NEW

Advanced
Information
Display Systems

SCION

MicroAngelo
MA 512
User's Manual

SCION Corporation ©1980

SCION Corporation **April 1981**

Warranty

SCION Corporation certifies that each computer system will be free from defective materials and workmanship for ninety (90) days from date of shipment to the original customer.

SCION Corporation agrees to correct any of the above defects when the system is returned to the factory prepaid. Written authorization must be obtained and confirmed in writing by the Customer Service Department before returning the equipment to the factory.

Under this warranty, SCION Corporation will, at the request of the customer, provide the necessary components required by the customer to correct the equipment in the field. The components will be shipped, prepaid, on a billing memo which will be cancelled upon receipt of the defective components at the factory. When ordering components for repair or replacement, the model number and serial number must be included on the customer request.

This warranty is invalid if the system is subject to mis-use, neglect, accident, improper installation or application, alteration or negligence in use, storage, transportation or handling and where the serial number has been removed, defaced or changed.

FOREWORD

April 1981

This Spring marks the beginning of MicroAngelo's second year of wide acceptance across a broad range of applications. In this system we have attempted to bring a carefully designed and integrated hardware/firmware/software package to the marketplace in an affordable and powerful single board graphics computer. We are pleased to share our excitement in the design of MicroAngelo with you.

Charles J. Rieger, III
Vice President
Research & Design

Contents

1. General Information 7
   1.1 Brief System Overview 7
   1.2 Quick Integration Steps 8
   1.3 Driving MicroAngelo™ from High Level Software 8
2. System Integration 11
   2.1 Changing the Port Addresses 11
   2.2 Connecting a TV Monitor 12
   2.3 The Software Interface 12
      2.3.1 Sending a Byte to MicroAngelo™ 12
      2.3.2 Reading a Response From MicroAngelo™ 13
      2.3.3 Restarting MicroAngelo™ 14
      2.3.4 Summary of the Control Port 14
3. Screenware Pak I and Screenware Pak II - the Onboard Software 17
   3.1 ALPHA - the Dumb Terminal Emulator 18
      3.1.1 Dumb Terminal Screen Conventions 18
      3.1.2 Dumb Terminal ASCII Control Codes 19
      3.1.3 Dumb Terminal Printing Options 19
      3.1.4 The Dumb Terminal Interface Code 20
   3.2 GRAPHICS: The MicroAngelo™ Graphics System 20
      3.2.1 GRAPHICS Screen Conventions 21
      3.2.2 GRAPHICS Cursors and Coordinates 21
4. MicroAngelo™ Commands 25
   4.1 ALPHAMODE 26
   4.2 GCURSOR 28
   4.3 SCREEN 29
   4.4 POINT 30
   4.5 VECTOR 31
   4.6 REGION 32
   4.7 CHARACTER 33
   4.8 LIGHTPEN 35
   4.9 CROSSHAIRS 37
   4.10 MEMORY 38
   4.11 UTILITY 39
   4.12 USER 41
   4.13 TEST 42
   4.14 RGRAPHC 43
   4.15 SPLITSCR 44
   4.16 RPOINT 45
   4.17 RVECTOR 46
   4.18 RREGION 47
   4.19 CIRCLE 48
   4.20 FLOOD 49
   4.21 MACRO 50

5. System Details 55
5.1 MicroAngelo™ Memory Map 55
5.2 Defining the Alternate Character Set 56
5.3 Interfacing Onboard User Code to the Screenware 56
5.4 The MicroAngelo™ Physical I/O Ports 57
5.5 Interrupts 57
5.5.1 Enabling/Disabling the Maskable Interrupts 58
5.5.2 Enabling the Real Time Interrupt 58
5.5.3 Connecting Host-Side Interrupts 58
5.6 Connecting a Light Pen 59
5.7 Summary of Hardware Jumper Options and Connectors 59
5.7.1 Hardware Jumpers 60
5.7.2 Hardware Connectors 61
5.8 Adapting MicroAngelo™ to Non-S100 Bus Systems 63
5.9 Bit Mapping of Display RAM to Video Screen 64
6. Software Interface Examples 67
6.1 Graphics: Clear Screen, Draw Triangle, Embed in Region 67
6.2 Turn on and Read the Tracking Cross 67
6.3 Write a Message Around the Border of a square 68
6.4 Underlining in Dumb Terminal Mode 69
6.5 Sample BASIC Interface 70
6.6 Interfacing to FORTRAN 71

Appendix 1 - Summary of Screenware Commands 75
Appendix 2 - The Standard Character Font 79
Appendix 3 - Internal Entry Points [Screenware™ Pak I] 83
Appendix 4 - Internal Entry Points [Screenware™ Pak II] 87
Appendix 5 - MicroAngelo™ Schematics 93

General Information

The MicroAngelo™ Graphics System User's Manual

1. General Information

MicroAngelo™ is an intelligent high resolution refreshed raster scan graphics display system capable of drawing character and graphics images at high speed on a standard television monitor. Completely contained on a single IEEE-696 (S100) bus card, MicroAngelo is an independent Z80A-based computer with its own 32K byte display memory and 4K resident operating system, Screenware Pak I™, or optionally, Screenware Pak II™ (6K). By talking in concise high level commands over a simple inferface, your host computer directs MicroAngelo in generating graphics and text displays and in controlling the light pen interface. Because of its self-reliant architecture, MicroAngelo places no computing load or memory space demand on your computer. This means that, after giving directions to MicroAngelo, your CPU can continue with its own computing as MicroAngelo concurrently carries out those directions using its own separate memory and CPU. The results are a more responsive and convenient graphics/text display system than ever before possible with traditional graphics board designs.

1.1 Brief System Overview

The MicroAngelo hardware resides on a single S100 bus board. This board contains all the electronics and software for generating a 512 dot wide, 480 dot high, black and white display for a high-resolution TV monitor (10 mhz bandwidth or better). Since the board includes a Z80A microprocessor, complete with its own RAM (32K bytes), EPROM (up to 8K bytes), and TV display circuitry, MicroAngelo is actually an independent, single card computer which when inserted into your computer, appears to your system as two parallel ports. This architecture makes it possible for your computer to direct MicroAngelo via simple, powerful high-level graphics commands sent over the two parallel ports, then proceed with its own computations while MicroAngelo carries out the display generation in parallel. Because of this simple and fast two-port interface, MicroAngelo is easy to integrate and does not require **any** of your system's valuable address space.

The MicroAngelo software, Screenware Pak I or Screenware Pak II, has been designed so that the system can be used either as your main console output display, or as a separate graphics display processor, or both. Logically, the Screenware consists of two largely independent software subsystems called ALPHA and GRAPHICS. ALPHA emulates a "dumb terminal" interface, while GRAPHICS supports all the graphics primitives. To get on the air with MicroAngelo as your main output device, you need only implement the simple interface to ALPHA shown below.

1.2 Quick Integration Steps

[Unless otherwise indicated, all memory addresses and operation codes throughout the manual are in hexidecimal notation.]

To interface MicroAngelo to your computer as the main output device, do the following three things:

1. Decide whether or not the MicroAngel parallel ports, mapped from F0-FF, are compatible with your system. If your system currently uses any port in this range, you may have to alter the Port Address Jumpers to some other 16-port boundary. This procedure is described in the section entitled "Changing the Port Addresses".

2. Install the following interface code as your system's main ["console"] output routine. This code will send the byte in the A register to MicroAngelo's ALPHA component, and appear to your operating system to be a "dumb terminal" interface:

```
ttyout  push  psw   save the output byte
tyo1    in    0F1H  read the Control Port
        ani   1     test buffer-full bit
        jnz   ty01  wait until not full
        pop   psw   restore the output byte
        out   0F0H  send it to the Data Port
        ret         return
```

   If you have changed the port addressing as the result of Step 1 above, replace the references to output ports F0 and F1 in this code to the appropriate new values. The software interface to MicroAngelo is described in more detail in the section entitled "Screenware Pak I and Screenware Pak II - The Onboard Software"

3. Connect MicroAngelo to a TV monitor, as described in the section entitled "System Integration".

At power-up time, MicroAngelo will clear the screen and display the winking text cursor in the upper left corner of the screen.

After getting on the air, you will then be able to take full advantage of the MicroAngelo graphics facilities, described in detail in later sections.

1.3. Driving MicroAngelo from High Level Software

If you will be driving MicroAngelo primarily from software written in a higher level language [e.g., BASIC, FORTRAN], you will find the interface very straightforward. Read the section entitled "The Software Interface", then refer to the sections 6.5 and 6.6 for examples.

System Integration

2. System Integration

The system is supplied fully assembled and tested, and is ready to insert into virtually any S100 bus computer after the port addresses have been set to be compatible with the host. [MicroAngelo can be easily adapted to non-S100 bus structures. See the section entitled "Adapting MicroAngelo to Non-S100 Systems".] As shipped, the two MicroAngelo ports are mapped as F0 and F1 in your system's port address space. Because of the way MicroAngelo interprets port addresses, however, the hardware will actually respond to 8 different ports within the group F0-FF, with port addresses F0, F4, F8, FC responding as one port, and F1, F5, F9, FD responding as the second port. Before inserting MicroAngelo into your system, therefore, verify that your system does not already currently use one of these 8 port addresses.

2.1 Changing the Port Addresses

If the MicroAngelo default port addressing is not appropriate for your system, you may move to any other 16 port boundary by altering the Port Address Jumpers J11-J14, which are located near the bottom right corner of the board. As shipped, all four jumpers are set to logic "1" by default printed circuit traces between the center and right hole. To switch a jumper to "0", scratch through the default trace and connect the center and left hole with a short length of wire. Set J11-J14 according to the following table to obtain the desired port mapping:

Desired Ports	J14	J13	J12	J11
00-0F	0	0	0	0
10-1F	0	0	0	1
20-2F	0	0	1	0
30-3F	0	0	1	1
40-4F	0	1	0	0
50-5F	0	1	0	1
60-6F	0	1	1	0
70-7F	0	1	1	1
80-8F	1	0	0	0
90-9F	1	0	0	1
A0-AF	1	0	1	0
B0-BF	1	0	1	1
C0-CF	1	1	0	0
D0-DF	1	1	0	1
E0-EF	1	1	1	0
F0-FF	1	1	1	1

For example, to map the ports in the C0-CF group, cut through the default traces on J11 and J12, and solder in a short wire between the left and center holes on each of these two jumpers.

2.2 Connecting a TV Monitor

The final video signals are available at connector JB at the extreme top left of the board. These pins are numbered 1-6 from left to right, and deliver the following signals:

JB-1	RS-170 composite video
JB-2	ground
JB-3	direct-drive TTL video
JB-4	ground
JB-5	direct-drive, horizontal sync
JB-6	direct-drive, vertical sync

The system can drive either a composite video monitor or a direct-drive monitor, or both simultaneously. Connect a composite video monitor to JB-1, JB-2. Connect a direct-drive monitor to JB-3, JB-4, JB-5, JB-6.

After setting the port addresses and connecting the TV monitor, the MicroAngelo hardware will be fully operational in your host system, and you will then be able to install the simple software interface described in the next sections. The section entitled "System Details" describes other hardware options you may eventually wish to use.

2.3 The Software Interface

All communications between your host computer and MicroAngelo occur over the two ports which have been situated at some 16 port boundary in your system. The lower-addressed port of this pair (e.g., F0) is the **Data Port**, the higher-addressed port (e.g., F1) is the **Control Port**. The Data Port is used for communicating 8-bit data and command bytes to and from MicroAngelo, the Control Port for handshaking and for restarting MicroAngelo. The Screenware constantly monitors these two ports in anticipation of the next graphics command or data byte.

When power is first applied to MicroAngelo, automatic restart circuitry initializes the system hardware and software. The screen is cleared, all cursors and software options described in sections below are set to their default values, and the Screenware begins listening over the Data Port for a command or data.

2.3.1 Sending a Byte to MicroAngelo

The Data Port is a latched, bi-directional pathway with handshaking. "Handshaking" means that before sending a byte, the sender must first verify that the previous byte has been processed by the receiver. Without handshaking the preceding data or command byte, which may not yet have been acted upon by the receiver, might inadvertently be overwritten by the sender's next byte. A latched, handshaking port is essential when each side of the interface is an intelligent system running asynchronously with respect to the other. Handshaking applies symmetrically to both sides of the interface.

Handshaking is accomplished with MicroAngelo as follows. The rightmost bit of the Control Port byte will be "1" when there is a host command or data byte in the outbound Data Port which MicroAngelo has not yet acted upon. Thus, before sending any command or data byte over the Data Port, your system should always read the Control Port, test this "outbound buffer full" bit, and wait for it to become "0", if it is not already.

The following 8080 assembly language subroutine is the standard method of sending a data or command byte from the host's A register to MicroAngelo [without destroying any other registers]:

```
dport       equ     0F0H    declare the Data Port address
cport       equ     0F1H    declare the Control Port address

sendbyte    push    psw     save the byte a moment
sdb1        in      cport   read the Control Port
            ani     1       examine the status bit
            jnz     sbd1    loop if buffer is full
            pop     psw     restore the byte to send
            out     dport   send to the Data Port
            ret             return
```

[Note that this code is exactly what would be used if you were driving a dumb terminal.] See the section entitled "Software Interface Examples" for an equivalent interface written in BASIC.

In the opposite direction, when the Screenware sends the host system a response, an identical mechanism will cause the Screenware to wait for the host [i.e., your software] to read the response from the Data Port before sending the next response byte.

2.3.2. Reading a Response from MicroAngelo

The second from the right bit of the Control Port indicates to the host computer whether or not there is a response byte back from MicroAngelo waiting to be read from the inbound port. When "1", this bit indicates that a response byte is ready to be read over the Data Port; "0" means there is no byte to be read. When the host reads the byte from the Data Port, this bit is automatically reset to "0" to inform the Screenware that it is free to send the next response byte, if any.

The following code is the standard method of reading a response from the Screenware. It waits for a response byte to enter the interface from the MicroAngelo side, then reads it and returns it in the host's A register [without altering any other registers].

```
readbyte    in      cport   read the Control Port
            ani     2       isolate the "data available" bit
            jz      readbyte wait if no byte ready yet
            in      dport   read the byte from the Data Port
            ret             return
```

The SENDBYTE and READBYTE routines implement a complete MicroAngelo interface. In a typical CP/M-based system, these two subroutines should be coded and placed in the USER I/O area, where they can be called by high-level system and user software to control MicroAngelo.

2.3.3. Restarting MicroAngelo

Your system can restart MicroAngelo at any time via the Control Port. By outputting a 01 byte (actually, any byte with the rightmost bit "1") to the Control Port, the host causes the hardware reset condition to begin on the MicroAngelo board. This reset will persist until a 00 byte is sent to the Control Port, and is functionally identical to the power-on reset generated by the MicroAngelo hardware at the time the system was first turned on. Immediately after the host releases the MicroAngelo from the reset, Screenware Pak I will clear the screen and reinitialize all modes and parameters to their default values. All current context will be lost. Screenware Pak II reacts somewhat differently, see Section 4.15 for details.

Example code for restarting MicroAngelo is:

```
graphrst    mvi     a,1         send a "1" to the Control Port
            out     cport
            mvi     a,0
            out     cport
            ret                 return
```

You may wish to include this code in your operating system's warm- and/or cold-start initialization code so that the MicroAngelo display will be restarted each time the host goes through its own initialization sequence. On the other hand, the only condition under which you actually **have** to use the reset is when user software has sent an erroneous or incomplete command sequence to MicroAngelo, or when user-loaded code has lost control onboard MicroAngelo (see the UTILITY and USER commands).

2.3.4. Summary of the Control Port

To summarize, the Control Port plays two roles. Reading this port delivers the interface handshaking bits:

7	6	5	4	3	2	1	0
XX	XX	XX	XX	XX	XX	IF	OF

IF: Inbound buffer (from MicroAngelo to host) is full

OF: Outbound buffer(from host to MicroAngelo) is full

XX: Unused

Writing to this port controls the MicroAngelo hardware reset:

7	6	5	4	3	2	1	0
XX	XX	XX	XX	XX	XX	XX	HR

HR: "1" causes the hardware reset to begin
"0" releases the reset condition, allowing MicroAngelo to restart

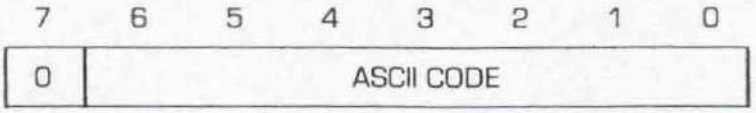
XX: Unused

Screenware Pak I and Screenware Pak II The Onboard Software

3. Screenware Pak I and Screenware Pak II - the Onboard Software

Screenware responds to commands and data sent over the Data Port under the conventions described in the previous section. Screenware can be thought of as two largely independent components: ALPHA and GRAPHICS. The ALPHA (standing for "alpha-numeric") component manages the graphics display as though it were a text-only "dumb terminal". This allows you to get on the air quickly, using MicroAngelo as your system's primary output device. The GRAPHICS component recognizes a variety of graphics commands for operations such as point, vector, region and special character generation, and light pen control. Because of the way the Screenware interprets commands and data, ALPHA and GRAPHICS are both always active, so that you are not forced to be in one mode or the other at each moment, as with some other types of graphics systems.

