUMP TO 31708	
7BC1	31681	36	54			RESET LINE COUNTER
7BC2	31682	0	0	LOAD (HL), 0		
7BC3	31683	21	33			INCREMENT COLUMN COUNT. IF COLUMN COUNT = 80 THEN RESET TO ZERO
7BC4	31684	F	15			
7BC5	31685	70	112	LOAD HL, 28687	ADDRESS OF "COLUMN" VARIABLE INTO HL	
7BC6	31686	34	52	INC, (HL)	INCREMENT COLUMN COUNT	
7BC7	31687	5E	94	LOAD E, (HL)	COLUMN INTO E	
7BC8	31688	3E	62			
7BC9	31689	50	80	LOAD A, 80	80 INTO A	
7BCA	31690	93	147	SUB E	SUBTRACT COLUMN FROM 80	
7BCB	31691	20	32			
7BCC	31692	E	14	JR NZ, 14	IF ANSWER NOT 0 THEN COLUMN < 80. DON'T RESET TO 0 AND DON'T INCREMENT ROW	
7BCD	31693	36	54			
7BCE	31694	0	0	LOAD (HL), 0	RESET "COLUMN" TO 0	
7BCF	31695	21	33			ON COLUMN COUNT RESET, INCREMENT ROW COUNT. IF ROW COUNT = 60 THEN RESET TO ZERO
7BD0	31696	11	17			
7BD1	31697	70	112	LOAD HL, 28689	ADDRESS OF "ROW" VARIABLE INTO HL	
7BD2	31698	34	52	INC, (HL)	INCREMENT ROW COUNT	
7BD3	31699	5E	94	LOAD E, (HL)	ROW INTO E	
7BD4	31700	3E	62			
7BD5	31701	3C	60	LOAD A, 60	80 INTO A	
7BD6	31702	93	147	SUB E	SUBTRACT ROW FROM 60	
7BD7	31703	20	32			
7BD8	31704	2	2	JR NZ, 2	IF ANSWER NOT 0 THEN ROW < 60. DON'T RESET TO 0	
7BD9	31705	36	54			
7BDA	31706	0	0	LOAD (HL), 0	RESET "COLUMN" TO 0	
7BDB	31707	C9	201	RET		RETURN TO BASIC
7BDC	31708	C3	195			JUMP FOR NEXT CHARACTER LINE
7BDD	31709	24	36			
7BDE	31710	7B	123	JP, 31524		
7BDF	31711	0	0			
7BE0	31712	0	0			
7BE1	31713	0	0			
7BE2	31714	0	0			
7BE3	31715	0	0			
7BE4	31716	0	0			
7BE5	31717	0	0			
7BE6	31718	0	0			
7BE7	31719	0	0			
7BE8	31720	0	0			
7BE9	31721	0	0			
7BEA	31722	0	0			
7BEB	31723	0	0			
7BEC	31724	0	0			
7BED	31725	0	0			
7BEE	31726	0	0			
7BEF	31727	0	0			
7BF0	31728	0	0			
7BF1	31729	0	0			
7BF2	31730	0	0			
7BF3	31731	0	0			
7BF4	31732	0	0			
7BF5	31733	0	0			
7BF6	31734	0	0			
7BF7	31735	0	0			
7BF8	31736	0	0			
7BF9	31737	0	0			
7BFA	31738	0	0			
7BFB	31739	0	0			
7BFC	31740	0	0			
7BFD	31741	0	0			
7BFE	31742	0	0			
7BFF	31743	0	0			
7C00	31744	0	0	CHARACTER ROM. 128 CHARACTER "PETSCII" SET: ASCII CONTROL CODES 0 - 31 REPLACED WITH GRAPHICAL CHARACTERS		0 GRAPHICAL CHARACTER
7C01	31745	0	0			
7C02	31746	0	0			
7C03	31747	0	0			
7C04	31748	0	0			
7C05	31749	0	0			
7C06	31750	0	0			
7C07	31751	0	0			
7C08	31752	F0	240			1 GRAPHICAL CHARACTER
7C09	31753	F0	240			
7C0A	31754	F0	240			
7C0B	31755	F0	240			
7C0C	31756	F0	240			
7C0D	31757	F0	240			

7C0E	31758	F0	240			
7C0F	31759	F0	240			
7C10	31760	0	0			
7C11	31761	0	0			
7C12	31762	0	0			
7C13	31763	0	0			2 GRAPHICAL CHARACTER
7C14	31764	FF	255			
7C15	31765	FF	255			
7C16	31766	FF	255			
7C17	31767	FF	255			
7C18	31768	FF	255			
7C19	31769	0	0			
7C1A	31770	0	0			
7C1B	31771	0	0			
7C1C	31772	0	0			3 GRAPHICAL CHARACTER
7C1D	31773	0	0			
7C1E	31774	0	0			
7C1F	31775	0	0			
7C20	31776	0	0			
7C21	31777	0	0			
7C22	31778	0	0			
7C23	31779	0	0			4 GRAPHICAL CHARACTER
7C24	31780	0	0			
7C25	31781	0	0			
7C26	31782	0	0			
7C27	31783	FF	255			
7C28	31784	80	128			
7C29	31785	80	128			
7C2A	31786	80	128			
7C2B	31787	80	128			5 GRAPHICAL CHARACTER
7C2C	31788	80	128			
7C2D	31789	80	128			
7C2E	31790	80	128			
7C2F	31791	80	128			
7C30	31792	AA	170			
7C31	31793	55	85			
7C32	31794	AA	170			
7C33	31795	55	85			6 GRAPHICAL CHARACTER
7C34	31796	AA	170			
7C35	31797	55	85			
7C36	31798	AA	170			
7C37	31799	55	85			
7C38	31800	1	1			
7C39	31801	1	1			
7C3A	31802	1	1			
7C3B	31803	1	1			7 GRAPHICAL CHARACTER
7C3C	31804	1	1			
7C3D	31805	1	1			
7C3E	31806	1	1			
7C3F	31807	1	1			
7C40	31808	0	0			
7C41	31809	0	0			
7C42	31810	0	0			
7C43	31811	0	0			8 GRAPHICAL CHARACTER
7C44	31812	AA	170			
7C45	31813	55	85			
7C46	31814	AA	170			
7C47	31815	55	85			
7C48	31816	99	153			
7C49	31817	33	51			
7C4A	31818	66	102			
7C4B	31819	CC	204			9 GRAPHICAL CHARACTER
7C4C	31820	99	153			
7C4D	31821	33	51			
7C4E	31822	66	102			
7C4F	31823	CC	204			
7C50	31824	3	3			
7C51	31825	3	3			
7C52	31826	3	3			
7C53	31827	3	3			10 GRAPHICAL CHARACTER
7C54	31828	3	3			
7C55	31829	3	3			
7C56	31830	3	3			
7C57	31831	3	3			
7C58	31832	8	8			
7C59	31833	8	8			
7C5A	31834	8	8			
7C5B	31835	8	8			11 GRAPHICAL CHARACTER
7C5C	31836	0F	15			
7C5D	31837	8	8			
7C5E	31838	8	8			
7C5F	31839	8	8			
7C60	31840	0	0			
7C61	31841	0	0			
7C62	31842	0	0			
7C63	31843	0	0			12 GRAPHICAL CHARACTER
7C64	31844	0F	15			
7C65	31845	0F	15			
7C66	31846	0F	15			
7C67	31847	0F	15			
7C68	31848	8	8			
7C69	31849	8	8			
7C6A	31850	8	8			
7C6B	31851	8	8			13 GRAPHICAL CHARACTER
7C6C	31852	0F	15			
7C6D	31853	0	0			
7C6E	31854	0	0			
7C6F	31855	0	0			
7C70	31856	0	0			
7C71	31857	0	0			
7C72	31858	0	0			
7C73	31859	0	0			14 GRAPHICAL CHARACTER
7C74	31860	F8	248			

7C75	31861	8	8			
7C76	31862	8	8			
7C77	31863	8	8			
7C78	31864	0	0			
7C79	31865	0	0			
7C7A	31866	0	0			
7C7B	31867	0	0			
7C7C	31868	0	0			
7C7D	31869	0	0			
7C7E	31870	FF	255			
7C7F	31871	FF	255			
7C80	31872	0	0			
7C81	31873	0	0			
7C82	31874	0	0			
7C83	31875	0	0			
7C84	31876	0F	15			
7C85	31877	8	8			
7C86	31878	8	8			
7C87	31879	8	8			
7C88	31880	8	8			
7C89	31881	8	8			
7C8A	31882	8	8			
7C8B	31883	8	8			
7C8C	31884	FF	255			
7C8D	31885	0	0			
7C8E	31886	0	0			
7C8F	31887	0	0			
7C90	31888	0	0			
7C91	31889	0	0			
7C92	31890	0	0			
7C93	31891	0	0			
7C94	31892	FF	255			
7C95	31893	8	8			
7C96	31894	8	8			
7C97	31895	8	8			
7C98	31896	8	8			
7C99	31897	8	8			
7C9A	31898	8	8			
7C9B	31899	8	8			
7C9C	31900	F8	248			
7C9D	31901	8	8			
7C9E	31902	8	8			
7C9F	31903	8	8			
7CA0	31904	C0	192			
7CA1	31905	C0	192			
7CA2	31906	C0	192			
7CA3	31907	C0	192			
7CA4	31908	C0	192			
7CA5	31909	C0	192			
7CA6	31910	C0	192			
7CA7	31911	C0	192			
7CA8	31912	E0	224			
7CA9	31913	E0	224			
7CAA	31914	E0	224			
7CAB	31915	E0	224			
7CAC	31916	E0	224			
7CAD	31917	E0	224			
7CAE	31918	E0	224			
7CAF	31919	E0	224			
7CB0	31920	7	7			
7CB1	31921	7	7			
7CB2	31922	7	7			
7CB3	31923	7	7			
7CB4	31924	7	7			
7CB5	31925	7	7			
7CB6	31926	7	7			
7CB7	31927	7	7			
7CB8	31928	FF	255			
7CB9	31929	FF	255			
7CBA	31930	0	0			
7CBB	31931	0	0			
7CBC	31932	0	0			
7CBD	31933	0	0			
7CBE	31934	0	0			
7CBF	31935	0	0			
7CC0	31936	FF	255			
7CC1	31937	FF	255			
7CC2	31938	FF	255			
7CC3	31939	0	0			
7CC4	31940	0	0			
7CC5	31941	0	0			
7CC6	31942	0	0			
7CC7	31943	0	0			
7CC8	31944	0	0			
7CC9	31945	0	0			
7CCA	31946	0	0			
7CCB	31947	0	0			
7CCC	31948	0	0			
7CCD	31949	FF	255			
7CCE	31950	FF	255			
7CCF	31951	FF	255			
7CD0	31952	1	1			
7CD1	31953	2	2			
7CD2	31954	44	68			
7CD3	31955	48	72			
7CD4	31956	50	80			
7CD5	31957	60	96			
7CD6	31958	40	64			
7CD7	31959	0	0			
7CD8	31960	0	0			
7CD9	31961	0	0			
7CDA	31962	0	0			
7CDB	31963	0	0			