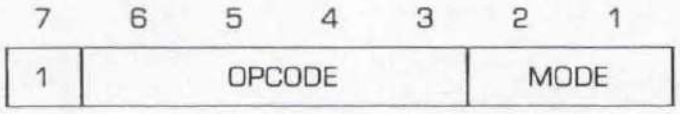
Upon receiving a byte from the host over the Data Port, the Screenware first inspects the high-order bit of the byte. If this bit is "0":



the byte is sent to the ALPHA processor. Since the ALPHA processor is emulating a dumb terminal, the byte will be interpreted as an ASCII character, and acted upon appropriately. If the code is a printing character, it is printed on the screen at the current ALPHA cursor, and the cursor is advanced, possibly invoking the ALPHA scrolling mechanism. Alternatively, if it is an ASCII control character (e.g., carriage-return, backspace), then the ALPHA processor takes the appropriate control action. (The specific ASCII control codes to which ALPHA responds are described below.) Thus, the ALPHA component provides a complete dumb terminal emulation.

If the high bit of a received byte is "1", the byte is interpreted as a command, with the next five high-order bits specifying the opcode. Except for opcode 0 (which relates to the dumb terminal emulator), all commands are handled by the GRAPHICS component.

The Screenware Pak I and Screenware Pak II commands are:



Opcode	Command Name	Function
Screenware Pak I and Screenware Pak II:		
0	ALPHAMODE	select various ALPHA mode options
1	GCURSOR	set or read the graphics cursor
2	SCREEN	clear the screen, set figure/ground
3	POINT	turn on or read a point
4	VECTOR	draw a vector (line)
5	REGION	draw a rectangular region
6	CHARACTER	plot or define a graphics character
7	LIGHTPEN	turn on or off, or read the light pen
8	CROSSHAIRS	control the graphics crosshairs
9	MEMORY	dump, load screen or memory
10	UTILITY	arm USER, call user code, arm RTI
11	USER	call user-defined function

12	TEST	perform diagnostic EPROM, RAM, ALPHA, or Munching Squares test
13	RGRAPHC	move the graphics cursor by a relative amount
14	SPLITSCR	split the screen, or load the default character generator or ASCII control code group
15	RPOINT	plot a point at relative displacement from current cursor
16	RVECTOR	draw a vector to endpoint specified by relative coordinates
17	RREGION	paint a region of extent specified by relative coordinates
18	CIRCLE	draw a circle of specified radius at the current graphics cursor
19	FLOOD	flood a bordered region with all 1's or 0's
20	MACRO	define, invoke, or delete a named graphics object
21-31	RESERVED	reserved for future use

The two rightmost bits of a GRAPHICS command byte are used in specifying a mode or subfunction within these 20 categories. The ALPHAMODE command is described below.

3.1 ALPHA - The Dumb Terminal Emulator

At startup time, the Screenware clears the display, displays a winking text cursor in the upper left corner of the screen, and begins emulating a "dumb terminal" capable of at least a 300 character per second data rate (3000 baud equivalent) under most conditions. Screenware Pak II enhances this rate to more than 6000 baud. (The limiting factor for the data rate is the scrolling software. For applications requiring higher data rates, "rolling" instead of scrolling may work to your advantage. See the ALPHAMODE command.)

Each ASCII code your system sends over the Data Port is treated by the dumb terminal emulator as either a printing ASCII character or an ASCII control code, and will cause the appropriate screen activity to occur automatically.

3.1.1 Dumb Terminal Screen Conventions

The ALPHA processor treats the screen as a text grid of 40 lines of 85 characters per line. Row 0 is at the top, row 39 is at the bottom, column 0 is at the left, column 84 is at the right. The ALPHA CURSOR, (AR, AC), always identifies the screen position to which the next ALPHA character will be written, and is initialized at restart time to (0, 0).

Characters on the screen are 12 pixels high, 6 pixels wide, and are generated by the Screenware from its internal character generator table. (Appendix 2 shows this character set in detail.) However, using the CHARACTER and/or MEMORY commands, you can define a second, alternate set of 128 characters. (See the section "Defining the Alternate Character Set" for a description of this procedure.)

3.1.2. Dumb Terminal ASCII Control Codes

The ALPHA dumb terminal emulator recognizes and processes the following standard ASCII control codes:

BS	[08]	-	Backspace [back up to and erase previous character]
HT	[09]	-	Horizontal Tab [moves to next 8 column boundary]
LF	[0A]	-	Line Feed [ignored]
FF	[0C]	-	Form Feed [clears the screen]
CR	[0D]	-	Carriage Return [also does a line feed]
ESC	[1B]	-	Escape [causes the next ALPHA byte to be printed literally]
DEL	[7F]	-	Delete [treated as BS]

Screenware Pak II conditionally recognizes

HOME	[01]	-	Home alpha cursor
DELEOL	[0E]	-	Delete text to end-of-line
DELEOP	[0F]	-	Delete text to end-of-page
CURUP	[11]	-	Cursor up
CURDN	[12]	-	Cursor down
CURLF	[13]	-	Cursor left
CURRT	[14]	-	Cursor right

3.1.3. Dumb Terminal Printing Options

The dumb terminal emulator can be conditioned to print text in a number of special modes. If you do not need any of these modes, no action is required. However, the following modes are available and can be selected by calls to the ALPHAMODE command described in the section entitled "MicroAngelo Commands":

1. Figure/ground [whether to print white-on-black or black-on-white characters]
2. Underlining [whether or not to underline characters as they are printed]
3. Overstrike [whether to overstrike or print as usual]
4. Font [whether to use the standard or user-defined font]
5. Cursor [whether or not the winking cursor should be displayed]
6. Scroll [how much to pop up when text would fall off the bottom of the screen]
7. Coordinates [where to print the next text character]

The defaults for these are:

1. Light Characters on dark background
2. Underlining off
3. Overstrike off
4. Standard font
5. Visible cursor
6. 10-line pop-up
7. Starting cursor coordinates at row 0, column 0

See the ALPHAMODE command if you wish to change any of these defaults.

3.1.4. The Dumb Terminal Interface Code

Because of the ALPHA component's ability to emulate a standard terminal, MicroAngelo will become your system's main output device after a simple integration step. To make MicroAngelo your main output device, install the following code in your system's User area as the subroutine to be called to output the A register to the screen. In this code (which is repeated from the section entitled "Quick Integration Steps"), DPORT and CPORT refer to the two communications ports described earlier. Unless you have changed the port mapping, these are F0 and F1, respectively.

```
dport     equ     0F0H    declare the Data Port
cport     equ     0F1H    declare the Control Port

ttyout    push    psw     save the output character
tt01      in      cport   read the MicroAngelo Control Port
          ani     1       test the output status bit
          jnz     tt01    loop if interface buffer still full
          pop     psw     send the character
          out     dport   to the MicroAngelo Data Port
          ret             return
```

If you wish warm- and/or cold-starts of your system to restart MicroAngelo, also insert the following reset code in your host system's initialization sequence[s]:

```
ttyrst    mvi     a,1     send a hardware reset
          out     cport   to the Control Port
          mvi     a,0     release the reset condition
          out     cport
          ret             return
```

3.2. GRAPHICS - The MicroAngelo Graphics System

The GRAPHICS processor is responsible for plotting points, vectors, regions and characters of special size or orientation, and for controlling the light pen interface. GRAPHICS responds to various commands described in the section entitled "MicroAngelo Commands", and is largely independent of the ALPHA processor, which emulates a dumb, text-only terminal. The sections below describe the GRAPHICS conventions and cursors.

3.2.1. GRAPHICS Screen Conventions

The Screen is a 512 wide by 480 high grid of on/off pixels ["picture elements"]. X coordinates range from 0-511 left to right, Y coordinates from 0-479 bottom to top. In the descriptions below, the term "graphics coordinates" refers to this coordinate system. Since a graphics coordinate requires 9 bits, two bytes are used when specifying a graphics coordinate to MicroAngelo. By convention, the high byte is always sent first, the low byte second. For example, to send the coordinate 293 decimal [125 hex], send a first byte of 01 hex, a second byte of 25 hex. Any graphics X coordinate larger than 511 or Y coordinate larger than 479 sent to Screenware will be clipped to its maximum value.

A pixel is "on" when a "1" bit is stored in its corresponding location in the MicroAngelo display memory. However, whether an "on" condition is seen as a light dot on a dark background or a dark dot on a light background is determined by the setting of the screen's figure/ground hardware, described in the SCREEN primitive below.

3.2.2. GRAPHICS Cursors and Coordinates

The Screenware continously maintains six cursor and coordinate pairs:

[AR,AC] - the current row and column of the ALPHA CURSOR; AR ranges from 0-39 top to bottom, AC from 0 to 84 left to right

[AX,AY] - the graphics coordinates of the lower left pixel of the character at [AR,AC]

[CX,CY] - the main GRAPHICS CURSOR'S coordinates

[LX,LY] - the coordinates of the most recent light pen firing

[TX,TY] - the graphics coordinates of the tracking cross

[HX,HY] - the graphics coordinates of the crosshairs

[AR,AC] and [AX,AY] are maintained by the ALPHA component. The others are described in the following sections, and are all initialized to [0,0] at restart time.

MicroAngelo Commands

4. MicroAngelo Commands

This section describes the 12 Screenware Pak I and Pak II commands, and 9 Screenware Pak II commands. In these descriptions the calling sequence is indicated by

CALL: ⟨ hex opcode ⟩ ⟨ byte ⟩ . . . ⟨ byte ⟩

i.e., to use the command, send the hex opcode followed by the specified byte-sized parameters, all over the Data Port. MicroAngelo responses, if any, are indicated by

RESPONSE: ⟨ byte ⟩ . . . ⟨ byte ⟩

If a command generates responses, your software must always read those responses. Otherwise, the Screenware will become backlogged and will eventually stop responding until any outstanding responses are read.

The first command, ALPHAMODE, is used to set the various dumb terminal printing options, and relates more to the ALPHA component than to the GRAPHICS component. The remaining commands relate to MicroAngelo graphics. Appendix 1 summarizes all commands and gives decimal and octal equivalents for the opcodes.

4.1. ALPHAMODE

	7	6	5	4	3	2	1	0
OPCODE 0 - ALPHAMODE	1	0	0	0	0	0	M	M

MODE 0: SET ALPHA MODE BITS
CALL: 80 ‹ mode ›
RESPONSE: none

The ALPHA MODE word is set to the ‹ mode › byte. The format of the ALPHA MODE word is:

SC	EC	HS	CU	FO	OS	UL	FG

SC "0" [PAK I] means do not clear screen or home [AR,AC]
"1" [PAK I] means clear screen and home [AR,AC]
"0" [PAK II] means do not clear alpha area or home [AR,AC]
"1" [PAK II] means clear alpha area and home [AR,AC]

[SC is not actually stored as part of the ALPHA MODE word, but has only a one-time effect at command time.]

EC "0" [Pak II only] disables special ASCII code interpretation
"1" [Pak II only] enables special ASCII code interpretation

HS "0" [Pak II only] selects normal mode
"1" [Pak II only] selects high speed mode

CU "0" enables display of the winking cursor
"1" inhibits display of the cursor

FO "0" selects the standard Screenware Pak character set
"1" selects the user define character set

OS "0" selects normal erase-before-print mode
"1" selects character overstrike mode

UL "0" inhibits underlining
"1" turns on underlining

FG "0" selects light characters on dark background
"1" selects dark characters on light background

Bits 20H and 40H of the ALPHA mode word have meaning in Pak II. Bit 20H of the ALPHA MODE word is now defined as the "high speed select" bit. When set to 1, the new high speed ALPHA mode is selected, when set to 0 the normal [although also somewhat improved] mode is selected. the poweron default is normal mode. In high speed mode, only the innermost 8 scan lines of the character are generated, leaving the top and bottom 2 of all characters' 12 scan lines ungenerated. While this is adequate for all characters in the default character set, user-defined characters that make use of the top or bottom 2 lines will not be fully generated in high speed mode. Additionally, the high speed mode ignores the figure/ground, underline, and overstrike option bits.

Bits 40H of the ALPHA mode word governs whether or not the special ASCII control codes for cursor and screen control will be enabled [see the SPLITSCR command]. When this bit is 1, special codes will be processed, and will take precedence over any other interpretation of those 8 ASCII characters. When this bit is 0 [the power on default], codes will not be recognized.

MODE 1: POSITION ALPHA CURSOR
CALL: 81 ⟨ row ⟩ ⟨ col ⟩
RESPONSE: none

> The ALPHA CURSOR is set to [⟨ row ⟩ , ⟨ col ⟩]. This "escape sequence" allows for quick repositioning of the cursor. Subsequent text will be printed starting at the new location.

MODE 2: READ ALPHA CURSOR
CALL: 82
RESPONSE: ⟨ row ⟩ ⟨ col ⟩

> The Current ALPHA CURSOR location is returned, row first then column.

MODE 3: SET ALPHA SCROLL NOT SUPPORTED
CALL: 83 ⟨ n ⟩
RESPONSE: none

> The ALPHA scroll parameter is set to ⟨ n ⟩ . If ⟨ n ⟩ = 0, "roll mode" is selected. In this mode, rather than popping up, the cursor wraps around to the top line and clears one line at a time in advance as it reuses the screen. This mode is fastest, since it requires no pop-up time, but can be somewhat visually confusing. If ⟨ n ⟩ is greater than 0 and less than 40, the screen will be popped up ⟨ n ⟩ lines each time text is about to fall off the bottom . If ⟨ n ⟩ is greater than 39, the entire screen will be cleared at pop-up time, and new text begun at the top.

Notes

The SPLITSCR command augments the ALPHAMODE command and provides two other services relating to the ALPHA facility. In particular, the ALPHA screen can now be restricted to a user defined number of bottom screen lines. When the screen has been split by this command, issuing the ALPHA screen clear command clears only this bottom region. Also, the scroll parameter applies to this bottom region, and is set by SPLITSCR. Refer to the SPLITSCR sections for details.

4.2. GCURSOR

OPCODE 1 - GCURSOR

7	6	5	4	3	2	1	0
1	0	0	0	0	1	M	M

MODE 0: SET GRAPHICS CURSOR
CALL: 84 ⟨ xh ⟩ ⟨ xl ⟩ ⟨ yh ⟩ ⟨ yl ⟩
RESPONSE: none

The Graphics cursor [CX,CY] is set to the values specified. [⟨ xh ⟩ is the high byte of the CX coordinate, ⟨ xl ⟩ is the low byte, ⟨ yh ⟩ is the high byte of the CY coordinate, ⟨ yl ⟩ the low byte.] The main graphics cursor is never actually visible, but serves as the relative origin of several graphics operations. [CX,CY] is automatically moved by several graphics operations.

MODE 1: READ GRAPHICS CURSOR
CALL: 85
RESPONSE: ⟨ xh ⟩ ⟨ xl ⟩ ⟨ yh ⟩ ⟨ yl ⟩

The current [CX,CY] coordinates are reported.

MODE 2: SET [CX,CY] TO [AX,AY]
CALL: 86
RESPONSE: none

CX is set to AX, CY is set to AY. This is useful for coordinating text and graphics.

MODE 3: SET [CX,CY] TO [TX,TY]
CALL: 87
RESPONSE: none

[CX,CY] are set to [TX,TY].

4.3. SCREEN

OPCODE 2 - SCREEN

7	6	5	4	3	2	1	0
1	0	0	0	1	0	M	M

MODE 0: CLEAR SCREEN
CALL: 88
RESPONSE: none

The display screen is cleared by turning all pixels "off". If the figure/ground has been set to light-on-dark, the screen goes completely dark. If the figure/ground has been set to dark-on-light, the screen goes completely light.

NOTES

In Screenware Pak II the CLEAR SCREEN command applies only to the top region of the screen, in case the SPLITSCR command has been issued to divide the screen between top (graphics/text) and bottom (dumb terminal text only). If the screen is not divided (i.e., all 40 lines are allocated to the ALPHA screen), CLEAR SCREEN will clear the entire screen. Refer to the SPLITSCR command for details. Also, the tracking cross and crosshairs are momentarily removed (if on) during a clear so that they are not erroneously erased.

MODE 1: SET SCREEN FIGURE/GROUND
CALL: 89 ⟨ fg ⟩
RESPONSE: none

The figure ground is set according to the rightmost bit of the following byte, ⟨ fg ⟩. A "0" bit selects light-on-dark, a "1" bit selects dark-on-light.

MODE 2: TOGGLE SCREEN FIGURE/GROUND
CALL: 8A
RESPONSE: none

The current figure/ground is toggled. This is useful, for example, in rapid screen flashes to attract the user's attention.

MODE 3: READ SCREEN FIGURE/GROUND
CALL: 8B
RESPONSE: ⟨ fg ⟩

The current figure/ground status is returned as the rightmost bit of the response byte.

4.4 POINT

OPCODE 3 - POINT

7	6	5	4	3	2	1	0
1	0	0	0	1	1	M	M

MODE 0: TURN POINT OFF
CALL: 8C ⟨ xh ⟩ ⟨ xl ⟩ ⟨ yh ⟩ ⟨ yl ⟩
RESPONSE: none

The point at the specified graphics coordinates is turned off. [CX, CY] are set to this location.

MODE 1: TURN POINT ON
CALL: 8D ⟨ xh ⟩ ⟨ xl ⟩ ⟨ yh ⟩ ⟨ yl ⟩
RESPONSE: none

The point at the specified graphics coordinates is turned on. [CX, CY] are set to this location.

MODE 2: COMPLEMENT POINT
CALL: 8E ⟨ xh ⟩ ⟨ xl ⟩ ⟨ yh ⟩ ⟨ yl ⟩
RESPONSE: none

The point at the specified graphics coordinates is complemented. [CX, CY] are set to this location.

MODE 3: READ POINT
CALL: 8F ⟨ xh ⟩ ⟨ xl ⟩ ⟨ yh ⟩ ⟨ yl ⟩
RESPONSE: ⟨ val ⟩

A byte containing only the requested pixel is returned. If this byte is zero, the point is off; if non-zero, the point is on. [CX, CY] are set to this location.

4.5 VECTOR

OPCODE 4 - VECTOR

7	6	5	4	3	2	1	0
1	0	0	1	0	0	M	M

MODE 0: TURN VECTOR OFF
CALL: 90 ⟨ xh ⟩ ⟨ xl ⟩ ⟨ yh ⟩ ⟨ yl ⟩
RESPONSE: none

All points lying along the vector between and including [CX, CY] and the coordinates specified in the command are turned off. [CX, CY] are set to the new endpoint after the operation.

MODE 1: TURN VECTOR ON
CALL: 91 ⟨ xh ⟩ ⟨ xl ⟩ ⟨ yh ⟩ ⟨ yl ⟩
RESPONSE: none

All points lying along the vector between and including [CX, CY] and the coordinates specified in the command are turned on. [CX, CY] are set to the new endpoint after the operation.

MODE 2: COMPLEMENT VECTOR
CALL: 92 ⟨ xh ⟩ ⟨ xl ⟩ ⟨ yh ⟩ ⟨ yl ⟩
RESPONSE: none

All points lying along the vector between and including [CX, CY] and the coordinates specified in the command are complemented. [CX, CY] are set to the new endpoint after the operation.

MODE 3: NO OPERATION

4.6 REGION

OPCODE 5 - REGION

7	6	5	4	3	2	1	0
1	0	0	1	0	1	M	M

MODE 0: TURN REGION OFF
CALL: 94 ⟨x1h⟩ ⟨x1l⟩ ⟨y1h⟩ ⟨y1l⟩ ⟨x2h⟩ ⟨x2l⟩ ⟨y2h⟩ ⟨y2l⟩
RESPONSE: none

All bits in the rectangular region identified by the diagonally opposing corner points given in the command are turned off. [CX, CY] are unaffected.

MODE 1: TURN REGION ON
CALL: 95 ⟨x1h⟩ ⟨x1l⟩ ⟨y1h⟩ ⟨y1l⟩ ⟨x2h⟩ ⟨x2l⟩ ⟨y2h⟩ ⟨y2l⟩
RESPONSE: none

All bits in the rectangular region identified by the diagonally opposing corner points given in the command are turned on. [CX, CY] are unaffected.

MODE 2: COMPLEMENT REGION
CALL: 96 ⟨x1h⟩ ⟨x1l⟩ ⟨y1h⟩ ⟨y1l⟩ ⟨x2h⟩ ⟨x2l⟩ ⟨y2h⟩ ⟨y2l⟩
RESPONSE: none

All bits in the rectangular region identified by the diagonally opposing corner points given in the command are complemented. [CX, CY] are unaffected.

MODE 3: NO OPERATION

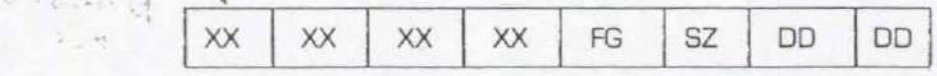
4.7 CHARACTER

OPCODE 6 - CHARACTER

7	6	5	4	3	2	1	0
1	0	0	1	1	0	M	M

MODE 0: PLOT GRAPHICS CHARACTER
CALL: 98 ⟨ c ⟩
RESPONSE: none

The character identified by the following byte, ⟨ c ⟩, is plotted at [CX, CY], and [CX, CY] is advanced to the position at which the next graphics character of similar type would be plotted. [CX, CY] defines where the lower left pixel of the character [with respect to the character's frame of reference] is to be plotted. The low-order 7 bits of ⟨ c ⟩ are the ASCII code of the desired character. The high-order bit identifies the font: "0" for standard, "1" for user-defined. [These are the same fonts as used by ALPHA.] The plotting of the character is carried out according to the four mode bits in the GRAPHICS MODE WORD [see MODE 1 below]:



DD: These two bits determine the character's print direction and orientation, as follows:

0: left to right, character upright
1: right to left, character upside-down
2: bottom to top, character 90 degrees ccw
3: top to bottom, character 90 degrees cw

SZ: "0" selects normal size character [6 by 12]
"1" selects double size character [12 by 24]

FG: "0" selects light on dark figure/ground
"1" selects dark on light figure/ground

For example, to write a double-size, dark on light message up the left edge of the screen [characters 90 degrees CCW], set the mode word to 0E. Note that GRAPHICS characters plotted by this command have no relation to the ALPHA component, except that both rely on the same fonts. Because of the added complexity, the GRAPHICS mode character plotting takes somewhat longer than ALPHA mode.

MODE 1: SET GRAPHICS CHARACTER MODE
CALL: 99 ⟨ mode ⟩
RESPONSE: none

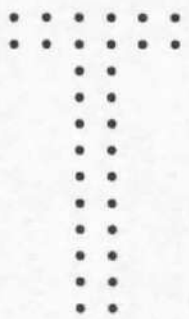
The GRAPHICS MODE word is set to ⟨ mode ⟩. The modes thus defined apply to all subsequent GRAPHICS characters. [See above].

MODE 2: DEFINE ALTERNATE CHARACTER
CALL: 9A ⟨ asc ⟩ ⟨ s11 ⟩ ... ⟨ s0 ⟩
RESPONSE: none

The 6 by 12 bit pattern for ASCII character code ⟨ asc ⟩ is defined and inserted into the user-defined font. The bit pattern is sent as 12 bytes ⟨ s11 ⟩, ..., ⟨ s0 ⟩ which represent 12 scan lines of the character, from top to bottom. Each ⟨ si ⟩ byte's low order 6 bits define the 6 pixels across that scan line of the character. For example, to define ASCII code 13 as a bold, full-height "T", you would call the Screenware as follows:

9A 13 3F 3F 0C 0C 0C 0C 0C 0C 0C 0C 0C 0C

When printed, this character would then appear on the screen as:



To install a complete user font, the UTILITY primitive's block DEPOSIT mode is faster. The user-defined font is stored in MicroAngelo's memory beginning at address 0F940H. By depositing 12*128 = 1536 continuous bytes starting at this address, you will effectively be loading the entire user-defined font in one command.

MODE 3: LOAD DEFAULT CHARACTER SET [Screenware Pak II only]
CALL: 9B
RESPONSE: none

The standard MicroAngelo character set in EPROM is copied to the user-defined font region. Note that this region may also be in use for other purposes [see the USER and MACRO commands], so that care should be taken in managing this storage. This command is useful when the user wishes the alternate character set to be largely similar to the default, except where changed via the DEFINE ALTERNATE CHARACTER command.

4.8 LIGHTPEN

OPCODE 7 - LIGHTPEN

7	6	5	4	3	2	1	0
1	0	0	1	1	1	M	M

The light pen interface [described electrically in the section entitled "Connecting a Lightpen"] provides a method of communicating with host software by pointing rather than typing. When operating, the light pen will generate pulses that are converted to coordinates by the Screenware. In Screenware, the light pen software is always enabled, and is always ready to record the most recent light pen signal coordinates, [LX, LY]. These coordinates are accurate to two pixels vertically and horizontally when a quality light pen is used [see the section entitled "Connecting a Light Pen"].

When the "tracking cross" is turned on [and visible as a small complemented cross on the screen], any light pen activity within the vicinity of the cross is interpreted as a command to adjust the cross so that it is dead-centered under the light pen. With the Screenware continulusly [and at high speed] adjusting its location to remain under the light pen, the cross appears to follow the pen where ever the user moves it. When the tracking cross is enabled, its coordinates are known as [TX, TY].

The following commands deal with the light pen interface.

MODE 0: TURN TRACKING CROSS OFF
CALL: 9C
RESPONSE: none

The light pen tracking cross is removed from the screen, if present. The system powers up with the cross off.

MODE 1: TURN TRACKING CROSS ON
CALL: 9D ⟨ xh ⟩ ⟨ xl ⟩ ⟨ yh ⟩ ⟨ yl ⟩
RESPONSE: none

If the tracking cross is on, it is turned off. The cross is then displayed at the specified coordinates, and [TX, TY] are set to this position.

MODE 2: READ TRACKING CROSS
CALL: 9E
RESPONSE: 00
or
01 ⟨ xh ⟩ ⟨ xl ⟩ ⟨ yh ⟩ ⟨ yl ⟩

The current tracking cross coordinates, [TX, TY], are returned.

MODE 3: READ LIGHT PEN
CALL: 9F
RESPONSE: 00
or
01 〈xh〉 〈xl〉 〈yh〉 〈yl〉

Regardless of whether or not the tracking cross is on, if the light pen has fired since the last reading via this command, a 01 byte, followed by the most recent light pen coordinates, is returned. A 00 response is returned if the light pen has not fired since the last reading. The light pen is logically reset to await another firing. This mode is useful, for example, in detecting when the user is pointing at a menu item on the screen.

Notes

In Screenware Pak II the tracking cross pen-following algorithm has been improved to provide a more stable cross display, and to provide better tracking response. Also, the tracking cross is now momentarily removed [if on] during either an ALPHA or GRAPHICS screen clear or ALPHA scroll to prevent its erroneous erasure or duplication

4.9 CROSSHAIRS

OPCODE 8 - CROSSHAIRS

7	6	5	4	3	2	1	0
1	0	1	0	0	0	M	M

The Screenware "crosshairs" are a full-screen vertical line and horizontal line which, when visible, intersect at the current crosshair coordinates [HX, HY]. Crosshairs are useful for indicating the coordinates of the next graphics operation in an interactive design environment. The crosshairs are independent of the main graphics cursor [CX, CY] and the tracking cross and lightpen coordinates [TX, TY] and [LX, LY]. However, simple user software that constantly monitors these other coordinates can logically couple the crosshairs to any of them.

MODE 0: TURN CROSSHAIRS OFF
CALL: A0
RESPONSE: none

If the crosshairs are on, they are turned off. [HX, HY] remain as they are.

MODE 1: DRAW CROSSHAIRS
CALL: A1 ⟨ xh ⟩ ⟨ xl ⟩ ⟨ yh ⟩ ⟨ yl ⟩
RESPONSE: none

If the crosshairs are on, they are turned off. The crosshairs are then turned on at the specified coordinates, and [HX, HY] are set to these coordinates.

MODE 2: READ CROSSHAIRS
CALL: A2
RESPONSE: ⟨ xh ⟩ ⟨ xl ⟩ ⟨ yh ⟩ ⟨ yl ⟩

The current crosshair coordinates, [HX, HY], are returned.

MODE 3: DRAW CROSSHAIRS AT [CX, CY]
CALL: A3
RESPONSE: none

If the crosshairs are on, they are turned off. [HX, HY] are set to [CX, CY] and the crosshairs are drawn at this new location.

Notes

In Screenware Pak II the crosshairs are now momentarily removed [if on] during ALPHA or GRAPHICS screen clears and for ALPHA scrolling to prevent their erroneous erasure or duplication.

4.10 MEMORY

OPCODE 9 - MEMORY

7	6	5	4	3	2	1	0
1	0	1	0	0	1	M	M

MODE 0: DUMP SCREEN
CALL: A4
RESPONSE: ⟨ b1 ⟩ ... ⟨ b7800 ⟩

The 7800H bytes of the display screen are reported, top screen scan line first, working left to right. This command is useful for storing screen images on disk.

MODE 1: LOAD SCREEN
CALL: A5 ⟨ b1 ⟩ ... ⟨ b7800 ⟩
RESPONSE: none

The 7800H bytes of the display screen are loaded, top screen scan line first, working left to right. This command will load a previously dumped screen image.

MODE 2; EXAMINE MEMORY BLOCK
CALL: A6 ⟨ nh ⟩ ⟨ nl ⟩ ⟨ ah ⟩ ⟨ al ⟩
RESPONSE: ⟨ b1 ⟩ ... ⟨ bn ⟩

The N bytes [specified by ⟨ nh ⟩ ⟨ nl ⟩] of MicroAngelo's memory starting at the address specified by ⟨ ah ⟩ ⟨ al ⟩ are reported. See the section entitled "The MicroAngelo Memory Map" for a description of how the system's memory space is allocated.

MODE 3: DEPOSIT MEMORY BLOCK
CALL: A7 ⟨ nh ⟩ ⟨ nl ⟩ ⟨ ah ⟩ ⟨ al ⟩ ⟨ b1 ⟩ ... ⟨ bn ⟩
RESPONSE: none

The memory block of specified length and starting address is loaded, using the N bytes following the command. This command is useful for loading the alternate font, and for loading user graphics code to augment the Screenware. To load a complete user-defined font of 128 ASCII characters of 12 scan lines (bytes) each, say:

A7 06 00 F9 40

then write the 600H font bytes to the Data Port. (See the section entitled "Defining the Alternate Character Set" for more details.) Before loading user code via this command, see the section entitled "The MicroAngelo Memory Map".

Notes

In Screenware Pak II memory deposits and screen loads run much faster because of a change in protocol. Memory examines and screen dumps run slightly faster.

OPCODE 10 - UTILITY

7	6	5	4	3	2	1	0
1	0	1	0	1	0	M	M

MODE 0: SET USER COMMAND ADDRESS
CALL: A8 ⟨ ah ⟩ ⟨ al ⟩
RESPONSE: none

The address of the code to be called by the USER command [opcode 11] is defined as ⟨ ah ⟩ ⟨ al ⟩. The code should have been deposited into MicroAngelo's RAM via a MEMORY command prior to this command. See the section entitled "The MicroAngelo Memory Map" before installing any user code.

MODE 1: CALL USER CODE
CALL: A9 ⟨ ah ⟩ ⟨ al ⟩ ⟨ imask ⟩ ⟨ iah ⟩ ⟨ ial ⟩
RESPONSE: none

The Screenware calls the user code at the specified address. The user code gains control of the MicroAngelo CPU, may alter all registers except the stack pointer, and can return by executing a RET instruction. If the stack pointer is altered, the Screenware should be reentered at location 0, [Pak I] or location 69H [Pak II], i.e., restarted.

As the user code is called, 3 types of logical interrupts can be enabled: DFHI [Data From Host], DTHI [Data To Host], and LPI [Light Pen]. [See the section entitled "Interrupts" for a description of MicroAngelo interrupts.] ⟨ imask ⟩ identifies which [if any] interrupt sources to enable:

7	6	5	4	3	2	1	0
XX	XX	XX	XX	XX	LP	DT	DF

LP enable Light Pen interrupts
DT enable Data To Host interrupts
DF enable Data From Host interrupts

When an enabled interrupt occurs, the user interrupt handling code at the address specified by ⟨ iah ⟩ ⟨ ial ⟩ will be called under the following context: [1] interrupts will be disabled, [2] an EX AF, AF', EXX sequence will have been done to save all registers, [3] the A register will contain the interrupt mask [in the format shown above] defining the source[s] of the current interrupt. After finishing, the interrupt handling code should return via the sequence EX AF, AF', EXX, EI, RET. This CALL command will permit you to install a completely independent operating system within MicroAngelo, and will give this operating system access to interrupts.

MODE 2: SWITCH REAL-TIME INTERRUPTS
CALL: AA 00
or
AA 01 〈ah〉 〈al〉
RESPONSE: none

If the second byte of the command is 00, the 1/60 second real-time interval interrupts are disabled. If the second byte is 01, real-time interrupts are enabled, and will call the user-defined code at location 〈ah〉 〈al〉. This code should protect all registers **on the stack** [i.e., not via an EX AF, AF', EXX sequence], and should return via a RETI instruction, since the real-time clock interrupt is non-maskable. Before arming or using the real-time clock, read the section entitled "Interrupts".

MODE 3: FORCE COLD START [Screenware Pak II only]
CALL: AB
RESPONSE: none

A cold poweron sequence is forced, causing the MicroAngelo to be completely reset ~~and send a byte 〈AB〉 to the host~~. This command is necessary because Screenware Pak II distinguishes between the first and subsequent hardware resets by storing and reading a flag byte [a byte which would be extremely unlikely to appear in RAM randomly at poweron.].

4.12 USER

OPCODE 11 - USER

7	6	5	4	3	2	1	0
1	0	1	0	1	1	M	M

MODES 0,1,2,3: CALL USER PRIMITIVE
CALLS: AC, AD, AE, AF
RESPONSES: user-defined

This command provides a simple interface wherein user-extensions to Screenware software can be called. Before using this command, first install the user code in MicroAngelo's RAM using the MEMORY command's DEPOSIT mode. Then declare the code's entry address via the UTILITY command's MODE 0. After this setup procedure, the four USER opcodes shown above will all be routed to this user code. At call time, the two mode bits (i.e., the bits that distinguish the four USER command opcodes) are available to the user code as the two rightmost bits of the B register (all other bits zero). The user code is permitted to alter any registers except the stack pointer, and should return to the Screenware via a RET instruction. Before using this feature, read the section entitled "The MicroAngelo Memory Map".

Notes

The USER command will usually consume memory which is also used by the CHARACTER commands (pertaining to the user-defined alternate character set). Since the MACRO facility (Screenware Pak II only) will also require some of this memory, additional care in allocating this space should be exercised. Refer to the MACRO command for details.