15
GRAPHICAL CHARACTER

16
GRAPHICAL CHARACTER

17
GRAPHICAL CHARACTER

18
GRAPHICAL CHARACTER

19
GRAPHICAL CHARACTER

20
GRAPHICAL CHARACTER

21
GRAPHICAL CHARACTER

22
GRAPHICAL CHARACTER

23
GRAPHICAL CHARACTER

24
GRAPHICAL CHARACTER

25
GRAPHICAL CHARACTER

26
GRAPHICAL CHARACTER

27

7CDC	31964	F0	240			GRAPHICAL CHARACTER
7CDD	31965	F0	240			
7CDE	31966	F0	240			
7CDF	31967	F0	240			
7CE0	31968	0F	15			
7CE1	31969	0F	15			
7CE2	31970	0F	15			
7CE3	31971	0F	15			28 GRAPHICAL CHARACTER
7CE4	31972	0	0			
7CE5	31973	0	0			
7CE6	31974	0	0			
7CE7	31975	0	0			
7CE8	31976	8	8			
7CE9	31977	8	8			
7CEA	31978	8	8			
7CEB	31979	8	8			29 GRAPHICAL CHARACTER
7CEC	31980	F8	248			
7CED	31981	0	0			
7CEE	31982	0	0			
7CEF	31983	0	0			
7CF0	31984	F0	240			
7CF1	31985	F0	240			
7CF2	31986	F0	240			
7CF3	31987	F0	240			30 GRAPHICAL CHARACTER
7CF4	31988	0	0			
7CF5	31989	0	0			
7CF6	31990	0	0			
7CF7	31991	0	0			
7CF8	31992	F0	240			
7CF9	31993	F0	240			
7CFA	31994	F0	240			
7CFB	31995	F0	240			31 GRAPHICAL CHARACTER
7CFC	31996	0F	15			
7CFD	31997	0F	15			
7CFE	31998	0F	15			
7CFF	31999	0F	15			
7D00	32000	0	0			
7D01	32001	0	0			
7D02	32002	0	0			
7D03	32003	0	0			
7D04	32004	0	0			32 SPACE
7D05	32005	0	0			
7D06	32006	0	0			
7D07	32007	0	0			
7D08	32008	8	8			
7D09	32009	8	8			
7D0A	32010	8	8			
7D0B	32011	8	8			33 !
7D0C	32012	0	0			
7D0D	32013	0	0			
7D0E	32014	8	8			
7D0F	32015	0	0			
7D10	32016	24	36			
7D11	32017	24	36			
7D12	32018	24	36			
7D13	32019	0	0			34 "
7D14	32020	0	0			
7D15	32021	0	0			
7D16	32022	0	0			
7D17	32023	0	0			
7D18	32024	24	36			
7D19	32025	24	36			
7D1A	32026	7E	126			
7D1B	32027	24	36			35 #
7D1C	32028	7E	126			
7D1D	32029	24	36			
7D1E	32030	24	36			
7D1F	32031	0	0			
7D20	32032	8	8			
7D21	32033	1E	30			
7D22	32034	28	40			
7D23	32035	1C	28			36 $
7D24	32036	0A	10			
7D25	32037	3C	60			
7D26	32038	8	8			
7D27	32039	0	0			
7D28	32040	0	0			
7D29	32041	62	98			
7D2A	32042	64	100			
7D2B	32043	8	8			37 %
7D2C	32044	10	16			
7D2D	32045	26	38			
7D2E	32046	46	70			
7D2F	32047	0	0			
7D30	32048	30	48			
7D31	32049	48	72			
7D32	32050	48	72			
7D33	32051	30	48			38 &
7D34	32052	4A	74			
7D35	32053	44	68			
7D36	32054	3A	58			
7D37	32055	0	0			
7D38	32056	4	4			
7D39	32057	8	8			
7D3A	32058	10	16			
7D3B	32059	0	0			39 '
7D3C	32060	0	0			
7D3D	32061	0	0			
7D3E	32062	0	0			
7D3F	32063	0	0			
7D40	32064	4	4			
7D41	32065	8	8			
7D42	32066	10	16			

7D43	32067	10	16			40 (
7D44	32068	10	16			
7D45	32069	8	8			
7D46	32070	4	4			
7D47	32071	0	0			
7D48	32072	20	32			41)
7D49	32073	10	16			
7D4A	32074	8	8			
7D4B	32075	8	8			
7D4C	32076	8	8			
7D4D	32077	10	16			
7D4E	32078	20	32			
7D4F	32079	0	0			
7D50	32080	8	8			42 *
7D51	32081	2A	42			
7D52	32082	1C	28			
7D53	32083	3E	62			
7D54	32084	1C	28			
7D55	32085	2A	42			
7D56	32086	8	8			
7D57	32087	0	0			
7D58	32088	0	0			43 +
7D59	32089	8	8			
7D5A	32090	8	8			
7D5B	32091	3E	62			
7D5C	32092	8	8			
7D5D	32093	8	8			
7D5E	32094	0	0			
7D5F	32095	0	0			
7D60	32096	0	0			44 ,
7D61	32097	0	0			
7D62	32098	0	0			
7D63	32099	0	0			
7D64	32100	0	0			
7D65	32101	8	8			
7D66	32102	8	8			
7D67	32103	10	16			
7D68	32104	0	0			45 -
7D69	32105	0	0			
7D6A	32106	0	0			
7D6B	32107	7E	126			
7D6C	32108	0	0			
7D6D	32109	0	0			
7D6E	32110	0	0			
7D6F	32111	0	0			
7D70	32112	0	0			46 .
7D71	32113	0	0			
7D72	32114	0	0			
7D73	32115	0	0			
7D74	32116	0	0			
7D75	32117	18	24			
7D76	32118	18	24			
7D77	32119	0	0			
7D78	32120	0	0			47 /
7D79	32121	2	2			
7D7A	32122	4	4			
7D7B	32123	8	8			
7D7C	32124	10	16			
7D7D	32125	20	32			
7D7E	32126	40	64			
7D7F	32127	0	0			
7D80	32128	3C	60			48 0
7D81	32129	42	66			
7D82	32130	46	70			
7D83	32131	5A	90			
7D84	32132	62	98			
7D85	32133	42	66			
7D86	32134	3C	60			
7D87	32135	0	0			
7D88	32136	8	8			49 1
7D89	32137	18	24			
7D8A	32138	28	40			
7D8B	32139	8	8			
7D8C	32140	8	8			
7D8D	32141	8	8			
7D8E	32142	3E	62			
7D8F	32143	0	0			
7D90	32144	3C	60			50 2
7D91	32145	42	66			
7D92	32146	2	2			
7D93	32147	0C	12			
7D94	32148	30	48			
7D95	32149	40	64			
7D96	32150	7E	126			
7D97	32151	0	0			
7D98	32152	3C	60			51 3
7D99	32153	42	66			
7D9A	32154	2	2			
7D9B	32155	1C	28			
7D9C	32156	2	2			
7D9D	32157	42	66			
7D9E	32158	3C	60			
7D9F	32159	0	0			
7DA0	32160	4	4			52 4
7DA1	32161	0C	12			
7DA2	32162	14	20			
7DA3	32163	24	36			
7DA4	32164	7E	126			
7DA5	32165	4	4			
7DA6	32166	4	4			
7DA7	32167	0	0			
7DA8	32168	7E	126			
7DA9	32169	40	64			