4.13 TEST (Screenware Pak II only)

	7	6	5	4	3	2	1	0
OPCODE 12 - TEST	1	0	1	1	0	0	M	M

MODE 0: TEST EPROM
CALL: B0 ⟨ blocks ⟩
RESPONSE: ⟨ cksum ⟩

⟨ n ⟩ 1024 byte blocks, starting at address 0, of the EPROM are checksummed, and the result returned as ⟨cksum ⟩ , computed by summing all bytes in the block, modulo 256. This command provides a means of verifying that the EPROMs are functioning correctly. Specify 6 blocks to test all of Screenware Pak II. The checksum for each EPROM is noted on the EPROM's label. When testing more than one EPROM (i.e., testing 4 or 6 blocks), add the individual EPROMs' checksums (in hexidecimal) to compare with the TEST EPROM's returned ⟨ cksum ⟩

MODE 1: TEST RAM
CALL: B1
RESPONSE: 0 or
1 ⟨ ah ⟩ ⟨ al ⟩ ⟨ eb ⟩ ⟨ fb ⟩

The entire 32K MicroAngelo RAM is tested by writing a cyclic test pattern, which ensures that every possible byte value has been successfully stored and read in every memory location. The test requires several minutes, and is visible as patterns of changing vertical bands on the screen. If the test discovers no faults, a 0 response is returned and a cold poweron sequence executed to reset the system. If a fault is discovered, a 1 followed by the faulty address high and low bytes, expected data byte, and faulty data byte, respectively, are returned. The Screenware then disables interrupts, and enters a halt loop, under the assumption that useful computations are no longer possible.

MODE 2: ALPHA TEST
CALL: B2
RESPONSE: none

The entire default character set is repetitively printed to the ALPHA screen, exercizing the figure/ground and underline options in various combinations. All ALPHA modes are left unaffected by the test.

MODE 3: MUNCHING SQUARES
CALL: B3 ⟨ s ⟩ ⟨ i ⟩ ⟨ n ⟩
RESPONSE: none

Visually interesting, changing geometric patterns are generated by the Munching Squares algorithm. The seed ⟨ s ⟩ and increment ⟨ i ⟩ are any 8 bit values, and determine the pattern that will be repetitively generated. ⟨ n ⟩ , any 6 bit value, determines how many cycles the display will run through before terminating and clearing the screen [⟨ n ⟩ = 0 causes 64 cycles]. Each ⟨ n ⟩ unit corresponds to about 45 seconds of real time. Try some of these values of [⟨ s ⟩ , ⟨ i ⟩] for starters: [1,1], [5,19], [2,2], [7,3].

4.14 RGRAPHC (Screenware Pak II only)

OPCODE 13 - RGRAPHC

7	6	5	4	3	2	1	0
1	0	1	1	0	1	M	M

MODE 0: SET RELATIVE GRAPHICS CURSOR
CALL: B4 〈 dxh 〉 〈 dxl 〉 〈 dyh 〉 〈 dyl 〉
RESPONSE: none

The graphics cursor is moved by an offset specified by the four calling bytes. 2's complement arithmetic is used for negative offsets. As with the GRAPHIC command, RGRAPHC clips if necessary to keep the graphics cursor in bounds.

MODE 1: NO OPERATION

MODE 2: NO OPERATION

MODE 3: NO OPERATION

4.15 SPLITSCR (Screenware Pak II only)

NOT SUPPORTED

	7	6	5	4	3	2	1	0
OPCODE 14 - SPLITSCR	1	0	1	1	1	0	M	M

MODE 0: SET ALPHA SCREEN SIZE
CALL: B8 ⟨ I ⟩
RESPONSE: none

The screen is logically split between a top graphics/text region and bottom text/scrolling region. ⟨ I ⟩ specifies the number of text lines to be allocated as the bottom region, and is clipped to the range 1-40 if not already in that range. Screenware Pak II powers on with an ⟨ I ⟩ value of 40 (i.e., the entire screen is available to the ALPHA processor, as in Screenware Pak I). Note that splitting the screen does not restrict graphics to the top region, but rather only restricts the ALPHA facility to the bottom region. Two side effects of this command are that the ALPHA cursor is homed, and that the ALPHA scroll parameter (the number of lines to pop up when the ALPHA region of the screen is full) is set to one-quarter the new ALPHA region height (or 1 minimum). However, the user is free to redefine the scroll parameter after a SPLITSCR. SPLITSCR may be called at any time to redefine the size of the ALPHA area.

MODE 1: DEFINE ALPHA CONTROL CODES
CALL: B9 ⟨ c1 ⟩ ... ⟨ c8 ⟩
RESPONSE: none

The ALPHA (dumb terminal) processor can now be instructed to recognize eight special ASCII control codes:

01H	HOME	the ALPHA cursor is homed to the top left of the ALPHA region
0EH	DELEOL	text at and beyond the current ALPHA cursor is deleted to the end of the line
0FH	DELEOP	text at and beyond the current ALPHA cursor is deleted to the end of the page (ALPHA region)
11H	CURUP	the ALPHA cursor is moved up one line if possible
12H	CURDN	the ALPHA cursor is moved down one line if possible
13H	CURLF	the ALPHA cursor is moved left one character if possible
14H	CURRT	the ALPHA cursor is moved right one character if possible
0CH	FF	the ALPHA region is cleared (form feed), and the cursor is homed

To maintain Screenware Pak I compatibility, the ALPHA processor will interpret these special codes only when the 40H bit of the ALPHA mode word is set (refer to the ALPHAMODE command). If the default codes are not acceptable, the user may redefine them via this command. All codes must be in the range 0-1FH (i.e., in the ASCII control code region). While this command requires that all eight codes be specified, it will leave unchanged any code whose new value is not in this range, allowing for selective alteration of the codes. ⟨ c1 ⟩ ... ⟨ c8 ⟩ correspond in order to the eight functions listed above. In addition to defining the special codes, this command enables their interpretation by the ALPHA processor (by setting the 40H bit of the ALPHAMODE word).

MODE 2: DON'T IGNORE LINE FEED
CALL: BA
RESPONSE: none

Where MicroAngelo receives command, LINEFEED will not be ignored until a cold start.

MODE 3: NO OPERATION

4.16 RPOINT (Screenware Pak II only)

OPCODE 19 - RPOINT

7	6	5	4	3	2	1	0
1	0	1	1	1	1	M	M

MODE 0: TURN RELATIVE POINT OFF
CALL: BC ⟨ dxh ⟩ ⟨ dxl ⟩ ⟨ dyh ⟩ ⟨ dyl ⟩
RESPONSE: none

MODE 1: TURN RELATIVE POINT ON
CALL: BD ⟨ dxh ⟩ ⟨ dxl ⟩ ⟨ dyh ⟩ ⟨ dyl ⟩
RESPONSE: none

MODE 2: COMPLEMENT RELATIVE POINT
CALL: BE ⟨ dxh ⟩ ⟨ dxl ⟩ ⟨ dyh ⟩ ⟨ dyl ⟩
RESPONSE: none

MODE 3: READ RELATIVE POINT
CALL: BF ⟨ dxh ⟩ ⟨ dxl ⟩ ⟨ dyh ⟩ ⟨ dyl ⟩
RESPONSE: ⟨ val ⟩

These commands are identical to the POINT commands, except that they interpret their parameters as the X and Y relative offset from the current graphics cursor, rather than absolute screen coordinates. As with the POINT commands, the graphics cursor is updated to the new absolute screen location resulting from the relative offset.

4.17 RVECTOR (Screenware Pak II only)

	7	6	5	4	3	2	1	0
OPCODE 16 - RVECTOR	1	1	0	0	0	0	M	M

MODE 0: TURN RELATIVE VECTOR OFF
CALL: C0 ⟨ dxh ⟩ ⟨ dxl ⟩ ⟨ dyh ⟩ ⟨ dyl ⟩
RESPONSE: none

MODE 1: TURN RELATIVE VECTOR ON
CALL: C1 ⟨ dxh ⟩ ⟨ dxl ⟩ ⟨ dyh ⟩ ⟨ dyl ⟩
RESPONSE: none

MODE 2: COMPLEMENT RELATIVE VECTOR
CALL: C2 ⟨ dxh ⟩ ⟨ dxl ⟩ ⟨ dyh ⟩ ⟨ dyl ⟩
RESPONSE: none

MODE 3: NO OPERATION

These commands are identical to the VECTOR commands, except that they interpret their parameters as the X and Y relative offset from the current graphics cursor, rather than absolute screen coordinates. As with the VECTOR commands, the graphics cursor is updated to the new absolute screen location resulting from the relative offset.

4.18 RREGION (Screenware Pak II only)

	7	6	5	4	3	2	1	0
OPCODE 17 - RREGION	1	1	0	0	0	1	M	M

MODE 0: TURN RELATIVE REGION OFF
CALL: C4 ⟨ dx1h ⟩ ⟨ dx1l ⟩ ⟨ dy1h ⟩ ⟨ dy1l ⟩ ⟨ dx2h ⟩ ⟨ dx2l ⟩ ⟨ dy2h ⟩ ⟨ dy2l ⟩
RESPONSE: none

MODE 1: TURN RELATIVE REGION ON
CALL: C5 ⟨ dx1h ⟩ ⟨ dx1l ⟩ ⟨ dy1h ⟩ ⟨ dy1l ⟩ ⟨ dx2h ⟩ ⟨ dx2l ⟩ ⟨ dy2h ⟩ ⟨ dy2l ⟩
RESPONSE: none

MODE 2: COMPLEMENT RELATIVE REGION
CALL: C6 ⟨ dx1h ⟩ ⟨ dx1l ⟩ ⟨ dy1h ⟩ ⟨ dy1l ⟩ ⟨ dx2h ⟩ ⟨ dx2l ⟩ ⟨ dy2h ⟩ ⟨ dy2l ⟩
RESPONSE: none

MODE 3: NO OPERATION

These commands are identical to the region commands, except that they interpret their parameters as the X and Y relative offset from the current graphics cursor, rather than absolute screen coordinates. Typically, to paint a region situated with one corner at the current graphics cursor, RREGION is called with coordinates 0,0,DX,DY, where DX and DY are the size of the desired region. As with the region commands, the graphics cursor is not moved.

4.19 CIRCLE (Screenware Pak II only)

OPCODE 18 - CIRCLE

7	6	5	4	3	2	1	0
1	1	0	0	1	0	M	M

MODE 0: TURN CIRCLE OFF
CALL: C8 ⟨r⟩
RESPONSE: none

Points on the circle of radius ⟨r⟩ centered at the current graphics cursor are turned off. ⟨r⟩ may be any single byte value. Points on the circle out of range in the Y dimension are clipped. Points out of range in the X dimension are wrapped around to the opposite side of the screen.

MODE 1: TURN CIRCLE ON
CALL: C9 ⟨r⟩
RESPONSE: none

Points on the circle of radius ⟨r⟩ centered at the current graphics cursor are turned on. Otherwise, this mode is identical to Mode 0.

MODE 2: COMPLEMENT CIRCLE
CALL: CA ⟨r⟩
RESPONSE: none

Points on the circle of radius ⟨r⟩ centered at the current graphics cursor are complemented. Otherwise, this mode is identical to Mode 0.

MODE 3: NO OPERATION

4.20 FLOOD (Screenware Pak II only)

	7	6	5	4	3	2	1	0
OPCODE 19 - FLOOD	1	1	0	0	1	1	M	M

MODE 0: FLOOD WITH ZEROES
CALL: CC ⟨ xh ⟩ ⟨ xl ⟩ ⟨ yh ⟩ ⟨ yl ⟩
RESPONSE: none

The bordered region containing the interior point specified by the arguments is flooded with zeroes. The region must be completely bordered by zeroes, and its interior must be completely filled with ones for the algorithm to work properly. The region may be any shape, and the starting interior point may be arbitrarily chosen. The flood algorithm is capable in principle of filling virtually any region. In practice, however, the algorithm is limited by stack space, and may not be able to fill an unusually complex region. Generally speaking, the amount of stack storage will relate to the degree of concavity detail in the border. Regions too complex for the 16-level stack will be rare, but can be flooded in pieces if necessary. Additionally, certain narrow 45 degree corridors [i.e., "necks" of complex regions which have a single bit wide, stair-step type of interior] pose logical problems, and cannot be filled because of potential confusion with the region's exterior. Since the flood algorithm checks screen limits, it can also be used to fill the exterior of an object, even though there are no borders at the screen edges.

MODE 1: FLOOD WITH ONES
CALL: CD ⟨ xh ⟩ ⟨ xl ⟩ ⟨ yh ⟩ ⟨ yl ⟩
RESPONSE: none

The region containing the specified interior point is flooded with ones. The region must be completely bordered by ones, and its interior must be completely zeroes. Otherwise, this mode is identical to Mode 0.

MODE 2: FLOOD RELATIVE WITH ZEROES
CALL: CE ⟨ dxh ⟩ ⟨ dxl ⟩ ⟨ dyh ⟩ ⟨ dyl ⟩
RESPONSE: none

This command is identical to Mode 0, except that the starting interior point is specified as a relative offset from the current graphics cursor.

MODE 3: FILL RELATIVE WITH ONES
CALL: CF ⟨ dxh ⟩ ⟨ dxl ⟩ ⟨ dyh ⟩ ⟨ dyl ⟩
RESPONSE: none

This command is identical to Mode 1, except that the starting interior point is specified as a relative offset from the current graphics cursor.

4.21 MACRO (Screenware Pak II only)

	7	6	5	4	3	2	1	0
OPCODE 20 - MACRO	1	1	0	1	0	0	M	M

The macro facility provides for the definition and automatic display of commonly used objects. It is useful both in streamlining the display of such objects, and in higher speed movement of screen objects than would otherwise be possible. The macro storage space can be up to 1536 (decimal) bytes long. Up to 255 distinct macros can be defined in this region, each individual macro being up to 256 bytes long. A macro is any sequence of commands, exactly as they would be sent normally, and is defined by declaring its number (from 0 to 254), then sending the bytes which represent the sequence of MicroAngelo commands to become its "body". Macros are executed by the INVOKE MACRO command described below. The ERASE MACRO command can erase a macro and return its number to the available pool.

The macro facility will issue responses to the Mode 0, 1, and 2 commands below (no response for Mode 3). A response is either 0, to indicate success, or a number from 1 to 6 indicating that a failure occurred and its nature:

RESPONSE	MEANING
0	SUCCESSFUL TRANSACTION
1	DEFINITION ALREADY IN PROGRESS
2	MACRO ALREADY EXISTS
3	MACRO FACILITY SPACE EXHAUSTED
4	NO DEFINITION IN PROGRESS
5	MACRO IS TOO LONG (OVER 256 BYTES)
6	MACRO DOES NOT EXIST

Response bytes must always be read for proper MicroAngelo protocol to proceed.

Because of limited MicroAngelo RAM, the macro processor uses the memory which is also allocated as the user-defined character font, and/or USER code area. While the user can arrange to use all three features simultaneously, care must be taken to manage this 1536 byte area properly. Each macro occupies 2 bytes plus the number of bytes in its body. Each ASCII character in the user-defined character generator area occupies 12 bytes. Thus, by arranging never to use the first N alternate character codes, the user can have a macro storage area of 12*N bytes at the beginning of the 1536 byte area. To assist in the management of this shared memory, the size of the macro definition area can be restricted via the ERASE MACRO command.

MODE 0: START/STOP MACRO DEFINITION
CALL: DO ⟨ n ⟩
or
DO FF
RESPONSE: ⟨ code ⟩

If ⟨ n ⟩ is any value but 0FFH, this command begins the definition of the macro whose reference number will be ⟨ n ⟩. The new definition will not be begun if there is another definition in progress, if ⟨ n ⟩ is already in use as a macro number, or if macro space has been exhausted. The response code indicating success or one of these failures should always be read by the user code, since otherwise the MicroAngelo to host communication port will remain blocked. After having opened the definition, the ADD NEXT MACRO BYTE command is used repetitively to build the macro body. Having built the body, the user instructs the macro facility to end the definition and "install" the macro by calling the START/STOP MACRO DEFINITION command a second time, but with ⟨ n ⟩ = 0FFH. At that time, the macro becomes usable by the INVOKE MACRO command.

MODE 1: ADD NEXT MACRO BYTE
CALL: D1 ⟨ byte ⟩
RESPONSE: ⟨ code ⟩

⟨ byte ⟩ is added to the body of the macro under current definition. A failure code will be returned if there is no definition in progress, if macro space is exhausted or if the macro has become too long. In case of failure, the current definition is closed and partially built macro discarded. The user should always read the response ⟨ code ⟩.

MODE 2: ERASE MACRO OR CLEAR FACILITY
CALL: D2 ⟨ n ⟩
or
D2 FF ⟨ sh ⟩ ⟨ sl ⟩
RESPONSE: ⟨ code ⟩

In the first case, if ⟨ n ⟩ is the number of a defined macro, that macro is deleted from the macro space, and its storage number returned for reuse. If the named macro does not exist, the appropriate error code is returned. In the second case, when ⟨ n ⟩ = 0FFH, the command is interpreted as a macro facility reset directive. In this case, all macros are erased, the number of bytes of the 1536 shared memory region to be allocated to the macro facility is specified by ⟨ sh ⟩, ⟨ sl ⟩, which should be in the range 0-1536. After this command, any attempt to build macros beyond this limit will return a failure code. The macro facility powers up in a reset condition, with all 1536 bytes allowed for macro definitions. Both forms of this command return a condition ⟨ code ⟩, which should always be read by the user.

MODE 3: INVOKE MACRO
CALL: D3 ⟨ n ⟩
RESPONSE: none

The macro whose number is ⟨ n ⟩ is invoked, i.e., its body is fed to the command interpreter just as if it were coming straight from the user. If there is no macro number ⟨ n ⟩, A NO OPERATION results. While the macro's invocation itself may cause a reponse to be generated, the INVOKE MACRO command itself never returns a success or failure response. When the invoked macro's body has been completely read, Screenware Pak II reverts to its normal command loop. However, since there are cases where it may be convenient for one macro to invoke other macros, Screenware Pak II allows a macro invocation nesting depth of 8. Nestings beyond this depth are ignored. When a nested macro completes, control is resumed in the previous (calling) macro, and so forth until the normal command processor is again active. Naturally, care should be exercised in defining macros, since, if a macro's body is incorrect, it may throw Screenware Pak II and the user out of logical touch with each other, just as would happen in any improperly formed direct command sequence.

Macros will typically rely heavily on the new relative cursor, point, vector, and region commands, and on the new circle and flood commands. Generally, the strategy for writing a macro is to work from the current cursor, and ensure that the cursor is left either where it was originally, or at some meaningful place for the next macro (if there will be a sequence of them, or, if they have been nested) to pick up. For macros that are capable of moving objects at relatively high speed on the screen, use only the complement mode of all drawing commands, so that the first invocation of the macro will draw, the second erase.

The following example illustrates how to set up, then use a macro. Suppose the goal is to define a macro that will draw a triangle with lower left vertex at the current graphics cursor, flood the triangle's interior with 1's, draw a circle of 0's inside the triangle, flood the circle's interior with 0's, then leave the graphics cursor at the lower left vertex of the triangle where it began. The sequence of commands that are to form the macro's body is therefore:

RVECTOR	+25 +50	draw first side of triangle
RVECTOR	+25 −50	draw second side
RVECTOR	−50 0	draw third
RFLOODO	+1 +1	flood triangle interior with ones
RGRAPHC	+25 +25	move to triangle center point
CIRCLEZ	15	draw circle with zeroes
RFLOODZ	0 0	flood circle interior with zeroes
RGRAPHC	−25 −25	return cursor to starting point

Hence, the sequence which defines this sequence as, say, macro 0 is:

D0 00	start macro 0 definition
D1 C1 D1 00 D1 19 D1 00 D1 32	send first vector command
D1 C1 D1 00 D1 19 D1 FF D1 CE	send second vector command
D1 C1 D1 FF D1 CE D1 00 D1 00	send third vector command
D1 CF D1 00 D1 01 D1 00 D1 01	send triangle flood command
D1 B4 D1 00 D1 19 D1 00 D1 19	send rel cursor move command
D1 C8 D1 0A	send circle command
D1 CE D1 00 D1 00 D1 00 D1 00	send circle flood command
D1 B4 D1 FF D1 E7 D1 FF D1 E7	send rel cursor move command
D0 FF	terminate and install macro

This macro can then be invoked by calls of the form:

D3 00	invoke macro number 0 at current graphics cursor

System Details

MicroAngelo can be effectively used without a knowledge of the information in this section. However, if you wish to install a lightpen, read the subsection entitled "Connecting a Light Pen". If you plan on augmenting Screenware Pak I or Screenware Pak II with additional software, read this entire section.

5.1 The MicroAngelo Memory Map

Unless you plan on sending user code across to MicroAngelo via the MEMORY command, you need not be concerned with the internal memory map of a Screenware Pak. However, in order to install and interface user-defined graphics code, it is important to understand how a Screenware Pak uses the MicroAngelo memory space.

REGION	USE
0000-0FFF	Screenware Pak I in EPROM
0000-17FF	Screenware Pak II in EPROM
1000-7FFF	Unimplemented (SW PK I)
1800-7FFF	Unimplemented (SW PK II)
8000-FFFF	Read-write memory, subdivided as follows:
8000-F7FF	Visible display
F800-F8BF	2 and one-half visible scan lines (which should be kept blanked)
F8C0-F93F	Screenware system stack
F940-FF3F	User-defined character generator, or user code area
FF40-FFFF	Screenware working RAM

If the alternate character set is defined and used, there is no space for user code. If, however, the alternate character set is not used (or if only a portion is used), the region F940-FF3F (1.5K bytes) can be used in whole or in part for user code.

User code should not make any unusual alterations to the system stack, nor should it alter any location in the FF40-FFFF region.

5.2 Defining the Alternate Character Set

The alternate character set resides in the F940-FF3F region of MicroAngelo's RAM. Each character symbol occupies 12 bytes, top scan line first. Thus, the region F940-F94B holds the symbol for ASCII code 0, with the top scan line at F940, the bottom line at F94B. Within each byte, the low-order six bits define the pixels across a scan line of the character. The CHARACTER and ALPHAMODE commands allow you to select this alternate character set, or toggle between the alternate and standard sets.

The alternate character set can be defined all at once by the Pak II command LOAD DEFAULT CHARACTER SET (Section 4.7), or by depositing (via the MEMORY command) all 128*12 bytes starting at location F940. (If not all 128 symbols need to be defined, you need not send the entire set, and can use any remaining space for user code.) Alternatively, symbols for individual ASCII codes can be defined using the CHARACTER command's Mode 2.

As an example, suppose you wish initially to define alternate symbols for ASCII codes 0-63 (the lower half of the character set). To do this, you say:

```
A7      deposit 64*12 bytes at F940
03      64*12 = 300 (hex)
00
F9      location F940
40      send the 768 (decimal) bytes
```

Suppose then at a later time you wish to alter the symbol for ASCII code 7. Then you say:

```
9A      define individual symbol via CHARACTER
07      ASCII code 7
....    send the twelve bytes, top scan line first
```

5.3 Interfacing Onboard User Code to The Screenware

User code installed in the MicroAngelo RAM will probably need to interact with the Screenware software primitives. Appendix 3, "Screenware Pak I User Entry Points" and Appendix 4, "Screenware Pak II Entry Points" gives entry point addresses and calling conventions for the various user-callable Screenware Pak I and Pak II functions.

5.4 The MicroAngelo Physical I/O Ports

When running your own software in the MicroAngelo memory, you may occasionally wish to bypass the Screenware software and interact directly with the MicroAngelo hardware. When interacting directly with the hardware, user code has access to the following information as Z80A I/O ports 0-3:

PORT	MODE	FUNCTION
0	Input	Data Port, from host
	Output	Data Port, to host
1	Input	Status Bits: 0 (rightmost bit) host-to-MicroAngelo data buffer is full 1 MicroAngelo-to-host data buffer is full 2 Light Pen strobe has fired 3 Screen Figure/Ground status 4-7 Unused
	Output	The rightmost bit sets the screen figure/ground ("0" for light on dark, "1" for dark on light). All other bits are unused.
2	Input	Light Pen horizontal counter latch (left of screen is count 0, right of screen is count 255), accurate to 2 pixels
	Output	Unused
3	Input	Light Pen vertical counter latch (top of screen is count 0, bottom of screen is count 239), accurate to 2 scan lines. Reading this port also resets the light pen interface, allowing it to trigger on the next light pen strobe. (See the section entitled "Connecting a Light Pen" for more discussion.)
	Output	Unused

5.5 Interrupts

There are four potential interrupt sources for the MicroAngelo's Z80A:

DFHI (Data From Host) - the host has just written a byte to the MicroAngelo Data Port

DTHI (Data To Host) - the host has just read a byte from the Data Port

LPI (Light Pen) - the light pen has just fired

RTI (Real-Time) - the 60 hz interval timer has just fired

The first three interrupt sources are connectable as maskable Z80A interrupts. The Real-Time Interrupt, when enabled by a hardware jumper, will generate a Z80 NMI (non-maskable interrupt) every 1/60 second.

5.5.1 Enabling/Disabling the Maskable Interrupts

As shipped, only the LPI and DFHI are physically enabled. The DTHI has been disabled by removing U59 pin 9 from its socket. Reinsert this pin to enable the DTHI. [Doing so will not logically interfere with the Screenware's logical operation. However, it will slow the software down somewhat when sending responses back to the host.]

To disable the DFHI, remove U59 pin 10 from its socket. To disable the LPI, remove U59 pin 13 from its socket. [Do not disable these, however, unless you are installing a completely new operating system in EPROM! The Screenware assumes that these two interrupts are enabled, and will not run properly with them disabled.] See the UTILITY command [Mode 1] for a description of the logical user interface to these three maskable interrupts.

5.5.2 Enabling the Real-Time Interrupt

The RTI non-maskable interrupt can be enabled by scratching through the default trace between holes 2 and 3 of J3, and jumpering holes 1 and 2 together. After this procedure, a non-maskable interrupt will be generated every 1/60 second. See the UTILITY command [Mode 2] for a description of the logical user interface to this non-maskable interrupt.

It should be noted that with the RTI connected, there is a very remote possibility that MicroAngelo will not power up correctly. Immediately after beginning, the Screenware software stores a specific code in one byte of its read-write memory to remind itself that RTI interrupts are logically disabled. If, however, an RTI occurs in the several microseconds between powering on and storing this disabling code, and if the MicroAngelo memory randomly happens to power up with this special code already present in the RTI enabling byte [**very** unlikely], then the Screenware will erroneously branch to what it thinks is the user-defined RTI handling code. This, of course, would cause the system to lose control. To be absolutely certain that MicroAngelo has powered up correctly with the RTI enabled, use the MEMORY command to examine the RTI logical status byte at location FFC5 immediately after system power-on [i.e., put this in your cold-start initialization code]. If this byte is not OCCH, keep resetting MicroAngelo [over the Control Port] until it is. Then reset the system one final time. [The chance of a bad power-up because of these circumstances is quite remote. You can therefore get along without these procedures for all but the most critical applications.]

5.5.3 Connecting Host-Side Interrupts

Jumper J5 on the MicroAngelo board can be set so that the **host** will be interrupted whenever MicroAngelo reads or writes a byte over the Data Port. J5 Pin 5 goes to logic "0" when MicroAngelo writes a byte to the host. J5 Pin 10 goes to logic "0" when MicroAngelo reads a byte from the host [i.e., when the host can write another byte to MicroAngelo]. J5 Pins 6, 1, 7, 2, 8, 3, 9, 4 connect to the S100 bus vectored interrupt lines [S100 fingers 4-11, respectively]. By jumpering J5 Pin 5 and/or J5 Pin 10 to these vectored interrupt lines, you can route these two interrupt signals to the host CPU, if it is equipped to process them. Doing so permits the host operating system software to support an interrupt-driven protocol with MicroAngelo.

5.6 Connecting a Light Pen

Connector JA at the top right corner of the board is the Light Pen Connector. Pin1 accepts the rising edge triggered Light Pen Strobe, Pin 2 is the Light Pen Ground connection, Pin 3 accepts the active high Light Pen Enable, and Pin 4 is a regulated +5 volt, 100 ma power source for the light pen. When Pin 3 is a logic "1" and a positive edge occurs on Pin 1, the light pen hardware latch captures the display counters to record the X-Y location of the light pen. Further positive edges at Pin 1 will not be honored until the Screenware software (or user software) reads the counter value from the light pen hardware latch. As shipped, both Pin 1 and Pin 3 are pulled down to logic "0" (by resistors R18, R19, respectively) in the absence of a light pen.

If you wish to connect a light pen that generates both the strobe and enable signals, simply connect all 4 pins as described. (If your light pen is of the low-power type, you may have to remove R18 and R19, since these pull-down resistors may present an excessive current drain to the light pen.) If your light pen has no enable line, jumper Pin 3 and Pin 4 together to enable the light pen permanently.

See the LIGHTPEN command and the section entitled "Interrupts" for descriptions of the logical light pen interface and light pen interrupts.

5.7 Summary of Hardware Jumper Options and Connectors

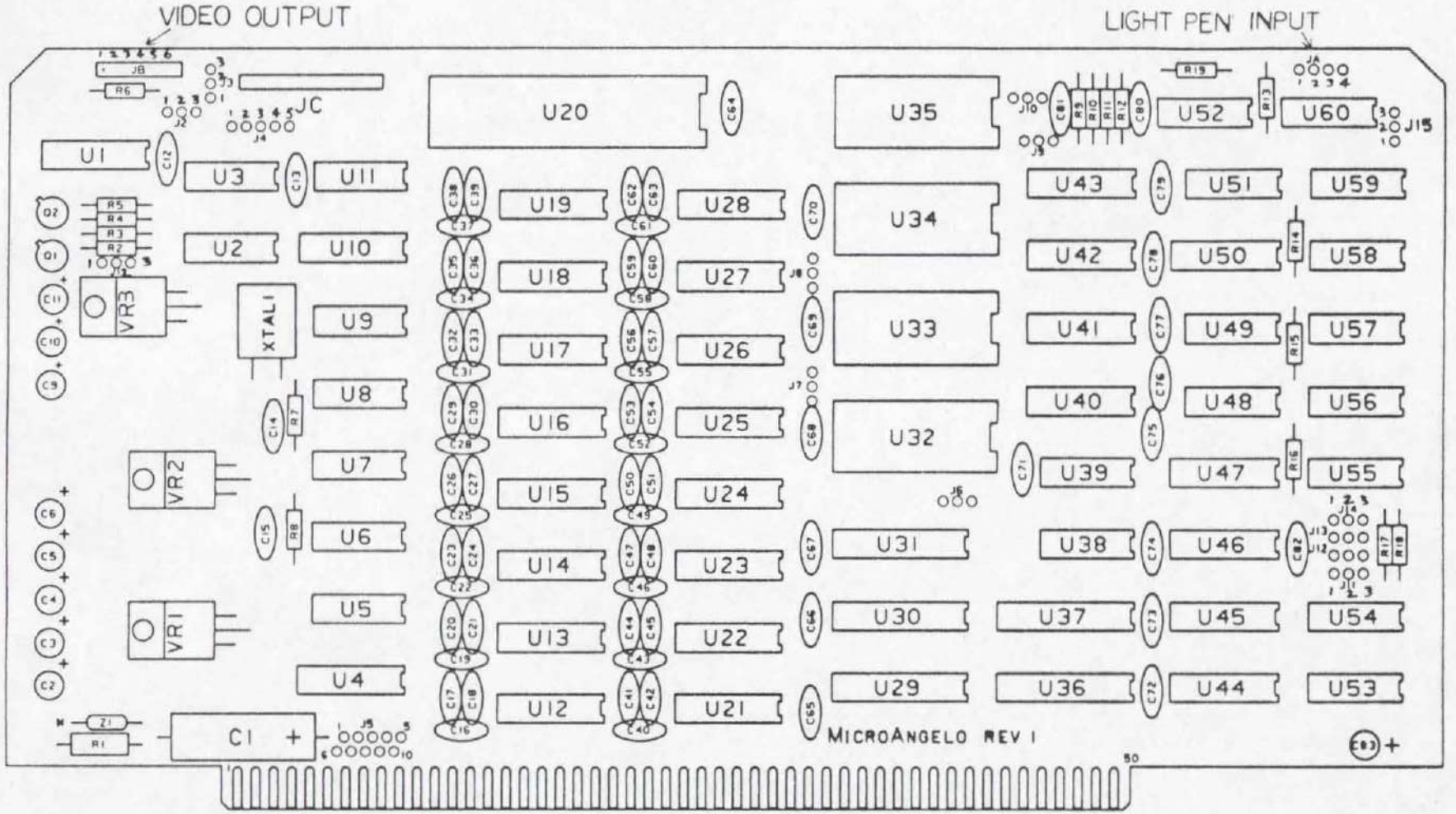
There are 15 jumpers and 3 connectors on the MicroAngelo board. The tables and diagram below summarize and describe these. For most applications there will be no need to alter any jumpers. Default settings are indicated with asterisks.