7DAA	32170	78	120			53 5
7DAB	32171	4	4			
7DAC	32172	2	2			
7DAD	32173	44	68			
7DAE	32174	38	56			
7DAF	32175	0	0			
7DB0	32176	1C	28			54 6
7DB1	32177	20	32			
7DB2	32178	40	64			
7DB3	32179	7C	124			
7DB4	32180	42	66			
7DB5	32181	42	66			
7DB6	32182	3C	60			
7DB7	32183	0	0			
7DB8	32184	7E	126			55 7
7DB9	32185	42	66			
7DBA	32186	4	4			
7DBB	32187	8	8			
7DBC	32188	10	16			
7DBD	32189	10	16			
7DBE	32190	10	16			
7DBF	32191	0	0			
7DC0	32192	3C	60			56 8
7DC1	32193	42	66			
7DC2	32194	42	66			
7DC3	32195	3C	60			
7DC4	32196	42	66			
7DC5	32197	42	66			
7DC6	32198	3C	60			
7DC7	32199	0	0			
7DC8	32200	3C	60			57 9
7DC9	32201	42	66			
7DCA	32202	42	66			
7DCB	32203	3E	62			
7DCC	32204	2	2			
7DCD	32205	4	4			
7DCE	32206	38	56			
7DCF	32207	0	0			
7DD0	32208	0	0			58 :
7DD1	32209	0	0			
7DD2	32210	8	8			
7DD3	32211	0	0			
7DD4	32212	0	0			
7DD5	32213	8	8			
7DD6	32214	0	0			
7DD7	32215	0	0			
7DD8	32216	0	0			59 ;
7DD9	32217	0	0			
7DDA	32218	8	8			
7DDB	32219	0	0			
7DDC	32220	0	0			
7DDD	32221	8	8			
7DDE	32222	8	8			
7DDF	32223	10	16			
7DE0	32224	0E	14			60 <
7DE1	32225	18	24			
7DE2	32226	30	48			
7DE3	32227	60	96			
7DE4	32228	30	48			
7DE5	32229	18	24			
7DE6	32230	0E	14			
7DE7	32231	0	0			
7DE8	32232	0	0			61 =
7DE9	32233	0	0			
7DEA	32234	7E	126			
7DEB	32235	0	0			
7DEC	32236	7E	126			
7DED	32237	0	0			
7DEE	32238	0	0			
7DEF	32239	0	0			
7DF0	32240	70	112			62 >
7DF1	32241	18	24			
7DF2	32242	0C	12			
7DF3	32243	6	6			
7DF4	32244	0C	12			
7DF5	32245	18	24			
7DF6	32246	70	112			
7DF7	32247	0	0			
7DF8	32248	3C	60			63 ?
7DF9	32249	42	66			
7DFA	32250	2	2			
7DFB	32251	0C	12			
7DFC	32252	10	16			
7DFD	32253	0	0			
7DFE	32254	10	16			
7DFF	32255	0	0			
7E00	32256	1C	28			64 @
7E01	32257	22	34			
7E02	32258	4A	74			
7E03	32259	56	86			
7E04	32260	4C	76			
7E05	32261	20	32			
7E06	32262	1E	30			
7E07	32263	0	0			
7E08	32264	18	24			65 A
7E09	32265	24	36			
7E0A	32266	42	66			
7E0B	32267	7E	126			
7E0C	32268	42	66			
7E0D	32269	42	66			
7E0E	32270	42	66			
7E0F	32271	0	0			
7E10	32272	7C	124			

7E11	32273	22	34			66 B
7E12	32274	22	34			
7E13	32275	3C	60			
7E14	32276	22	34			
7E15	32277	22	34			
7E16	32278	7C	124			
7E17	32279	0	0			
7E18	32280	1C	28			67 C
7E19	32281	22	34			
7E1A	32282	40	64			
7E1B	32283	40	64			
7E1C	32284	40	64			
7E1D	32285	22	34			
7E1E	32286	1C	28			
7E1F	32287	0	0			
7E20	32288	78	120			68 D
7E21	32289	24	36			
7E22	32290	22	34			
7E23	32291	22	34			
7E24	32292	22	34			
7E25	32293	24	36			
7E26	32294	78	120			
7E27	32295	0	0			
7E28	32296	7E	126			69 E
7E29	32297	40	64			
7E2A	32298	40	64			
7E2B	32299	78	120			
7E2C	32300	40	64			
7E2D	32301	40	64			
7E2E	32302	7E	126			
7E2F	32303	0	0			
7E30	32304	7E	126			70 F
7E31	32305	40	64			
7E32	32306	40	64			
7E33	32307	78	120			
7E34	32308	40	64			
7E35	32309	40	64			
7E36	32310	40	64			
7E37	32311	0	0			
7E38	32312	1C	28			71 G
7E39	32313	22	34			
7E3A	32314	40	64			
7E3B	32315	4E	78			
7E3C	32316	42	66			
7E3D	32317	22	34			
7E3E	32318	1C	28			
7E3F	32319	0	0			
7E40	32320	42	66			72 H
7E41	32321	42	66			
7E42	32322	42	66			
7E43	32323	7E	126			
7E44	32324	42	66			
7E45	32325	42	66			
7E46	32326	42	66			
7E47	32327	0	0			
7E48	32328	1C	28			73 I
7E49	32329	8	8			
7E4A	32330	8	8			
7E4B	32331	8	8			
7E4C	32332	8	8			
7E4D	32333	8	8			
7E4E	32334	1C	28			
7E4F	32335	0	0			
7E50	32336	0E	14			74 J
7E51	32337	4	4			
7E52	32338	4	4			
7E53	32339	4	4			
7E54	32340	4	4			
7E55	32341	44	68			
7E56	32342	38	56			
7E57	32343	0	0			
7E58	32344	42	66			75 K
7E59	32345	44	68			
7E5A	32346	48	72			
7E5B	32347	70	112			
7E5C	32348	48	72			
7E5D	32349	44	68			
7E5E	32350	42	66			
7E5F	32351	0	0			
7E60	32352	40	64			76 L
7E61	32353	40	64			
7E62	32354	40	64			
7E63	32355	40	64			
7E64	32356	40	64			
7E65	32357	40	64			
7E66	32358	7E	126			
7E67	32359	0	0			
7E68	32360	42	66			77 M
7E69	32361	66	102			
7E6A	32362	5A	90			
7E6B	32363	5A	90			
7E6C	32364	42	66			
7E6D	32365	42	66			
7E6E	32366	42	66			
7E6F	32367	0	0			
7E70	32368	42	66			78 N
7E71	32369	62	98			
7E72	32370	52	82			
7E73	32371	4A	74			
7E74	32372	46	70			
7E75	32373	42	66			
7E76	32374	42	66			
7E77	32375	0	0			

7E78	32376	18	24			79 O
7E79	32377	24	36			
7E7A	32378	42	66			
7E7B	32379	42	66			
7E7C	32380	42	66			
7E7D	32381	24	36			
7E7E	32382	18	24			
7E7F	32383	0	0			
7E80	32384	7C	124			80 P
7E81	32385	42	66			
7E82	32386	42	66			
7E83	32387	7C	124			
7E84	32388	40	64			
7E85	32389	40	64			
7E86	32390	40	64			
7E87	32391	0	0			
7E88	32392	18	24			81 Q
7E89	32393	24	36			
7E8A	32394	42	66			
7E8B	32395	42	66			
7E8C	32396	4A	74			
7E8D	32397	24	36			
7E8E	32398	1A	26			
7E8F	32399	0	0			
7E90	32400	7C	124			82 R
7E91	32401	42	66			
7E92	32402	42	66			
7E93	32403	7C	124			
7E94	32404	48	72			
7E95	32405	44	68			
7E96	32406	42	66			
7E97	32407	0	0			
7E98	32408	3C	60			83 S
7E99	32409	42	66			
7E9A	32410	40	64			
7E9B	32411	3C	60			
7E9C	32412	2	2			
7E9D	32413	42	66			
7E9E	32414	3C	60			
7E9F	32415	0	0			
7EA0	32416	3E	62			84 T
7EA1	32417	8	8			
7EA2	32418	8	8			
7EA3	32419	8	8			
7EA4	32420	8	8			
7EA5	32421	8	8			
7EA6	32422	8	8			
7EA7	32423	0	0			
7EA8	32424	42	66			85 U
7EA9	32425	42	66			
7EAA	32426	42	66			
7EAB	32427	42	66			
7EAC	32428	42	66			
7EAD	32429	42	66			
7EAE	32430	3C	60			
7EAF	32431	0	0			
7EB0	32432	42	66			86 V
7EB1	32433	42	66			
7EB2	32434	42	66			
7EB3	32435	24	36			
7EB4	32436	24	36			
7EB5	32437	18	24			
7EB6	32438	18	24			
7EB7	32439	0	0			
7EB8	32440	42	66			87 W
7EB9	32441	42	66			
7EBA	32442	42	66			
7EBB	32443	5A	90			
7EBC	32444	5A	90			
7EBD	32445	66	102			
7EBE	32446	42	66			
7EBF	32447	0	0			
7EC0	32448	42	66			88 X
7EC1	32449	42	66			
7EC2	32450	24	36			
7EC3	32451	18	24			
7EC4	32452	24	36			
7EC5	32453	42	66			
7EC6	32454	42	66			
7EC7	32455	0	0			
7EC8	32456	22	34			89 Y
7EC9	32457	22	34			
7ECA	32458	22	34			
7ECB	32459	1C	28			
7ECC	32460	8	8			
7ECD	32461	8	8			
7ECE	32462	8	8			
7ECF	32463	0	0			
7ED0	32464	7E	126			90 Z
7ED1	32465	2	2			
7ED2	32466	4	4			
7ED3	32467	18	24			
7ED4	32468	20	32			
7ED5	32469	40	64			
7ED6	32470	7E	126			
7ED7	32471	0	0			
7ED8	32472	3C	60			91 [
7ED9	32473	20	32			
7EDA	32474	20	32			
7EDB	32475	20	32			
7EDC	32476	20	32			
7EDD	32477	20	32			
7EDE	32478	3C	60			