5.7.1. Hardware Jumpers

NAME	PINS	FUNCTION
J1	1-2*	Select 480 visible scan lines
	2-3	Select 448 visible scan lines
		[Note that all Screenware software assumes that there are 480 visible lines. If you select the 448 option, you must assume responsibility for managing the display screen.]
J2	1-2*	Select 4 mhz Z80A operation
	2-3	Select 5 mhz Z80A operation
		A Z80A can usually run at 5 mhz. If you want to increase the speed of the system, select this option.
J3	1-2	Enable 60 hz Real-Time Interrupt (RTI)
	2-3*	Disable 60 hz RTI
		See section entitled "Interrupts"
J4		Holes 6, 1, 7, 2, 8, 3, 9, 4 connect to S100 bus fingers 4, 5, 6, 7, 8, 9, 10, 11 respectively. [These are the vectored interrupt lines.] The signal at hole 5 is the inverted DTHI interrupt, the signal at hole 10 is the true DFHI signal [see the section entitled "Interrupts"]. By connecting DTHI-inverted and/or DFHI-true to vectored interrupt lines, you can arrange for your host system to be interrupted whenever MicroAngelo reads the byte last sent from the host, or sends a byte to the host. [See the section entitled "Interrupts".] The board is shipped with neither interrupt source connected.
J6-J10		[These jumpers will allow future EPROM upgrade to an 8K operating system]
J11-J14	1-2	Select port address bit = "0"
	2-3*	Select address bit = "1"
		These four jumpers map the two parallel ports over which you communicate with MicroAngelo. See the section entitled "Changing the Port Addresses".
J15	1-2*	Enable DFHI and DTHI interrupts
	2-3	Disable DFHI and DTHI interrupts
		This jumper can cause the MicroAngelo Z80A to be interrupted by communications activities with the host, as described in the section entitled "Interrupts"

5.7.2. Hardware Connectors

NAME	PIN	FUNCTION
JA	1	Light Pen Strobe. A positive-going signal on this pin causes the Screenware software to update [LX, LY], the light pen coordinates
	2	Light Pen Ground
	3	Light Pen Enable. A logic "1" on this pin physically enables the Light Pen Strobe. It is typically fed by the activation switch in the light pen.
	4	+5 volt, 100 ma power source for light pen
JB	1	Composite Video. Connect a composite video TV monitor to this pin and Pin 2.
	2	Composite Video Ground
	3	TTL Video. Connect a direct-drive video monitor to this and Pins 4, 5, 6
	4	Direct-Drive Ground
	5	Direct-Drive Horizontal Sync
	6	Direct-Drive Vertical Sync
JC	1-20	[Reserved for color interface]


VIDEO OUTPUT
LIGHT PEN INPUT
JB
JC
U1
U2
U3
U4
U5
U6
U7
U8
U9
U10
U11
U12
U13
U14
U15
U16
U17
U18
U19
U20
U21
U22
U23
U24
U25
U26
U27
U28
U29
U30
U31
U32
U33
U34
U35
U36
U37
U38
U39
U40
U41
U42
U43
U44
U45
U46
U47
U48
U49
U50
U51
U52
U53
U54
U55
U56
U57
U58
U59
U60
VR1
VR2
VR3
XTAL1
C1
MICROANGELO REV 1

5.8 Adapting MicroAngelo to Non-S100 Bus Systems

Interfacing MicroAngelo to non-S100 bus systems is relatively straightforward because of its simple parallel port connection to the host system. Specifically, MicroAngelo requires the following S100 bus connections:

S100 PIN	NAME	FUNCTION
1,51	+8	Unregulated +8 volt power (2 amps)
50,100	GND	Ground
2	+18	Unregulated +18 volt power (1 amp)
52	−18	Unregulated −18 volt power (100 ma)
90	DO7	Outbound data line 7
40	DO6	Outbound data line 6
39	DO5	Outbound data line 5
38	DO4	Outbound data line 4
89	DO3	Outbound data line 3
88	DO2	Outbound data line 2
35	DO1	Outbound data line 1
36	DO0	Outbound data line 0
43	DI7	Inbound data line 7
93	DI6	Inbound data line 6
92	DI5	Inbound data line 5
91	DI4	Inbound data line 4
42	DI3	Inbound data line 3
41	DI2	Inbound data line 2
94	DI1	Inbound data line 1
95	DI0	Inbound data line 0
83	A7	Address line 7
82	A6	Address line 6
29	A5	Address line 5
30	A4	Address line 4
80	A1	Address line 1
79	A0	Address line 0
46	SINP	Input request
45	SOUT	Output request
78	PDBIN	Input strobe
77	PWR-BAR	Output strobe

The data input and output lines can be tied together to form one 8 line bidirectional data bus. Commands and data are written to MicroAngelo on the coincidence of SOUT = "1", PWR-BAR = "0" and Board Select. Responses and status flags are read from MicroAngelo on the coincidence of SINP = "1", PDBIN = "1" and Board Select. Board Select occurs when address lines A7-A4 match the settings of jumpers J14-J11 and A1 = "0". On a read or write operation, address line A0 determines whether the Data Port or Control Port is selected.

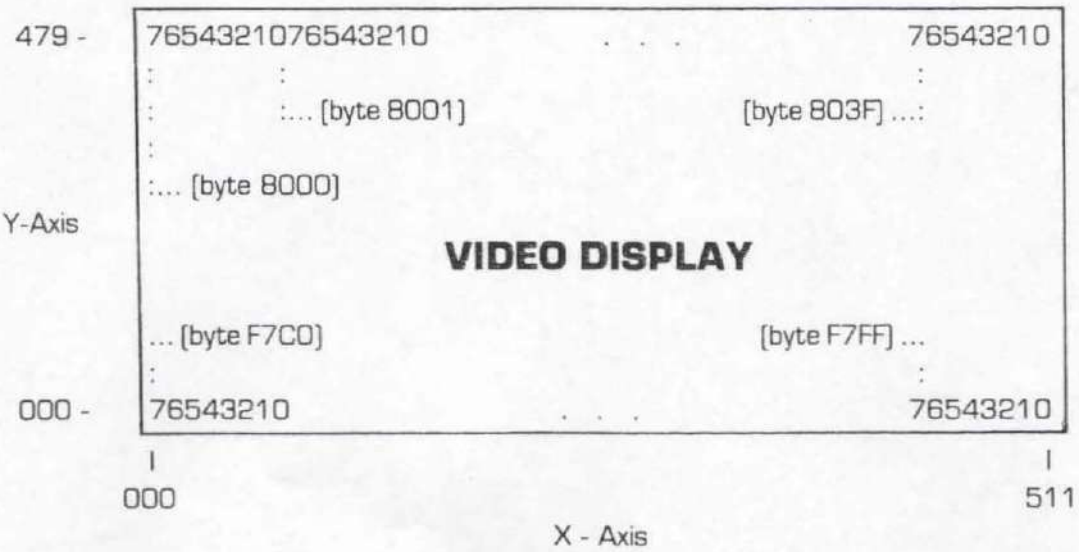
For a stand-alone environment in which MicroAngelo will be powered by its own power supply and will be unrelated to its host's address space, a simple bidirectional parallel port interface can be implemented as follows:

1. Tie the data inbound and outbound lines together and route them to the host as the 8 bit bidirectional parallel I/O port.
2. Tie A7, A6, A5, A4 permanently high (to match the default jumpers J14-J11), and tie A1 permanently low.
3. Tie PDBIN permanently high, PWR-BAR permanently low.
4. Route A0 to the host as the Data/Control Port select line (i.e., MicroAngelo looks like 2 logical I/O ports over one physical I/O port connection).
5. Route SINP and SOUT to the host as the input and output command lines.

Using this 12 conductor logical interface to the host (8 data lines, A0, SINP, SOUT, ground), MicroAngelo becomes a stand-alone graphics computer compatible with virtually any type of host system. By connecting the interrupt lines as described in the section entitled "Interrupts" and routing them to the host, the interface can also support a full interrupt protocol.

5.9 Bit Mapping of Display RAM to Video Screen

The address space of the MicroAngelo from locations 8000 to 0F7FF is RAM memory that is displayed on the video screen. Each of the 245, 670 bits within this range appears as a single picture element (pixel) on the screen. These bits are mapped onto the screen in a predefined way by the MicroAngelo hardware. The top leftmost point on the display is the most significant bit of the byte stored at location 8000. The point immediately to its right is the 2nd most significant bit of the byte at 8000. This continues for all the bits in byte 8000 and then proceeds on across the screen with the bits from byte 8001, then 8002, 8003, etc. for a total of 64 bytes. The second display row then begins with the most significant bit from the byte at location 8040. The bottom rightmost bit of the display is the least significant bit of the byte at location 0F7FF. The MEMORY commands "examine" and "deposit" can be used for experimenting with the direct modification of the video display.



Software Interface Examples

6. Software Interface Examples

Send and receive all bytes in these examples using the code shown in the section entitled "The Software Interface".

6.1. Graphics: Clear Screen, Draw Triangle, Embed in Region

```
88      clear the screen
84      set the graphics cursor to (128, 128) decimal
00
80
00
80
91      draw vector to (256,384)
01
00
01
80
91      draw a vector to (384,128)
01
80
00
80
91      draw a vector to (128,128)
00
80
00
80
96      embed triangle in region by complementing
00      make the region corners (64,64) and (448,448)
40
00
40
01
C0
01
C0
```

6.2. Turn On and Read the Tracking Cross

```
9D      turn the tracking cross on at screen center
01      X = 256
00
00      Y = 242
F2
....    [wait for user to drag it to destination, then type a key on the host
        keyboard]
9E      read the location
....    [The Screenware will send the coordinates as four response bytes
        which you then read.]
```

6.3. Write a Message Around the Border of a Square

This code writes the characters "MicroAngelo!" in a box shape (i.e., "Mic" is on the top, "roA" is on the right side going down, "nge" is upside-down from right to left on the bottom, and "lo!" is on the left side going up. Characters are double size and reversed figure/ground.

```
84      move the graphics cursor to the screen center
01
00
00
F2
99      set graphics character mode for top characters
0C      reversed figure/ground, double size
98      print "M"
4D
98      print "i"
69
98      print "c"
63
99      select new orientation
0F      90 degrees cw, top to bottom
98      print "r"
72
98      print "o"
6F
98      print "A"
41
99      select new orientation
0D      upside-down, right to left
98      print "n"
6E
98      print "g"
67
98      print "e"
65
99      select new orientation
0E      90 degrees ccw, bottom to top
98      print "l"
6C
98      print "o"
6F
98      print "!"
21
```

6.4. Underlining in Dumb Terminal Mode

The following code prints the message "Hello there" by switching into and out of ALPHA Underline Mode for a moment.

```
48      print "H"
65      print "e"
6C      print "l"
6C      print "l"
6F      print "o"
20      print space
80      give ALPHAMODE command to start underlining
02      second-from-right bit governs underlining
74      print "t"
68      print "h"
65      print "e"
72      print "r"
65      print "e"
80      turn off underlining
00
```

6.5. Sample BASIC Interface

Most high level graphics software is best developed in a higher level language. To illustrate how to drive MicroAngelo from North Star BASIC, four functions, FNO, FNI, FNS and FNR are shown below. FNO will wait for the Control Port to indicate a read-to-send condition, then send a single given byte to MicroAngelo. FNI will await a single byte MicroAngelo response, then return it as the functional value. FNS will send a 16 bit quantity (e.g., a coordinate or address), high order byte first, by two calls on FNO. FNR will assemble a 16 bit (two byte) response from MicroAngelo and return the 16 bit quantity as its functional value. In these examples it is assumed that the Control Port is F1 and the Data Port is F0 (241, 240 decimal, respectively). If you have changed the port addresses, substitute these with the appropriate port number.

```
10100  REM SEND A BYTE TO MICROANGELO
10200  DEF FNO(X)
10300  I = INP(241)
10400  IF I < > 2*INT(I/2)THEN 10300
10500  OUT 240, X
10600  RETURN 0
10700  FNEND

10800  REM READ A BYTE FROM MICROANGELO
10900  REM (CALL WITH A DUMMY PARAMETER)
11000  DEF FNI(X)
11100  I = INT(INP(241)/2)
11200  IF I = 2*INT(I/2)
11300  RETURN INP(240)
11400  FNEND

11500  REM SEND A 16 BIT QUANTITY TO MICROANGELO
11600  DEF FNS(X)
11700  I = FNO (INT(X/256))
11800  I = FNO(X-256*INT(X/256))
11900  RETURN 0
12000  FNEND

12100  REM READ A 16 BIT QUANTITY FROM MICROANGELO
12200  REM (CALL WITH A DUMMY PARAMETER)
12300  DEF FNR(X)
12400  Q = FNI(0)
12500  RETURN 256*Q + FNI(0)
12600  FNEND
```

6.6 Interfacing to FORTRAN

The following subroutines are five examples of FORTRAN routines to direct MicroAngelo.

```
C
C
C  ***        output a byte to MicroAngelo
C
             subroutine maout (ibyte)
10           if (inp(241). and.1) go to 10
             call out (240, ibyte)
             return
             end

C
C
C  ***        move graphics cursor to cx, cy
C
             subroutine cursor (cx, cy)
             call maout (84H)
             call coord (cx, cy)
             return
             end

C
C
C  ***        plot a point at cx, cy
C
             subroutine point (cx, cy)
             call maout (8DH)
             call coord (cx, cy)
             return
             end

C
C
C  ***        draw a vector to cx, cy
C
             subroutine vector (cx, cy)
             call maout (91H)
             call coord (cx, cy)
             return
             end

C
C
C  ***        output a 16 bit X and a 16 bit Y coordinate to MicroAngelo
C
             subroutine coord (cx, cy)
             ic = cx/256.0
             call maout (ic)
             ic = int (cx-ic*255.9)
             call maout (ic)
             ic = cy/256.0
             call maout (ic)
             ic = int(cy-ic*255.9)
             call maout (ic)
             return
             end
```

Appendix 1 - Summary of Screenware Commands

HEX	DEC	OCT	CALL/RESPONSE	FUNCTION
ALPHAMODE				
80	128	200	C: ⟨ mode ⟩ R:none	Set Alpha Mode Bits
81	129	201	C: ⟨ row ⟩ ⟨ col ⟩ R: none	Position Alpha Cursor
82	130	202	C: none R: ⟨ row ⟩ ⟨ col ⟩	Read Alpha Cursor
83	131	203	C: ⟨ n ⟩ R: none	Set Alpha Scroll
GCURSOR				
84	132	204	C: ⟨ xh ⟩ ⟨ xl ⟩ ⟨ yh ⟩ ⟨ yl ⟩ R: none	Set Graphics Cursor
85	133	205	C: none R: ⟨ xh ⟩ ⟨ xl ⟩ ⟨ yh ⟩ ⟨ yl ⟩	Read Graphics Cursor
86	134	206	C: none R: none	Set [CX, CY] to [AX, AY]
87	135	207	C: none R: none	Set [CX, CY] to [TX, TY]
SCREEN				
88	136	210	C: none R: none	Clear Screen
89	137	211	C: ⟨ fg ⟩ R: none	Set Screen Figure/Ground
8A	138	212	C: none R: none	Toggle Screen Figure/Ground
8B	139	213	C: none R: ⟨ fg ⟩	Read Screen Figure/Ground
POINT				
8C	140	214	C: ⟨ xh ⟩ ⟨ xl ⟩ ⟨ yh ⟩ ⟨ yl ⟩ R: none	Turn Point Off
8D	141	215	C: ⟨ xh ⟩ ⟨ xl ⟩ ⟨ yh ⟩ ⟨ yl ⟩ R: none	Turn Point On
8E	142	216	C: ⟨ xh ⟩ ⟨ xl ⟩ ⟨ yh ⟩ ⟨ yl ⟩ R: none	Complement Point
8F	143	217	C: ⟨ xh ⟩ ⟨ xl ⟩ ⟨ yh ⟩ ⟨ yl ⟩ R: ⟨ val ⟩	Read Point
VECTOR				
90	144	220	C: ⟨ xh ⟩ ⟨ xl ⟩ ⟨ yh ⟩ ⟨ yl ⟩ R: none	Turn Vector Off
91	145	221	C: ⟨ xh ⟩ ⟨ xl ⟩ ⟨ yh ⟩ ⟨ yl ⟩ R: none	Turn Vector On
92	146	222	C: ⟨ xh ⟩ ⟨ xl ⟩ ⟨ yh ⟩ ⟨ yl ⟩ R: none	Complement Vector

REGION

94	148	224	C: ⟨x1h⟩ ⟨x1l⟩ ⟨y1h⟩ ⟨y1l⟩ ⟨x2h⟩ ⟨x2l⟩ ⟨y2h⟩ ⟨y2l⟩ R: none	Turn Region Off
95	149	225	C: ⟨x1h⟩ ⟨x1l⟩ ⟨y1h⟩ ⟨y1l⟩ ⟨x2h⟩ ⟨x2l⟩ ⟨y2h⟩ ⟨y2l⟩ R: none	Turn Region On
96	150	226	C: ⟨x1h⟩ ⟨x1l⟩ ⟨y1h⟩ ⟨y1l⟩ ⟨x2h⟩ ⟨x2l⟩ ⟨y2h⟩ ⟨y2l⟩ R: none	Complement Region

CHARACTER

98	152	230	C: ⟨c⟩ R: none	Plot Graphics Character
99	153	231	C: ⟨mode⟩ R: none	Set Graphics Character Mode
9A	154	232	C: ⟨asc⟩ ⟨s11⟩ ... ⟨s0⟩ R: none	Define Alternate Character
9B	155	233	C: none R: none	Load Default Character Set

LIGHTPEN

9C	156	234	C: none R: none	Turn Tracking Cross Off
9D	157	235	C: ⟨xh⟩ ⟨xl⟩ ⟨yh⟩ ⟨yl⟩ R: none	Turn Tracking Cross On
9E	158	236	C: none R: ⟨xh⟩ ⟨xl⟩ ⟨yh⟩ ⟨yl⟩	Read Tracking Cross
9F	159	237	C: none R: 00 or 01 ⟨xh⟩ ⟨xl⟩ ⟨yh⟩ ⟨yl⟩	Read Light Pen

CROSSHAIRS

A0	160	240	C: none R: none	Turn Crosshairs Off
A1	161	241	C: ⟨xh⟩ ⟨xl⟩ ⟨yh⟩ ⟨yl⟩ R: none	Draw Crosshairs
A2	162	242	C: none R: ⟨xh⟩ ⟨xl⟩ ⟨yh⟩ ⟨yl⟩	Read Crosshairs
A3	163	243	C: none R: none	Draw Crosshairs at [CX, CY]

MEMORY

A4	164	244	C: none R: ⟨b1⟩ ... ⟨b7800⟩	Dump Screen
A5	165	245	C: ⟨b1⟩ ... ⟨b7800⟩ R: none	Load Screen
A6	166	246	C: ⟨nh⟩ ⟨nl⟩ ⟨ah⟩ ⟨al⟩ R: ⟨b1⟩ ... ⟨bn⟩	Examine Memory Block
A7	167	247	C: ⟨nh⟩ ⟨nl⟩ ⟨ah⟩ ⟨al⟩ ⟨b1⟩ ... ⟨bn⟩ R: none	Deposit Memory Block