7EDF	32479	0	0			
7EE0	32480	0	0			
7EE1	32481	40	64			
7EE2	32482	20	32			
7EE3	32483	10	16			
7EE4	32484	8	8			
7EE5	32485	4	4			
7EE6	32486	2	2			
7EE7	32487	0	0			
7EE8	32488	3C	60			
7EE9	32489	4	4			
7EEA	32490	4	4			
7EEB	32491	4	4			
7EEC	32492	4	4			
7EED	32493	4	4			
7EEE	32494	3C	60			
7EEF	32495	0	0			
7EF0	32496	0	0			
7EF1	32497	8	8			
7EF2	32498	1C	28			
7EF3	32499	2A	42			
7EF4	32500	8	8			
7EF5	32501	8	8			
7EF6	32502	8	8			
7EF7	32503	8	8			
7EF8	32504	0	0			
7EF9	32505	0	0			
7EFA	32506	10	16			
7EFB	32507	20	32			
7EFC	32508	7F	127			
7EFD	32509	20	32			
7EFE	32510	10	16			
7EFF	32511	0	0			
7F00	32512	0	0			
7F01	32513	0	0			
7F02	32514	0	0			
7F03	32515	0	0			
7F04	32516	FF	255			
7F05	32517	0	0			
7F06	32518	0	0			
7F07	32519	0	0			
7F08	32520	0	0			
7F09	32521	0	0			
7F0A	32522	38	56			
7F0B	32523	4	4			
7F0C	32524	3C	60			
7F0D	32525	44	68			
7F0E	32526	3A	58			
7F0F	32527	0	0			
7F10	32528	40	64			
7F11	32529	40	64			
7F12	32530	5C	92			
7F13	32531	62	98			
7F14	32532	42	66			
7F15	32533	62	98			
7F16	32534	5C	92			
7F17	32535	0	0			
7F18	32536	0	0			
7F19	32537	0	0			
7F1A	32538	3C	60			
7F1B	32539	42	66			
7F1C	32540	40	64			
7F1D	32541	42	66			
7F1E	32542	3C	60			
7F1F	32543	0	0			
7F20	32544	2	2			
7F21	32545	2	2			
7F22	32546	3A	58			
7F23	32547	46	70			
7F24	32548	42	66			
7F25	32549	46	70			
7F26	32550	3A	58			
7F27	32551	0	0			
7F28	32552	0	0			
7F29	32553	0	0			
7F2A	32554	3C	60			
7F2B	32555	42	66			
7F2C	32556	7E	126			
7F2D	32557	40	64			
7F2E	32558	3C	60			
7F2F	32559	0	0			
7F30	32560	0C	12			
7F31	32561	12	18			
7F32	32562	10	16			
7F33	32563	7C	124			
7F34	32564	10	16			
7F35	32565	10	16			
7F36	32566	10	16			
7F37	32567	0	0			
7F38	32568	0	0			
7F39	32569	0	0			
7F3A	32570	3A	58			
7F3B	32571	46	70			
7F3C	32572	46	70			
7F3D	32573	3A	58			
7F3E	32574	2	2			
7F3F	32575	3C	60			
7F40	32576	40	64			
7F41	32577	40	64			
7F42	32578	5C	92			
7F43	32579	62	98			
7F44	32580	42	66			
7F45	32581	42	66			

92
\

93
]

94
UP ARROW

95
LEFT ARROW

96
DASH

97
a

98
b

99
c

100
d

101
e

102
f

103
g

104
h

7F46	32582	42	66			
7F47	32583	0	0			
7F48	32584	8	8			
7F49	32585	0	0			
7F4A	32586	18	24			
7F4B	32587	8	8			105 i
7F4C	32588	8	8			
7F4D	32589	8	8			
7F4E	32590	1C	28			
7F4F	32591	0	0			
7F50	32592	4	4			
7F51	32593	0	0			
7F52	32594	0C	12			
7F53	32595	4	4			106 j
7F54	32596	4	4			
7F55	32597	4	4			
7F56	32598	44	68			
7F57	32599	38	56			
7F58	32600	40	64			
7F59	32601	40	64			
7F5A	32602	44	68			107 k
7F5B	32603	48	72			
7F5C	32604	50	80			
7F5D	32605	68	104			
7F5E	32606	44	68			
7F5F	32607	0	0			
7F60	32608	18	24			
7F61	32609	8	8			
7F62	32610	8	8			108 l
7F63	32611	8	8			
7F64	32612	8	8			
7F65	32613	8	8			
7F66	32614	1C	28			
7F67	32615	0	0			
7F68	32616	0	0			
7F69	32617	0	0			
7F6A	32618	76	118			109 m
7F6B	32619	49	73			
7F6C	32620	49	73			
7F6D	32621	49	73			
7F6E	32622	49	73			
7F6F	32623	0	0			
7F70	32624	0	0			
7F71	32625	0	0			
7F72	32626	5C	92			110 n
7F73	32627	62	98			
7F74	32628	42	66			
7F75	32629	42	66			
7F76	32630	42	66			
7F77	32631	0	0			
7F78	32632	0	0			
7F79	32633	0	0			
7F7A	32634	3C	60			111 o
7F7B	32635	42	66			
7F7C	32636	42	66			
7F7D	32637	42	66			
7F7E	32638	3C	60			
7F7F	32639	0	0			
7F80	32640	0	0			
7F81	32641	0	0			
7F82	32642	5C	92			112 p
7F83	32643	62	98			
7F84	32644	62	98			
7F85	32645	5C	92			
7F86	32646	40	64			
7F87	32647	40	64			
7F88	32648	0	0			
7F89	32649	0	0			
7F8A	32650	3A	58			113 q
7F8B	32651	46	70			
7F8C	32652	46	70			
7F8D	32653	3A	58			
7F8E	32654	2	2			
7F8F	32655	2	2			
7F90	32656	0	0			
7F91	32657	0	0			
7F92	32658	5C	92			114 r
7F93	32659	62	98			
7F94	32660	40	64			
7F95	32661	40	64			
7F96	32662	40	64			
7F97	32663	0	0			
7F98	32664	0	0			
7F99	32665	0	0			
7F9A	32666	3E	62			115 s
7F9B	32667	40	64			
7F9C	32668	3C	60			
7F9D	32669	2	2			
7F9E	32670	7C	124			
7F9F	32671	0	0			
7FA0	32672	10	16			
7FA1	32673	10	16			
7FA2	32674	7C	124			116 t
7FA3	32675	10	16			
7FA4	32676	10	16			
7FA5	32677	12	18			
7FA6	32678	0C	12			
7FA7	32679	0	0			
7FA8	32680	0	0			
7FA9	32681	0	0			
7FAA	32682	42	66			
7FAB	32683	42	66			117 u
7FAC	32684	42	66			

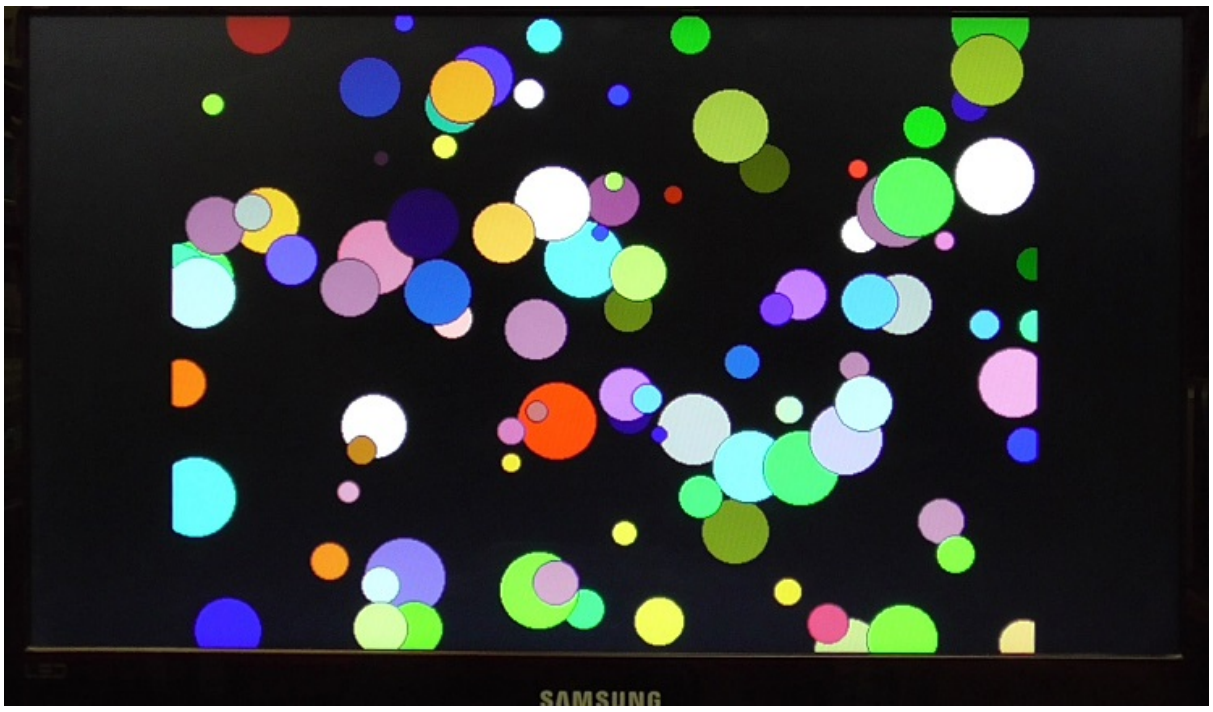
7FAD	32685	46	70			
7FAE	32686	3A	58			
7FAF	32687	0	0			
7FB0	32688	0	0			118 v
7FB1	32689	0	0			
7FB2	32690	42	66			
7FB3	32691	42	66			
7FB4	32692	42	66			
7FB5	32693	24	36			
7FB6	32694	18	24			
7FB7	32695	0	0			
7FB8	32696	0	0			119 w
7FB9	32697	0	0			
7FBA	32698	41	65			
7FBB	32699	49	73			
7FBC	32700	49	73			
7FBD	32701	49	73			
7FBE	32702	36	54			
7FBF	32703	0	0			
7FC0	32704	0	0			120 x
7FC1	32705	0	0			
7FC2	32706	42	66			
7FC3	32707	24	36			
7FC4	32708	18	24			
7FC5	32709	24	36			
7FC6	32710	42	66			
7FC7	32711	0	0			
7FC8	32712	0	0			121 y
7FC9	32713	0	0			
7FCA	32714	42	66			
7FCB	32715	42	66			
7FCC	32716	46	70			
7FCD	32717	3A	58			
7FCE	32718	2	2			
7FCF	32719	3C	60			
7FD0	32720	0	0			122 z
7FD1	32721	0	0			
7FD2	32722	7E	126			
7FD3	32723	4	4			
7FD4	32724	18	24			
7FD5	32725	20	32			
7FD6	32726	7E	126			
7FD7	32727	0	0			
7FD8	32728	8	8			123 GRAPHICAL CHARACTER
7FD9	32729	8	8			
7FDA	32730	8	8			
7FDB	32731	8	8			
7FDC	32732	FF	255			
7FDD	32733	8	8			
7FDE	32734	8	8			
7FDF	32735	8	8			
7FE0	32736	A0	160			124 GRAPHICAL CHARACTER
7FE1	32737	50	80			
7FE2	32738	A0	160			
7FE3	32739	50	80			
7FE4	32740	A0	160			
7FE5	32741	50	80			
7FE6	32742	A0	160			
7FE7	32743	50	80			
7FE8	32744	8	8			125 GRAPHICAL CHARACTER
7FE9	32745	8	8			
7FEA	32746	8	8			
7FEB	32747	8	8			
7FEC	32748	8	8			
7FED	32749	8	8			
7FEE	32750	8	8			
7FEF	32751	8	8			
7FF0	32752	CC	204			126 GRAPHICAL CHARACTER
7FF1	32753	CC	204			
7FF2	32754	33	51			
7FF3	32755	33	51			
7FF4	32756	CC	204			
7FF5	32757	CC	204			
7FF6	32758	33	51			
7FF7	32759	33	51			
7FF8	32760	CC	204			127 GRAPHICAL CHARACTER
7FF9	32761	66	102			
7FFA	32762	33	51			
7FFB	32763	99	153			
7FFC	32764	CC	204			
7FFD	32765	66	102			
7FFE	32766	33	51			
7FFF	32767	99	153			