UTILITY

A8	168	250	C: ⟨ ah ⟩ ⟨ al ⟩ R: none	Set User Command Address
A9	169	251	C: ⟨ ah ⟩ ⟨ al ⟩ ⟨ imask ⟩ ⟨ ih ⟩ ⟨ il ⟩ R: none	Call User Code
AA	170	252	C: AA 00 or AA 01 ⟨ ah ⟩ ⟨ al ⟩ R: none	Switch Real-Time Interrupts
AB	171	253	C: none R: none	Force Cold Start

USER

AC	172	254	C: [user defined] R: [user defined]	User
AD	173	255	C: [user defined] R: [user defined]	User
AE	174	256	C: [user defined] R: [user defined]	User
AF	175	257	C: [user defined] R: [user defined]	User

TEST

B0	176	260	C: ⟨ blocks ⟩ R: ⟨ cksum ⟩	Test EPROM
B1	177	261	C: none R: 0 or 1 ⟨ ah ⟩ ⟨ al ⟩ ⟨ eb ⟩ ⟨ fb ⟩	Test RAM
B2	178	262	C: none R: none	ALPHA Test
B3	179	263	C: ⟨ s ⟩ ⟨ i ⟩ ⟨ n ⟩ R: none	Munching Squares

RGRAPHC

B4	180	264	C: ⟨ dxh ⟩ ⟨ dxl ⟩ ⟨ dyh ⟩ ⟨ dyl ⟩ R: none	Set Relative Graphics Cursor

SPLITSCR

B8	184	270	C: ⟨ l ⟩ R: none	Set ALPHA Screen Size
B9	185	271	C: ⟨ c1 ⟩ ... ⟨ c8 ⟩ R: none	Define ALPHA Control Codes

RPOINT

BC	188	274	C: ⟨ dxh ⟩ ⟨ dxl ⟩ ⟨ dyh ⟩ ⟨ dyl ⟩ R: none	Turn Relative Point Off
BD	189	275	C: ⟨ dxh ⟩ ⟨ dxl ⟩ ⟨ dyh ⟩ ⟨ dyl ⟩ R: none	Turn Relative Point On
BE	190	276	C: ⟨ dxh ⟩ ⟨ dxl ⟩ ⟨ dyh ⟩ ⟨ dyl ⟩ R: none	Complement Relative Point
BF	191	277	C: ⟨ dxh ⟩ ⟨ dxl ⟩ ⟨ dyh ⟩ ⟨ dyl ⟩ R: ⟨ val ⟩	Read Relative Point

RVECTOR

C0	192	300	C: ⟨ dxh ⟩ ⟨ dxl ⟩ ⟨ dyh ⟩ ⟨ dyl ⟩ R: none	Turn Relative Vector Off
C1	193	301	C: ⟨ dxh ⟩ ⟨ dxl ⟩ ⟨ dyh ⟩ ⟨ dyl ⟩ R: none	Turn Relative Vector On
C2	194	302	C: ⟨ dxh ⟩ ⟨ dxl ⟩ ⟨ dyh ⟩ ⟨ dyl ⟩ R: none	Complement Relative Vector

RREGION

C4	196	304	C: ⟨ dx1h ⟩ ⟨ dx1l ⟩ ⟨ dy1h ⟩ ⟨ dy1l ⟩ ⟨ dx2h ⟩ ⟨ dx2l ⟩ ⟨ dy2h ⟩ ⟨ dy2l ⟩ R: none	Turn Relative Region Off
C5	197	305	C: ⟨ dx1h ⟩ ⟨ dx1l ⟩ ⟨ dy1h ⟩ ⟨ dy1l ⟩ ⟨ dx2h ⟩ ⟨ dx2l ⟩ ⟨ dy2h ⟩ ⟨ dy2l ⟩ R: none	Turn Relative Region On
C6	198	306	C: ⟨ dx1h ⟩ ⟨ dx1l ⟩ ⟨ dy1h ⟩ ⟨ dy1l ⟩ ⟨ dx2h ⟩ ⟨ dx2l ⟩ ⟨ dy2h ⟩ ⟨ dy2l ⟩ R: none	Complement Relative Region

CIRCLE

C8	200	310	C: ⟨ r ⟩ R: none	Turn Circle Off
C9	201	311	C: ⟨ r ⟩ R: none	Turn Circle On
CA	202	312	C: ⟨ r ⟩ R: none	Complement Circle

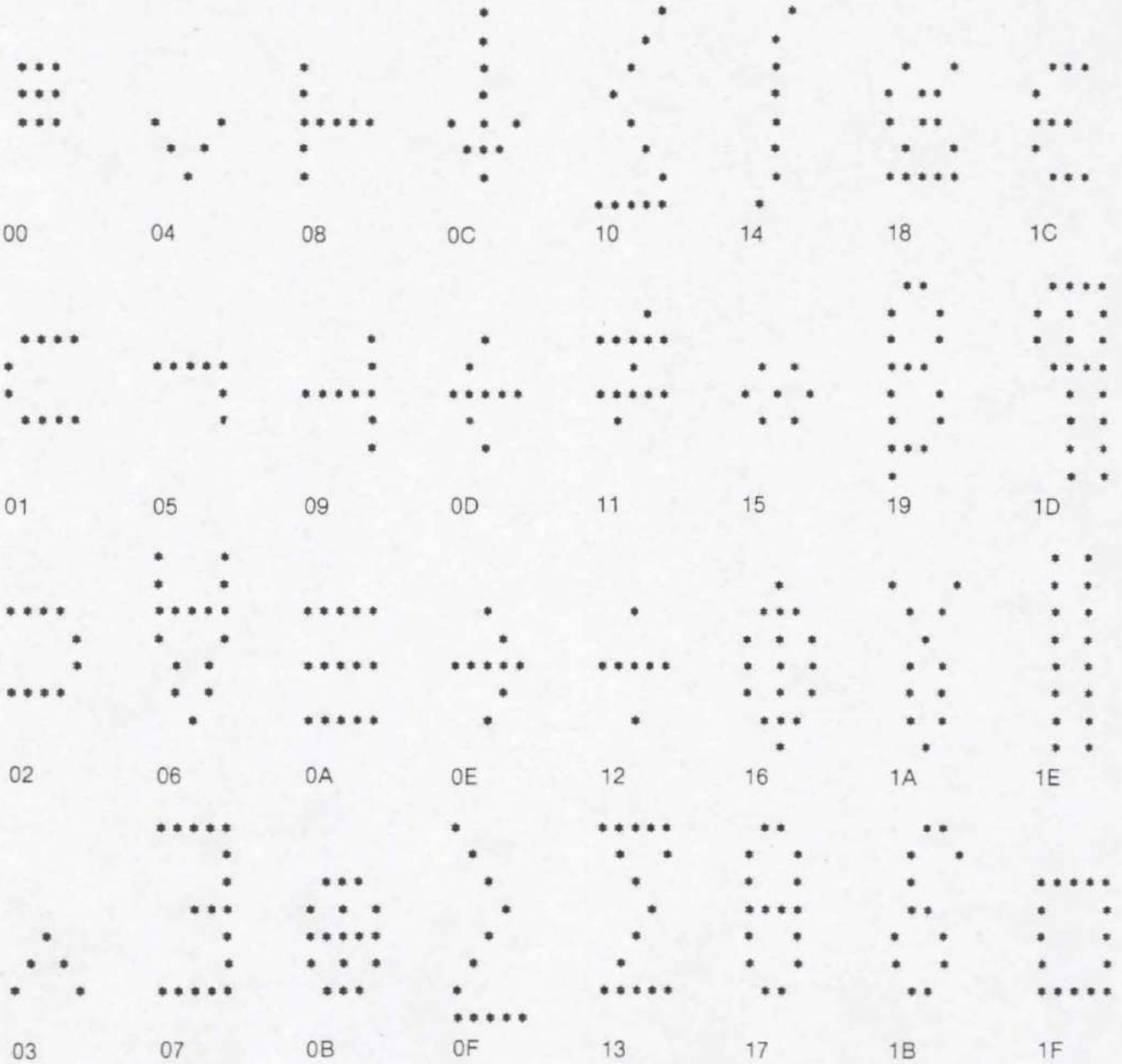
FLOOD

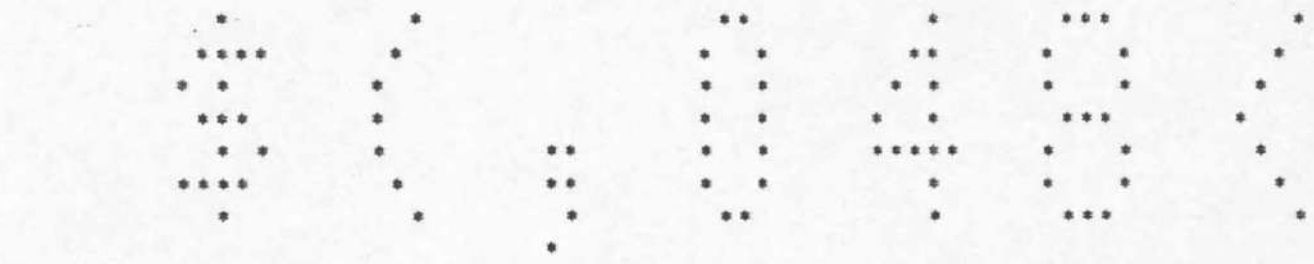
CC	204	314	C: ⟨ xh ⟩ ⟨ xl ⟩ ⟨ yh ⟩ ⟨ yl ⟩ R: none	Flood with Zeroes
CD	205	315	C: ⟨ xh ⟩ ⟨ xl ⟩ ⟨ yh ⟩ ⟨ yl ⟩ R: none	Flood with Ones
CE	206	316	C: ⟨ dxh ⟩ ⟨ dxl ⟩ ⟨ dyh ⟩ ⟨ dyl ⟩ R: none	Flood Relative with Zeroes
CF	207	317	C: ⟨ dxh ⟩ ⟨ dxl ⟩ ⟨ dyh ⟩ ⟨ dyl ⟩ R: none	Flood Relative with Ones

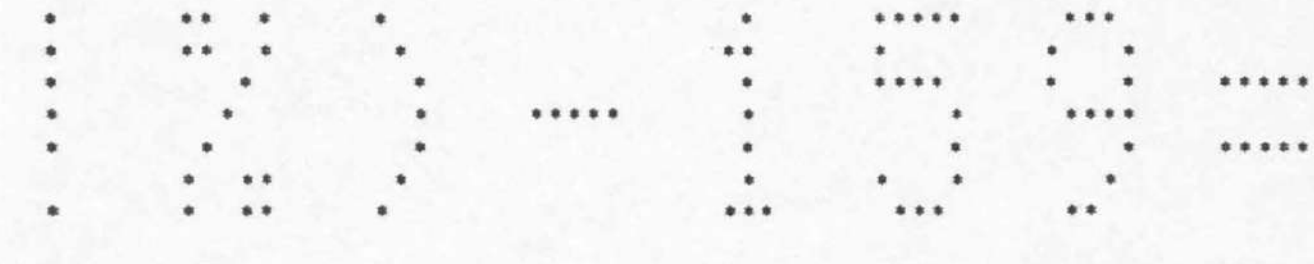
MACRO

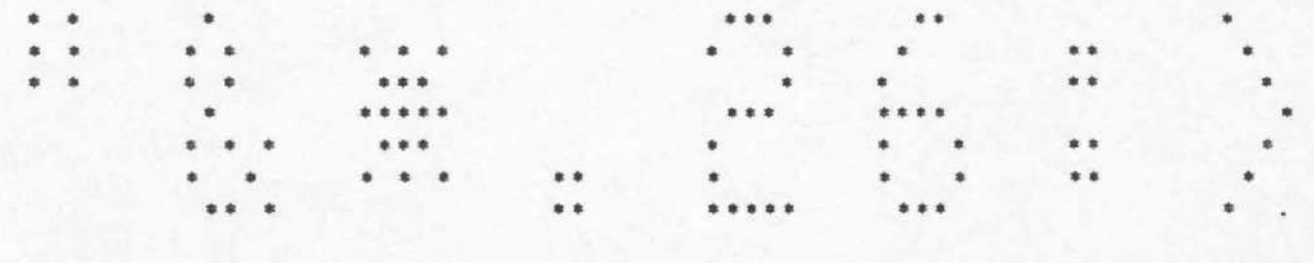
D0	208	320	C: ⟨ n ⟩ or FF R: ⟨ code ⟩	Start/Stop Macro Definition
D1	209	321	C: ⟨ byte ⟩ R: ⟨ code ⟩	Add Next Macro Byte
D2	210	322	C: ⟨ n ⟩ or FF ⟨ sh ⟩ ⟨ sl ⟩ R: ⟨ code ⟩	Erase Macro or Clear Facility
D3	211	323	C: ⟨ n ⟩	Invoke Macro