BASIC example programs

99 Luft Balloons

Very loosely inspired by an earworm that doesn't make a whole lot of sense, this program draws, to the screen, 99 circles of random position, colour and diameter. This short program demonstrates the use of the circle drawing routines. If we compute the product of the range of all four RND statements, we find that there are 503,193,600 completed display possibilities. Explore them all!

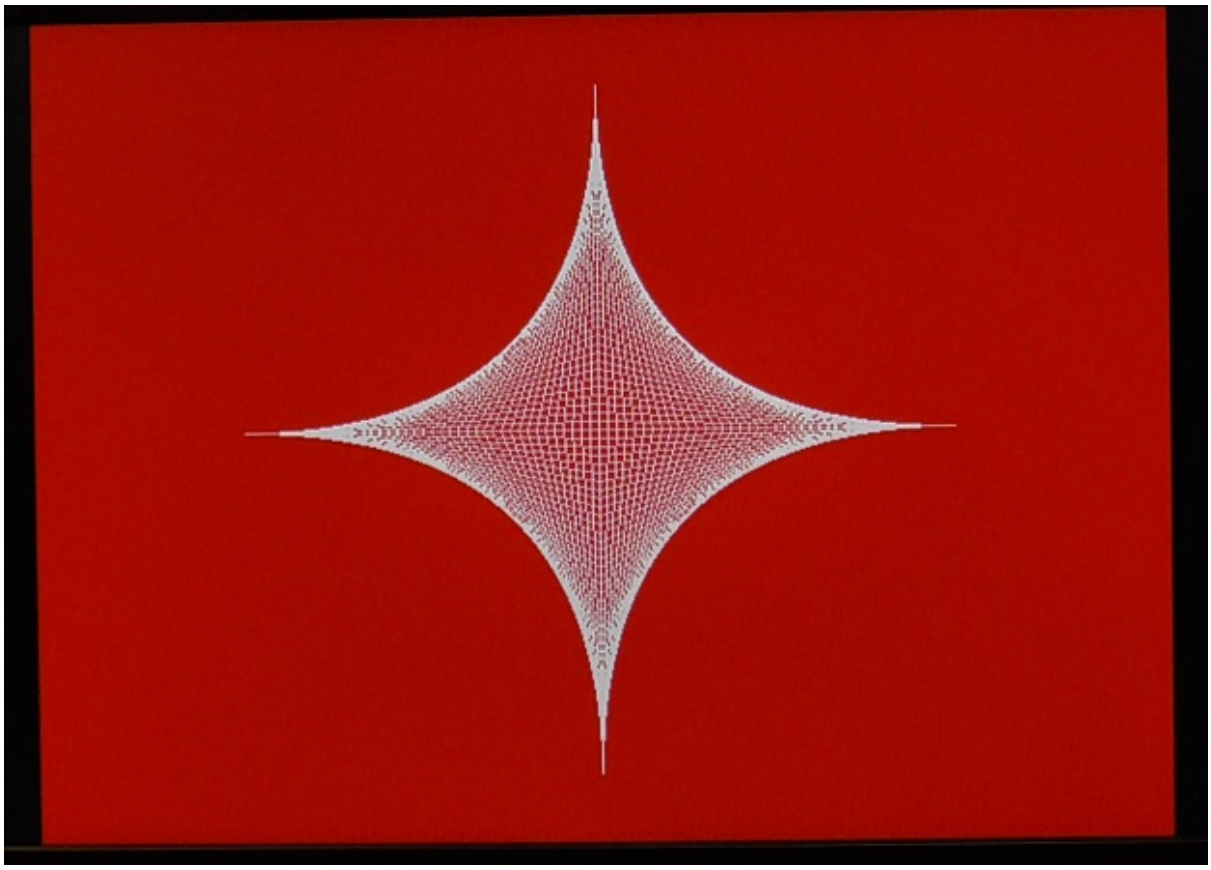
```
1000 REM -99 LUFT BALLOONS FOR TRS-80 LEVEL II BASIC
1010 REM -REQUIRES STAGE 1 MACHINE CODE GRAPHICS ROUTINES FOR
1020 REM -VGA VIDEO CARD B. WWW.GLENSSTUFF.COM
1030 POKE 28686,255 : COL=28673 : POKE 16527,112 : I=16526
1040 PX=110 : PY=117 : CA=197 : CF=194
1050 POKE I,64 : A=USR(0) : REM -CLEAR SCREEN TO BLACK
1060 FOR Q=1 TO 99: RAD=RND(25)+5
1070 POKE I,PX: A=USR(RND(639)+1)
1080 POKE I,PY: A=USR(RND(479))
1090 POKE I,CF: A=USR(RAD): REM -DRAW FILLED CIRCLE
1100 POKE COL,0: REM -COLOUR TO BLACK
1110 POKE I,CA: A=USR(RAD): REM -DRAW OUTLINE
1120 POKE COL,RND(62)+1: REM -RANDOM COLOUR FOR NEXT CIRCLE
1130 NEXT Q
READY
>_
```



Moiré Pattern Star

This program makes use of the **draw line (X1, Y1)-(X2, Y2)** routine.

```
1000 REM -MOIRE PATTERN STAR FOR TRS-80 LEVEL II BASIC
1010 REM -REQUIRES STAGE 1 MACHINE CODE GRAPHICS ROUTINES FOR
1020 REM -VGA VIDEO CARD B. WWW.GLENSSTUFF.COM
1030 POKE 28673,63 : POKE 16527,112 : I=16526
1040 L1=200 : X1=124 : Y1=131 : X2=138 : Y2=145
1050 POKE I,64 : A=USR(3) : REM -CLEAR SCREEN TO RED
1060 POKE I,Y1 : A=USR(240) : POKE I,X2 : A=USR(320)
1070 FOR Q=0 TO 200 STEP 5
1080 POKE I,X1: A=USR(320-Q)
1090 POKE I,Y2: A=USR(40+Q): POKE I,L1: A=USR(0)
1100 POKE I,Y2: A=USR(440-Q): POKE I,L1: A=USR(0)
1110 POKE I,X1: A=USR(320+Q): POKE I,L1: A=USR(0)
1120 POKE I,Y2: A=USR(40+Q): POKE I,L1: A=USR(0)
1130 NEXT Q
READY
>_
```



Lorenz Attractor

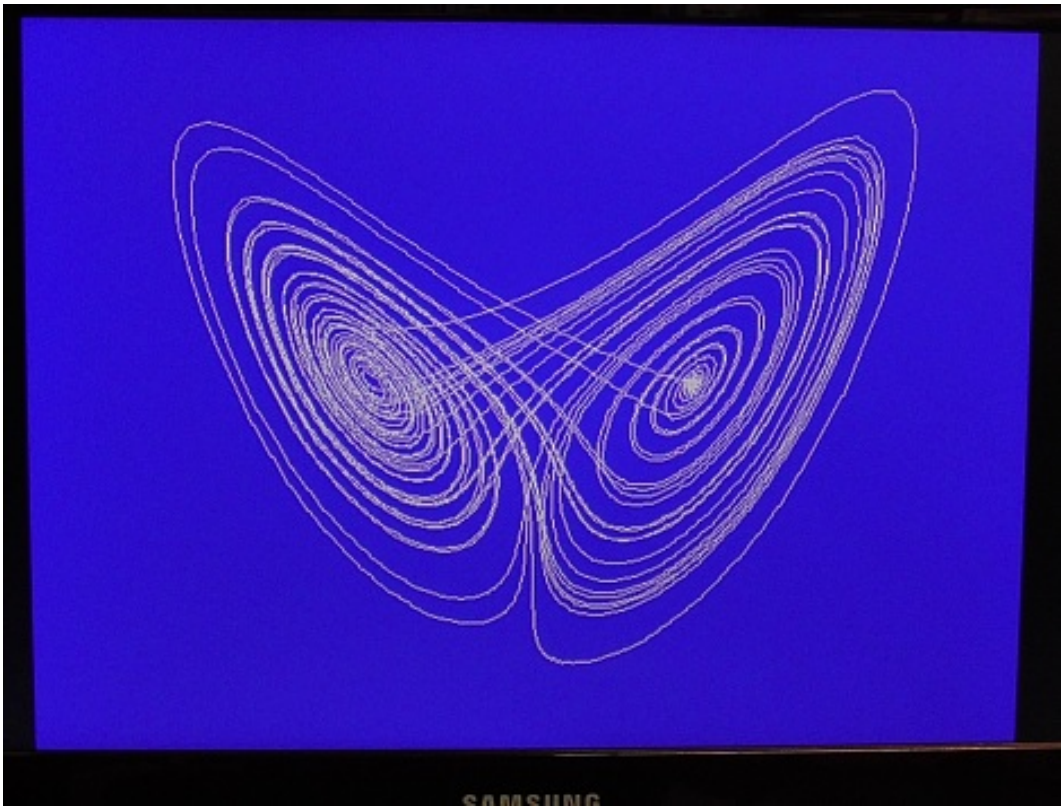
This program computes the three coupled ordinary differential equations of the famous Lorenz Attractor. These equations describe a system with a chaotic solution and their discovery in 1963 really opened up the field of chaos theory. Hardly anyone these days hasn't at least heard of the "butterfly effect".

The integration algorithm shown here is the only one I have ever programmed and is probably the most primitive that can be devised! This program provides a good demonstration of using **draw line –(X2, Y2)** to join sequentially computed points.

The main loop is infinite and the attractor will just keep drawing until you halt/exit the program by pressing the BREAK key. This is some heavy math for BASIC on a home computer more than 40 years old, though I do get some satisfaction from successfully running such problems on such elementary hardware!

The screen photo of the plotted attractor presented here was taken at approximately the ten-minute mark of run time.

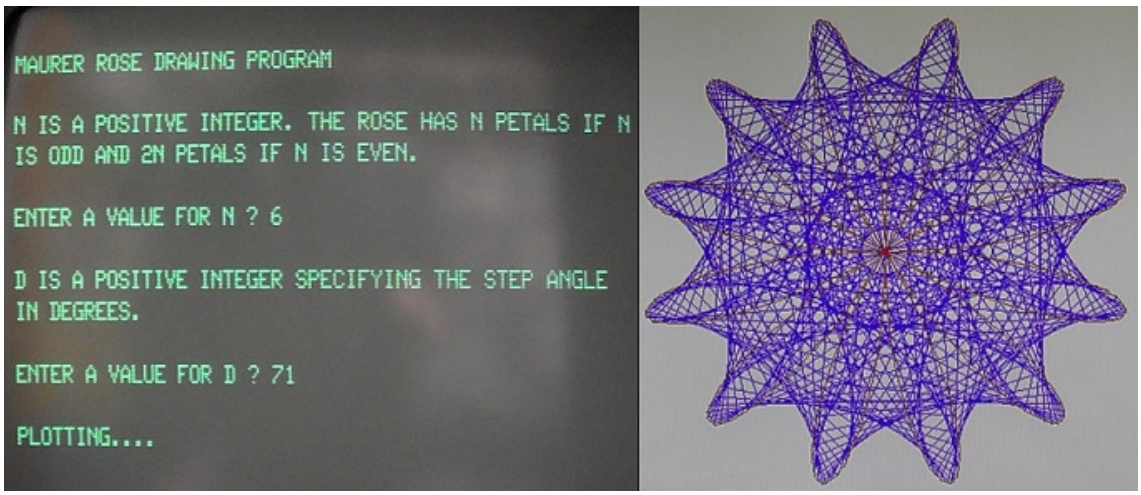
```
1000 REM -LORENZ ATTRACTOR FOR TRS-80 LEVEL II BASIC
1010 REM -REQUIRES STAGE 1 MACHINE CODE GRAPHICS ROUTINES FOR
1020 REM -VGA VIDEO CARD B. WWW.GLENSSTUFF.COM
1030 POKE 16527,112: I=16526: X1=124: Y1=131: X2=138: Y2=145
1040 L2=203: POKE 28673,63: POKE I,64: A=USR(48)
1050 ST=0: C=0.01: S=10: R=28: B=2.67: X=1: Y=0: Z=20
1060 REM -COMPUTE, SCALE AND PLOT ATTRACTOR TO VGA SCREEN
1070 DX=(-S*X)+(S*Y): DY=(R*X)-Y-(X*Z): DZ=(-B*Z)+(X*Y)
1080 X=X+(DX*C): Y=Y+(DY*C): Z=Z+(DZ*C)
1090 PX=(X*12)+320: PY=(-Z*8)+450: IF ST=1 THEN 1110
1100 POKE I,X1: A=USR(PX): POKE I,Y1: A=USR(PY): ST=1
1110 POKE I,X2: A=USR(PX): POKE I,Y2: A=USR(PY)
1120 POKE I,L2: A=USR(0): REM -CALL DRAWLINE -(X2,Y2)
1130 GOTO 1070 : REM -NEXT ITERATION
READY
>_
```

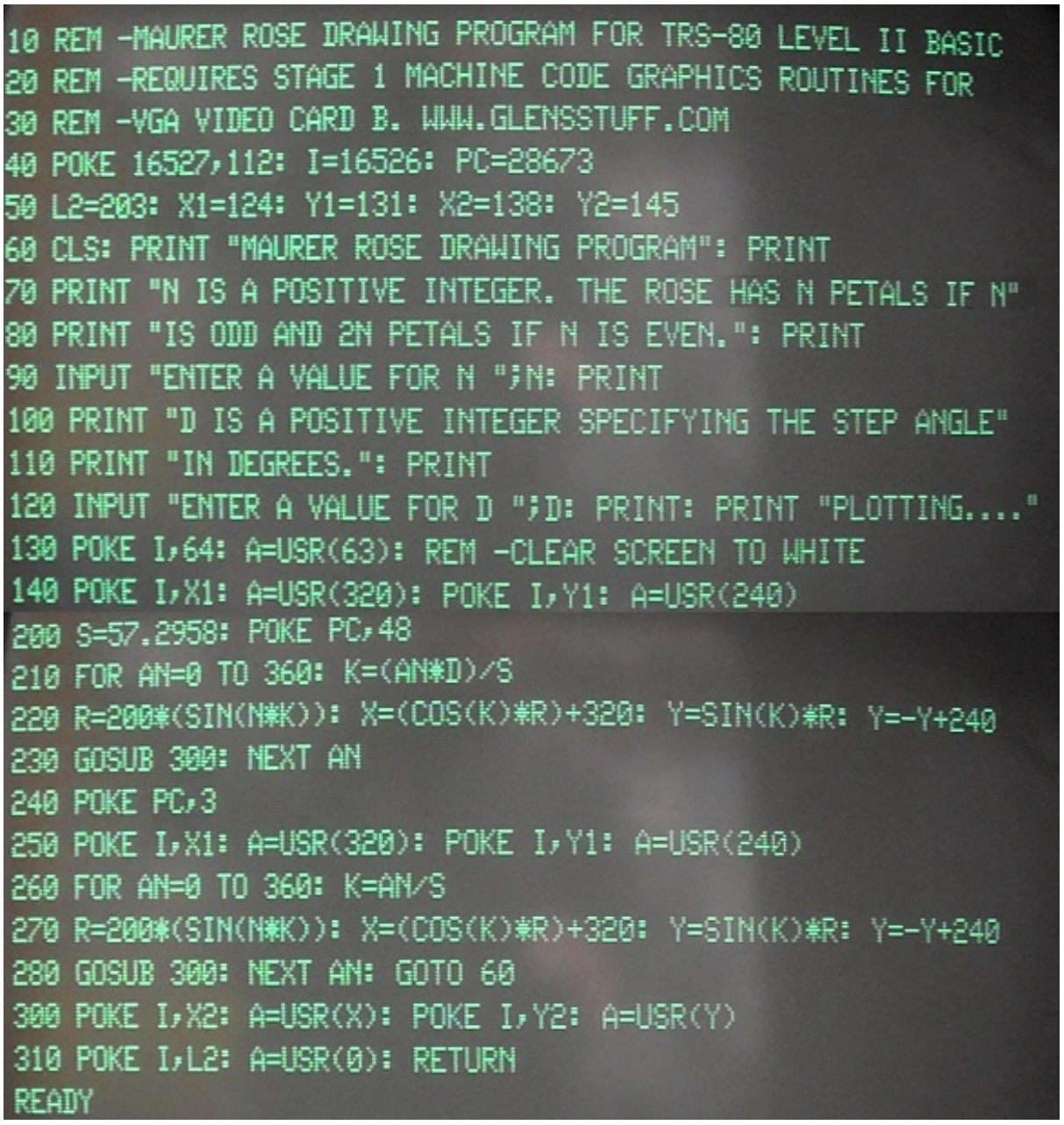


Maurer Rose Drawing Program

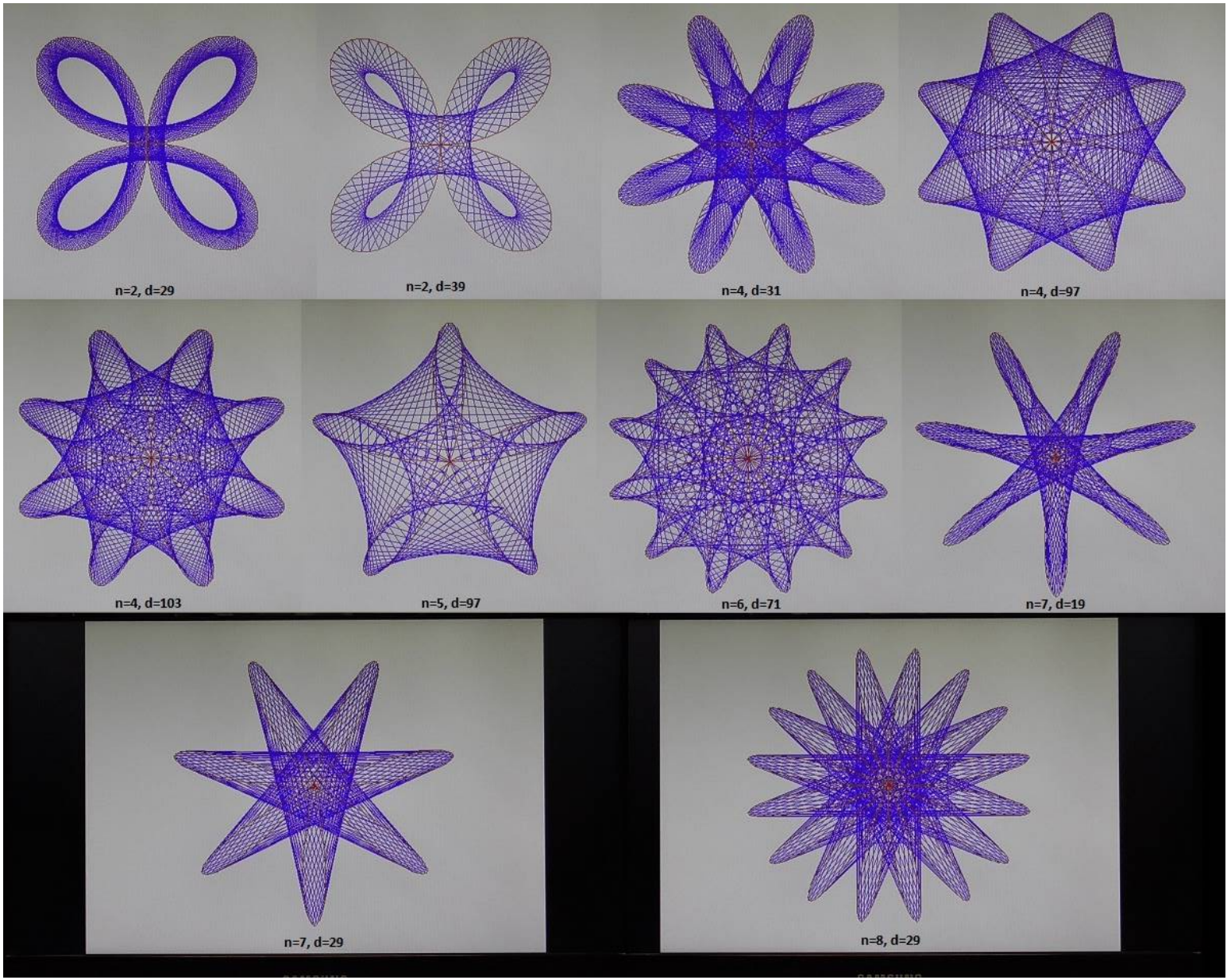
Wikipedia has a good entry on the Maurer rose. It's basically formed by a bunch of lines joining points on a rose curve plotted in polar coordinates. A Spirograph-like curiosity of geometry aside, I'm not too sure of the practical application, but some of the possible forms sure do look pretty!

These roses were photographed directly off my Samsung flat-screen monitor. The background colour is actually white, but my camera can't handle that for some reason and insists on rendering it in a light shade of grey. The text entries giving the n and d variables which define each rose pattern aren't part of the video display; I added those to each image in MS Paint for the purpose of documentation. Each rose has a radius of 200 pixels and takes an average of approximately two minutes to be fully computed and plotted.





```
10 REM -MAURER ROSE DRAWING PROGRAM FOR TRS-80 LEVEL II BASIC
20 REM -REQUIRES STAGE 1 MACHINE CODE GRAPHICS ROUTINES FOR
30 REM -VGA VIDEO CARD B. WWW.GLENSSTUFF.COM
40 POKE 16527,112: I=16526: PC=28673
50 L2=203: X1=124: Y1=131: X2=138: Y2=145
60 CLS: PRINT "MAURER ROSE DRAWING PROGRAM": PRINT
70 PRINT "N IS A POSITIVE INTEGER. THE ROSE HAS N PETALS IF N"
80 PRINT "IS ODD AND 2N PETALS IF N IS EVEN.": PRINT
90 INPUT "ENTER A VALUE FOR N ";N: PRINT
100 PRINT "D IS A POSITIVE INTEGER SPECIFYING THE STEP ANGLE"
110 PRINT "IN DEGREES.": PRINT
120 INPUT "ENTER A VALUE FOR D ";D: PRINT: PRINT "PLOTTING...."
130 POKE I,64: A=USR(63): REM -CLEAR SCREEN TO WHITE
140 POKE I,X1: A=USR(320): POKE I,Y1: A=USR(240)
200 S=57.2958: POKE PC,48
210 FOR AN=0 TO 360: K=(AN*D)/S
220 R=200*(SIN(N*K)): X=(COS(K)*R)+320: Y=SIN(K)*R: Y=-Y+240
230 GOSUB 300: NEXT AN
240 POKE PC,3
250 POKE I,X1: A=USR(320): POKE I,Y1: A=USR(240)
260 FOR AN=0 TO 360: K=AN/S
270 R=200*(SIN(N*K)): X=(COS(K)*R)+320: Y=SIN(K)*R: Y=-Y+240
280 GOSUB 300: NEXT AN: GOTO 60
300 POKE I,X2: A=USR(X): POKE I,Y2: A=USR(Y)
310 POKE I,L2: A=USR(0): RETURN
READY
```


n=2, d=29
n=2, d=39
n=4, d=31
n=4, d=97
n=4, d=103
n=5, d=97
n=6, d=71
n=7, d=19
n=7, d=29
n=8, d=29

Series I Land Rover

Here is a program to honour a classic of British automotive engineering. This BASIC listing makes full use of my machine code drawing routines.

```
100 REM -SERIES 1 LAND ROVER FOR TRS-80 LEVEL II BASIC
110 REM -REQUIRES STAGE 1 MACHINE CODE GRAPHICS ROUTINES FOR
120 REM -VGA VIDEO CARD B. WWW.GLENSSTUFF.COM
130 POKE 16527,112 : I=16526 : PC=28673 : COL=28687 : ROW=28689
140 PX=110 : PY=117 : X1=124 : Y1=131 : X2=138 : Y2=145
150 L1=200 : L2=203 : CA=197 : CF=194
160 RE=191 : RF=188 : BP=182 : TXT=179 : SP=206 : SEG=28686
170 POKE I,64 : A=USR(0) : REM -CLEAR SCREEN TO BLACK
180 READ A: REM -MAIN LOOP STARTS HERE
190 IF A=800 THEN 400: REM -DRAWLINE(X1,Y1)-(X2,Y2)
200 IF A=801 THEN 430: REM -MOVE POINT
210 IF A=802 THEN 450: REM -DRAWLINE-(X2,Y2)
220 IF A=803 THEN 470: REM -CIRCLE/ARC
230 IF A=804 THEN 490: REM -FILLED CIRCLE
240 IF A=805 THEN 510: REM -PLOT COLOUR
250 IF A=806 THEN 520: REM -RECTANGLE
260 IF A=807 THEN 550: REM -FILLED RECTANGLE
270 IF A=808 THEN 580: REM -BUCKET POUR
280 IF A=809 THEN 600: REM -MOVE TEXT CURSOR
290 IF A=810 THEN 610: REM -TEXT CHARACTER. 811 = EXIT CODE
300 IF A=812 THEN 640: REM -SET PIXEL
310 IF A=813 THEN END
400 READ A: POKE I,X1: A=USR(A): READ A: POKE I,Y1: A=USR(A)
410 READ A: POKE I,X2: A=USR(A): READ A: POKE I,Y2: A=USR(A)
420 POKE I,L1: A=USR(0): GOTO 180
430 READ A: POKE I,X1: A=USR(A): READ A: POKE I,Y1: A=USR(A)
440 GOTO 180
450 READ A: POKE I,X2: A=USR(A): READ A: POKE I,Y2: A=USR(A)
460 POKE I,L2: A=USR(0): GOTO 180
470 READ A: POKE I,PX: A=USR(A): READ A: POKE I,PY: A=USR(A)
480 READ A: POKE SEG,A: READ A: POKE I,CA: A=USR(A): GOTO 180
490 READ A: POKE I,PX: A=USR(A): READ A: POKE I,PY: A=USR(A)
500 READ A: POKE SEG,A: READ A: POKE I,CF: A=USR(A): GOTO 180
```

Continues on next page….

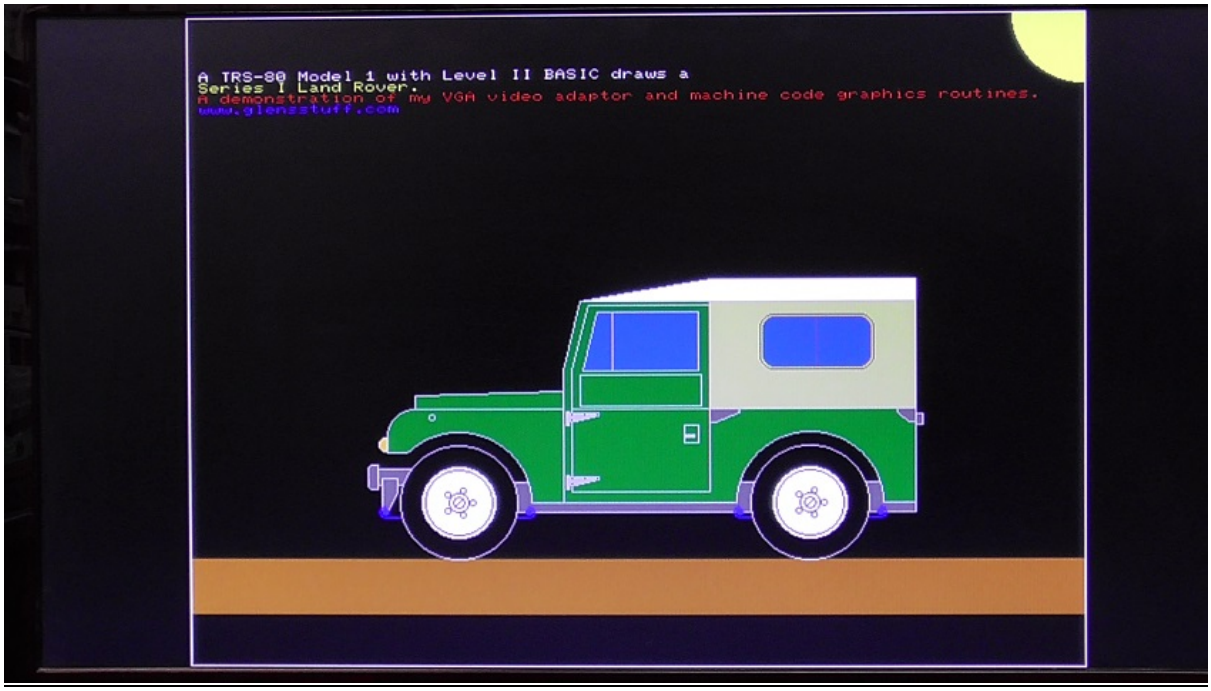
```
510 READ A: POKE PC,A: GOTO 180
520 READ A: POKE I,X1: A=USR(A): READ A: POKE I,Y1: A=USR(A)
530 READ A: POKE I,X2: A=USR(A): READ A: POKE I,Y2: A=USR(A)
540 POKE I,RE: A=USR(0): GOTO 180
550 READ A: POKE I,X1: A=USR(A): READ A: POKE I,Y1: A=USR(A)
560 READ A: POKE I,X2: A=USR(A): READ A: POKE I,Y2: A=USR(A)
570 POKE I,RF: A=USR(0): GOTO 180
580 READ A: POKE I,PX: A=USR(A): READ A: POKE I,PY: A=USR(A)
590 READ A: POKE I,BP: A=USR(A): GOTO 180
600 READ A: POKE COL,A: READ A: POKE ROW,A: GOTO 180
610 POKE I,TXT
620 READ A: IF A=811 THEN 180
630 A=USR(A): GOTO 620
640 READ A: POKE I,PX: A=USR(A): READ A: POKE I,PY: A=USR(A)
650 POKE I,SP: A=USR(0): GOTO 180
990 REM -TEXT MESSAGE
1000 DATA 805,63,809,1,5
1010 DATA 810,65,32,84,82,83,45,56,48,32,77,111,100,101,108,32
1020 DATA 49,32,119,105,116,104,32,76,101,118,101,108,32,73,73
1030 DATA 32,66,65,83,73,67,32,100,114,97,119,115,32,97,811
1040 DATA 809,1,6,805,15
1050 DATA 810,83,101,114,105,101,115,32,73,32,76,97,110,100
1060 DATA 32,82,111,118,101,114,46,811
1070 DATA 809,1,7,805,3
1080 DATA 810,65,32,100,101,109,111,110,115,116,114,97,116,105
1090 DATA 111,110,32,111,102,32,109,121,32,86,71,65,32,118,105
1100 DATA 100,101,111,32,97,100,97,112,116,111,114,32,97,110
1110 DATA 100,32,109,97,99,104,105,110,101,32,99,111,100,101,32
1120 DATA 103,114,97,112,104,105,99,115,32,114,111,117,116,105
1130 DATA 110,101,115,46,811
1140 DATA 809,1,8,805,48
1150 DATA 810,119,119,119,46,103,108,101,110,115,115,116,117
1160 DATA 102,102,46,99,111,109,811
```

Continues on next page….

```
1170 REM -BORDER AND GROUND
1180 DATA 805,63,806,1,0,640,479,805,6,807,2,400,639,440
1190 REM -CHASSIS AND SUSPENSION
1200 DATA 805,21,807,195,343,520,365,807,130,333,195,345
1210 DATA 805,42,806,195,343,520,365,806,130,333,195,345
1220 DATA 805,21,807,140,340,144,365,807,144,340,151,350
1230 DATA 805,42,801,140,340,802,151,340,802,151,350
1240 DATA 802,144,365,802,140,365,802,140,340,805,21
1250 DATA 808,145,351,1,805,48,804,142,365,15,2,804,245,365
1260 DATA 15,2,803,142,365,60,4,803,245,365,195,4,800,142,369
1270 DATA 245,369,804,395,365,15,2,804,495,365,15,2
1280 DATA 803,395,365,60,4,803,495,365,195,4
1290 DATA 800,395,369,495,369
1300 REM -WHEELS
1310 DATA 805,0,804,195,358,15,42,804,445,358,15,42
1320 DATA 805,42,803,195,358,255,42,803,445,358,255,42
1330 DATA 805,63,804,195,358,15,26,804,445,358,15,26
1340 DATA 805,0,803,195,358,255,24,803,445,358,255,24
1350 DATA 803,195,358,255,7,803,445,358,255,7
1360 DATA 803,195,358,255,4,803,445,358,255,4
1370 DATA 803,205,358,255,2,803,198,349,255,2,803,187,352,255,2
1380 DATA 803,187,364,255,2,803,198,367,255,2
1390 DATA 803,455,358,255,2,803,448,349,255,2,803,437,352,255,2
1400 DATA 803,437,364,255,2,803,448,367,255,2
1410 REM -BODY OUTLINE
1420 DATA 800,193,360,197,356,800,443,360,447,356
1430 DATA 805,42,803,195,358,224,52,803,445,358,240,52
1440 DATA 801,497,358,802,520,358,802,520,194,802,375,194
1450 DATA 802,282,210,802,270,261,802,270,289,802,164,289
1460 DATA 803,164,309,48,20,801,144,309,802,144,321,802,158,321
1470 DATA 800,247,358,393,358,805,0,800,246,343,394,343
1480 DATA 800,496,343,519,343,808,320,344,1,808,515,344,1
1490 DATA 805,42,801,270,276,802,164,279,802,164,289
```

Continues on next page….

```
1500 REM -BODY DETAILS
1510 DATA 806,276,289,374,350,800,270,289,270,358
1520 DATA 805,4,808,277,290,8,808,269,288,4
1530 DATA 808,269,290,1,808,145,320,2,805,42,801,520,289
1540 DATA 802,374,289,802,374,210,802,520,210,802,282,210
1550 DATA 805,63,808,400,209,2,805,4,808,380,290,1
1560 DATA 808,510,357,2,808,373,211,1,808,271,289,1
1570 DATA 805,42,801,284,261,802,366,261,802,366,218
1580 DATA 802,294,218,802,284,261,801,276,289,802,276,261
1590 DATA 802,288,213,802,374,213,806,282,264,368,287
1600 DATA 805,36,808,320,219,1,805,42,800,305,218,305,261
1610 DATA 805,26,808,445,288,2,805,21
1620 DATA 803,420,250,12,10,803,480,250,3,10
1630 DATA 803,420,230,48,10,803,480,230,192,10
1640 DATA 800,410,250,410,230,800,490,250,490,230
1650 DATA 800,420,220,480,220,800,420,260,480,260
1660 DATA 803,420,250,12,8,803,480,250,3,8
1670 DATA 803,420,230,48,8,803,480,230,192,8
1680 DATA 800,412,250,412,230,800,488,250,488,230
1690 DATA 800,420,222,480,222,800,420,258,480,258
1700 DATA 805,36,808,450,223,1,805,21,800,450,222,450,258
1710 DATA 805,11,804,143,315,6,5,805,42,803,143,315,60,5
1720 DATA 801,520,299,802,515,299,802,507,292,802,507,289
1730 DATA 801,374,299,802,378,299,802,395,292,802,395,289
1740 DATA 806,356,301,366,314,805,21,808,375,290,1,808,519,290
1750 DATA 1,805,42,807,356,308,363,310,807,272,291,274,301
1760 DATA 807,272,338,274,348,807,274,292,279,298
1770 DATA 807,274,339,279,345,801,279,292,802,295,292
1780 DATA 802,295,294,802,279,298,801,279,339,802,295,339
1790 DATA 802,295,341,802,279,345,808,280,293,8,808,280,340,8
1800 DATA 805,0,812,276,293,812,285,293,812,294,293,812,276,297
1810 DATA 812,276,340,812,285,340,812,294,340,812,276,344
1820 DATA 805,42,806,520,293,525,301,805,21,808,521,294,1
1830 DATA 804,133,333,4,3,804,133,345,2,3,807,134,330,137,348
1840 DATA 805,42,803,133,333,48,3,803,133,345,12,3
1850 DATA 803,176,295,255,2,801,134,330,802,137,330,802,137,348
1860 DATA 802,134,348,805,15,804,639,1,2,50
1870 DATA 813
READY
>
```


A TRS-80 Model 1 with Level II BASIC draws a
Series 1 Land Rover.
A demonstration of my VGA video adaptor and machine code graphics routines.
www.glenzstuff.com