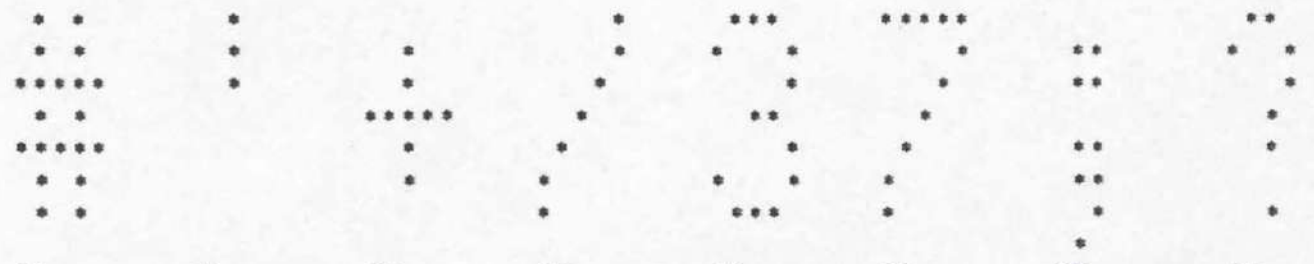
Appendix 2
The Standard Character Font

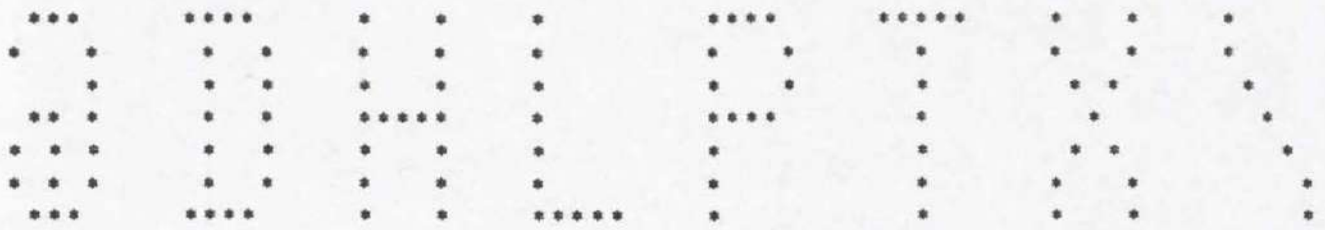



20
24
28
2C
30
34
38
3C

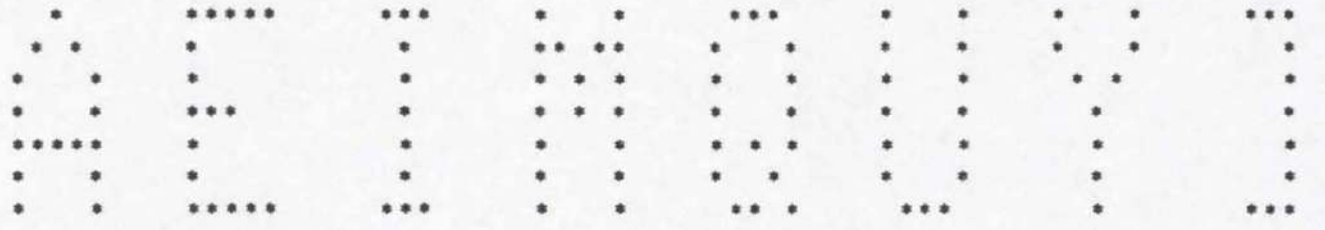

21
25
29
2D
31
35
39
3D


22
26
2A
2E
32
36
3A
3E

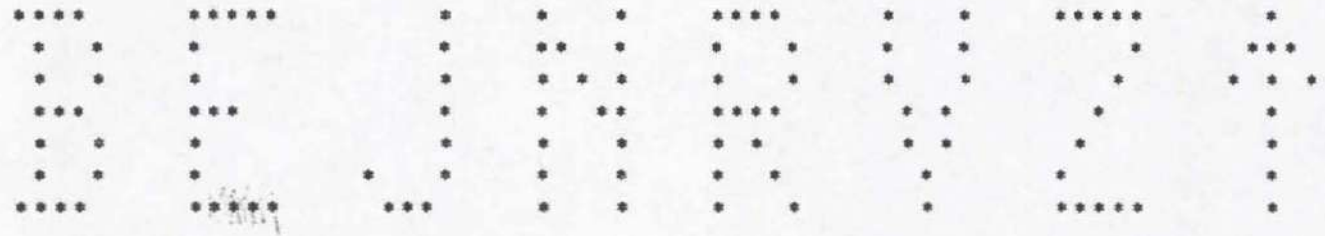

23
27
2B
2F
33
37
3B
3F



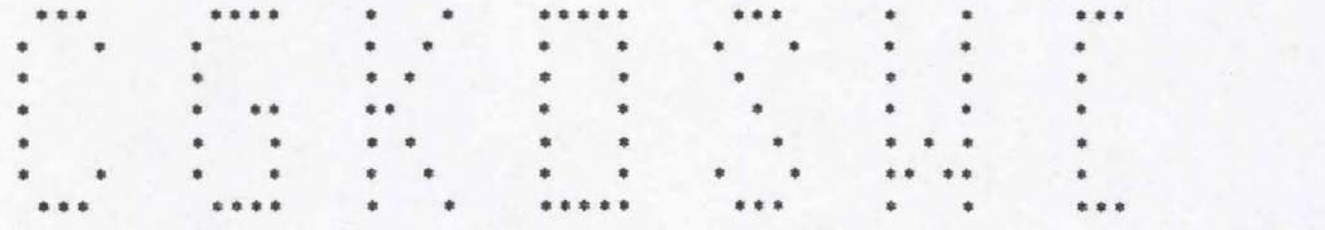
40 44 48 4C 50 54 58 5C



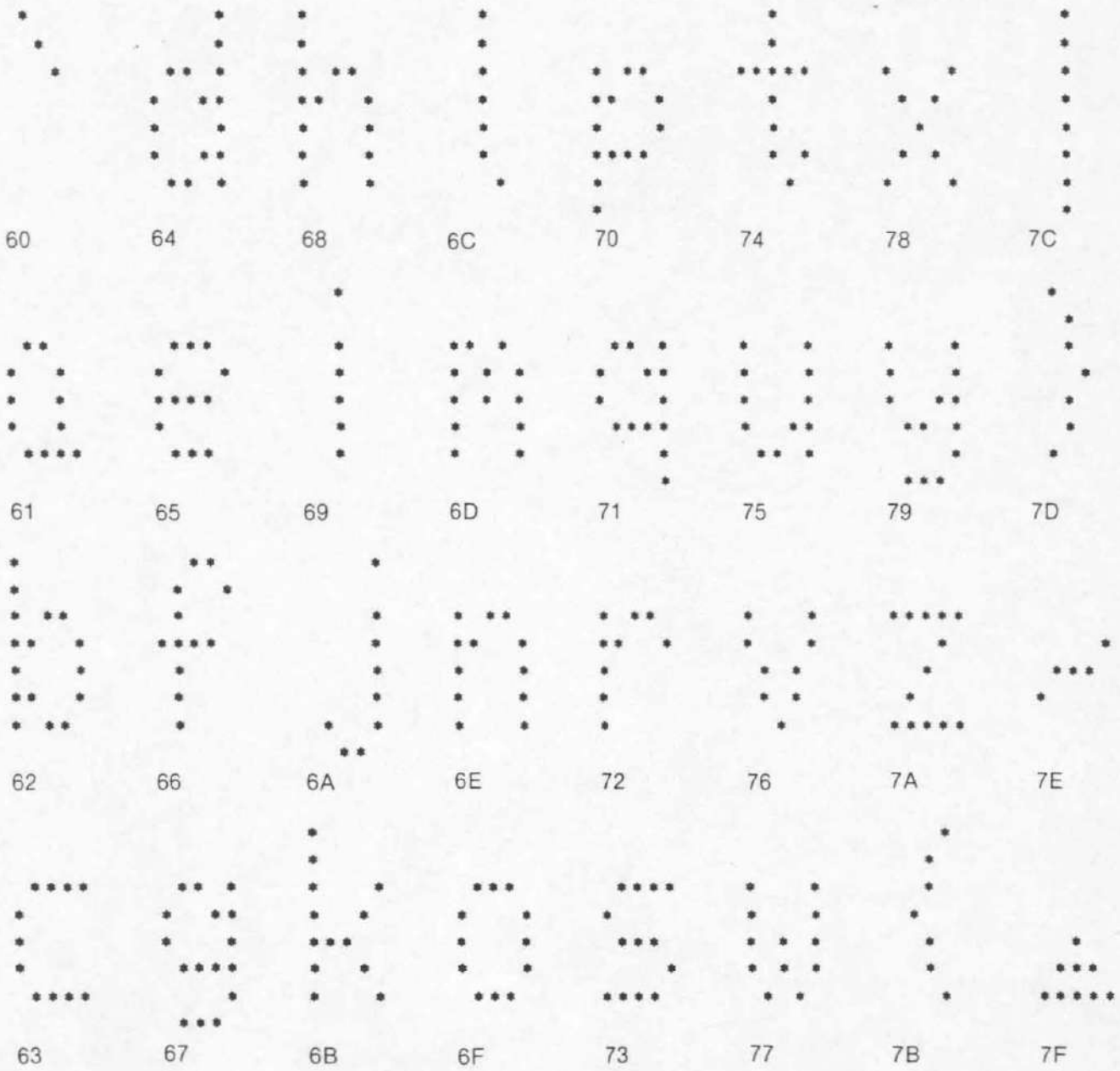
41 45 49 4D 51 55 59 5D



42 46 4A 4E 52 56 5A 5E



43 47 4B 4F 53 57 5B 5F


60
64
68
6C
70
74
78
7C
61
65
69
6D
71
75
79
7D
62
66
6A
6E
72
76
7A
7E
63
67
6B
6F
73
77
7B
7F

Appendix 3
Screenware™ Pak I Internal Entry Points

SYSTEM 0000H
entry: none
exit: none
destroys: NA
description: call here to restart the system as it would be at cold start

READBUF 012FH
entry: none
exit: carry flag set if a byte is available, cleared otherwise; if carry set, [A] = byte from host
destroys: A, D, E, H, L
description: call here to read a byte from the host (via the interrupt buffered interface) if a byte is available

GETBYTE 01A4H
entry: none
exit: [A] = byte from host
destroys: none
description: call here to read a byte from the host; GETBYTE waits until a byte is available

GETCOORD 005BH
entry: none
exit: [HL] = coordinate from host (sent high byte first)
destroys: H, L
description: call here to read a 9-bit coordinate from the host; the high 7 bits of H are zeroed

GETYCORD 0197H
entry: none
exit: [HL] = 9-bit coordinate clipped to 479
destroys: A
description: call here to read a 9-bit coordinate from the host; the coordinate is clipped to 479 if it is larger

GETADDR 038FH
entry: none
exit: [HL] = 16-bit address from host (sent high byte first)
destroys: A
description: call here to get a 16-bit quantity from the host

SENDBYTE 0259H
entry: [A] = byte
exit: none
destroys: B
description: call here to send a byte to the host; SENDBYTE waits until the outbound buffer is clear before sending

SENDCOORD 0254H
entry: [HL] = 16-bit value to send to host
exit: none
destroys: B
description: call here to send 16 bits (high order byte first) to the host

DPYLOC 021BH

entry: [DE] = X coordinate [0-511]
[HL] = Y coordinate [0-479]

exit: [A] = bit mask
[B] = bit mask
[C] = bit number [0 leftmost, 7 rightmost]
[HL] = display buffer address

destroys: D, E

description: call here to convert coordinates into a Z80 memory address [on the visible screen] and bit mask; the bit mask [containing one ON bit] identifies the pixel within the address byte; the bit number is the position of the ON bit within the byte

SCREENC 01AFH

entry: [A] = mode [0, 1, 2,]

exit: none

destroys: all

description: call here for SCREEN command, as described in the manual; do not call with mode = 3, since this mode will try to read a byte from the host

POINT 01F7H

entry: [B] = mode [0, 1, 2]
[DE] = X coordinate
[HL] = Y coordinate
[CX] = X coordinate
[CY] = Y coordinate

exit: none

destroys: all

description: call here for POINT command, as described in the manual; do not call with mode = 3, since the code will then send a response to the host

VECTOR 0547H

entry: [B] = mode
[DE] = x coordinate
[NEWCX] = x coordinate
[HL] = y coordinate
[NEWCY] = y coordinate

exit: none

destroys: all

description: call here for the VECTOR commands, as described in the manual

REGION 0275H

entry: enter via the following code sequence:

```
    LXI H,RETURN
    PUSH H
    MVI B, < mode >
    PUSH B
    LXI H, < Y1 >
    PUSH H
    LXI H, < X2 >
    PUSH H
    LXI D, < X1 >
    LXI H, < Y2 >
    JMP REGION
RETURN: ...
```

exit: none

destroys: all

description: call via the given sequence for the REGION commands, as described in the manual

CHAR 03E7H

entry: [A] = character or character mode bits
[B] = command mode [0, 1, 2]

exit: none

destroys: all

description: call here for the CHARACTER commands, as described in the manual; the command mode bits select plot character, set character mode, and define alternate characters; character mode bits are as described in the manual

DRAWCROSS 068EH

entry: [TX] = X coordinate
[TY] = Y coordinate

exit: none

destroys: all

description: call here to complement the bits on the tracking crross at [TX, TY], [i.e., if the cross is on at [TX, TY], turn it off, and vice versa]

DRAWHAIRS 07A2H

entry: [HX] = X coordinate
[HY] = Y coordinate

exit: none

destroys: all

description: call here to complement the bits on the crosshairs at [HX, HY] [i.e., if the crosshairs are on, turn them off, and vice versa]

ALPHINIT 07D9H

entry: none

exit: none

destroys: all

description: call here to reset the alpha interface: [clears the screen and sets AX, AY to top left of screen]

TTYCHAR 07EBH

entry: [A] = ASCII code

exit: none

destroys: all

description: call here to print an ASCII character at AX, AY [TTYCHAR does not advance AX, AY]

TTY 08EFH

entry: [A] = ASCII code

exit: none

destroys: all

description: call here to send an ASCII code to the ALPHA processor; control codes are recognized as described in the manual, and AX, AY are advanced, possibly invoking the scrolling mechanism

Variables and Parameters:

VARIABLE	ADDR	#BYTES	DESCRIPTION
CX	FFFB	2	the current graphics X coordinate
CY	FFF9	2	the current graphics Y coordinate
NEWCX	FFE9	2	[see the VECTOR entry point]
NEWCY	FFE7	2	[see the VECTOR entry point]
AX	FFD0	2	the current ALPHA screen X coordinate
AY	FFCE	2	the current ALPHA screen Y coordinate
AR	FFCD	1	the current ALPHA row number
AC	FFCC	1	the current ALPHA column number
ALPHSCRL	FFCB	1	the current ALPHA scroll parameter
ALPHMODE	FFCA	1	the current ALPHA mode bits
CHARMODE	FFC9	1	the current GRAPHICS character mode bits
TX	FFD4	2	the current tracking cross X coordinate
TY	FFD2	2	the current tracking cross Y coordinate
TSTAT	FFD6	1	1 if the tracking cross is visible, 0 otherwise [The DRAWCROSS entry point does not maintain this cell - you should do it manually when calling DRAWCROSS]
LPX	FFFE	1	the [X coordinate/2] of the last light pen interrupt
LPY	FFFF	1	the [Y coordinate/2] of the last light pen interrupt
LPSTAT	FFFD	1	0 if no light pen interrupt has occurred, 1 otherwise [you should reset to 0 to acknowledge a light pen interrupt]
HX	FFBF	2	the current X coordinate of the crosshair
HY	FFC1	2	the current Y coordinate of the crosshair
HSTAT	FFBE	1	1 if the crosshairs are visible, 0 otherwise [the DRAWHAIRS entry point does not maintain this cell - you should do it manually when calling DRAWHAIRS
ROMCHAR	09FA	-	the beginning of the ROM character generator

Appendix 4
Screenware™ Pak II Internal Entry Points

SYSTEM 0099H
entry: none
exit: none
destroys: NA
description: call here to restart the system as it would be at cold start

READBUF 01CCH
entry: none
exit: carry flag set if a byte is available, cleared otherwise; if carry set, [A] = byte from host
destroys: A, D, E, H, L
description: call here to read a byte from the host [via the interrupt buffered interface] if a byte is available

GETBYTE 0059H
entry: none
exit: [A] = byte from host
destroys: none
description: call here to read a byte from the host; GETBYTE waits until a byte is available

GETXCORD 0113H
entry: none
exit: [HL] = coordinate from host [sent high byte first]
destroys: H, L
description: call here to read a 9-bit coordinate from the host; the high 7 bits of H are zeroed

GETYCORD 003BH
entry: none
exit: [HL] = 9-bit coordinate clipped to 479
destroys: A
description: call here to read a 9-bit coordinate from the host; the coordinate is clipped to 479 if it is larger

GETADDR 0069H
entry: none
exit: [HL] = 16-bit address from host [sent high byte first]
destroys: A
description: call here to get a 16-bit quantity from the host

SENDBYTE 035DH
entry: [A] = byte
exit: none
destroys: B
description: call here to send a byte to the host; SENDBYTE waits until the outbound buffer is clear before sending

SENDCOORD 0358H
entry: [HL] = 16-bit value to send to host
exit: none
destroys: B
description: call here to send 16 bits [high order byte first] to the host

DPYLOC 031FH

entry: (DE) = X coordinate (0-511)
(HL) = Y coordinate (0-479)

exit: (A) = bit mask
(B) = bit mask
(C) = bit number (0 leftmost, 7 rightmost)
(HL) = display buffer address

destroys: D, E

description: call here to convert coordinates into a Z80 memory address (on the visible screen) and bit mask; the bit mask (containing one ON bit) identifies the pixel within the address byte; the bit number is the position of the ON bit within the byte

SCREENC 0293H

entry: (A) = mode (0, 1, 2,)

exit: none

destroys: all

description: call here for SCREEN command, as described in the manual; do not call with mode = 3, since this mode will try to read a byte from the host

POINT 02FFH

entry: (B) = mode (0, 1, 2)
(DE) = X coordinate
(HL) = Y coordinate
(CX) = X coordinate
(CY) = Y coordinate

exit: none

destroys: all

description: call here for POINT command, as described in the manual; do not call with mode = 3, since the code will then send a response to the host

VECTOR 0767H

entry: (B) = mode
(DE) = x coordinate
(NEWCX) = x coordinate
(HL) = y coordinate
(NEWCY) = y coordinate

exit: none

destroys: all

description: call here for the VECTOR commands, as described in the manual

REGION 049CH

entry: enter via the following code sequence:

```
    LXI H,RETURN
    PUSH H
    MVI B, < mode >
    PUSH B
    LXI H, < Y1 >
    PUSH H
    LXI H, < X2 >
    PUSH H
    LXI D, < X1 >
    LXI H, < Y2 >
    JMP REGION
RETURN: ...
```

exit: none

destroys: all

description: call via the given sequence for the REGION commands, as described in the manual

RPOINT 02D2H

entry: [B] = mode [0, 1, 2]
[DE] = X coordinate
[HL] = Y coordinate
[CX] = X coordinate
[CY] = Y coordinate
enter via the following code sequence:
CALL 0253H
CALL 02FFH
CALL 024FH
exit: none
destroys: all
description: call here for RPOINT command, as described in the manual; do not call with mode = 3, since the code will then send a response to the host

RVECTOR 07A8H

entry: [B] = mode
[DE] = x coordinate
[NEWCX] = x coordinate
[HL] = y coordinate
[NEWCY] = y coordinate
enter via the following code sequence:
CALL 0253H
CALL 0767H
CALL 024FH
exit: none
destroys: all
description: call here for the RVECTOR commands, as described in manual

RREGION 0492H

entry: enter via the following code sequence:

```
CALL 0253H
LXI H,RETURN
PUSH H
MVI B,[mode]
PUSH B
LXI H,[Y1]
PUSH H
LXI H,[X2]
PUSH H
LXI D,[X1]
LXI H,[Y2]
JMP  049CH
RETURN: ...
CALL 024FH
```

exit: none
destroys: all
description: call via the given sequence for the RREGION command, as described in the manual

CHAR 0609H

entry: [A] = character or character mode bits
[B] = command mode [0, 1, 2]
exit: none
destroys: all
description: call here for the CHARACTER commands, as described in the manual; the command mode bits select plot character, set character mode, and define alternate characters; character mode bits are as described in the manual

DRAWCROSS 0E28H

entry: [TX] = X coordinate
[TY] = Y coordinate

exit: none

destroys: all

description: call here to complement the bits on the tracking cross at [TX, TY], [i.e., if the cross is on at [TX, TY], turn it off, and vice versa]

DRAWHAIRS 11B4H

entry: [HX] = X coordinate
[HY] = Y coordinate

exit: none

destroys: all

description: call here to complement the bits on the crosshairs at [HX, HY] [i.e., if the crosshairs are on, turn them off, and vice versa]

ALPHINIT 08CBH

entry: none

exit: none

destroys: all

description: call here to reset the alpha interface: [clears the screen and sets AX, AY to top left of the alpha area]

TTYCHAR 091CH

entry: [A] = ASCII code

exit: none

destroys: all

description: call here to print an ASCII character at AX, AY [TTYCHAR does not advance AX, AY]

TTY 0A69H

entry: [A] = ASCII code

exit: none

destroys: all

description: call here to send an ASCII code to the ALPHA processor; control codes are recognized as described in the manual, and AX, AY are advanced, possibly invoking the scrolling mechanism

SPLO 022EH NOT SUPPORTED

entry: [A] = number of ALPHA lines

exit: none

destroys: all

description: call here to split the screen into a graphic area and an alpha area

WTI 05C7H

entry: none

exit: none

destroys: all

description: call here to force a cold start to the software and send a byte <AB> to the host

FLD 0C5AH

entry: [A] = mode [0, 1]
[DE] = X coordinate
[HL] = Y coordinate
enter via the following code sequence:

```
PUSH    PSW
JMP     FLD
```

exit: none

destroys: all

description: call here to flood the bordered region around point X,Y

CIR 0F00H

entry: [A] = radius
[B] = mode [0, 1, 2]

exit: none

destroys: all

description: call here to draw a circle at current graphic cursor

INVOKE 1065H

entry: [A] = macro #

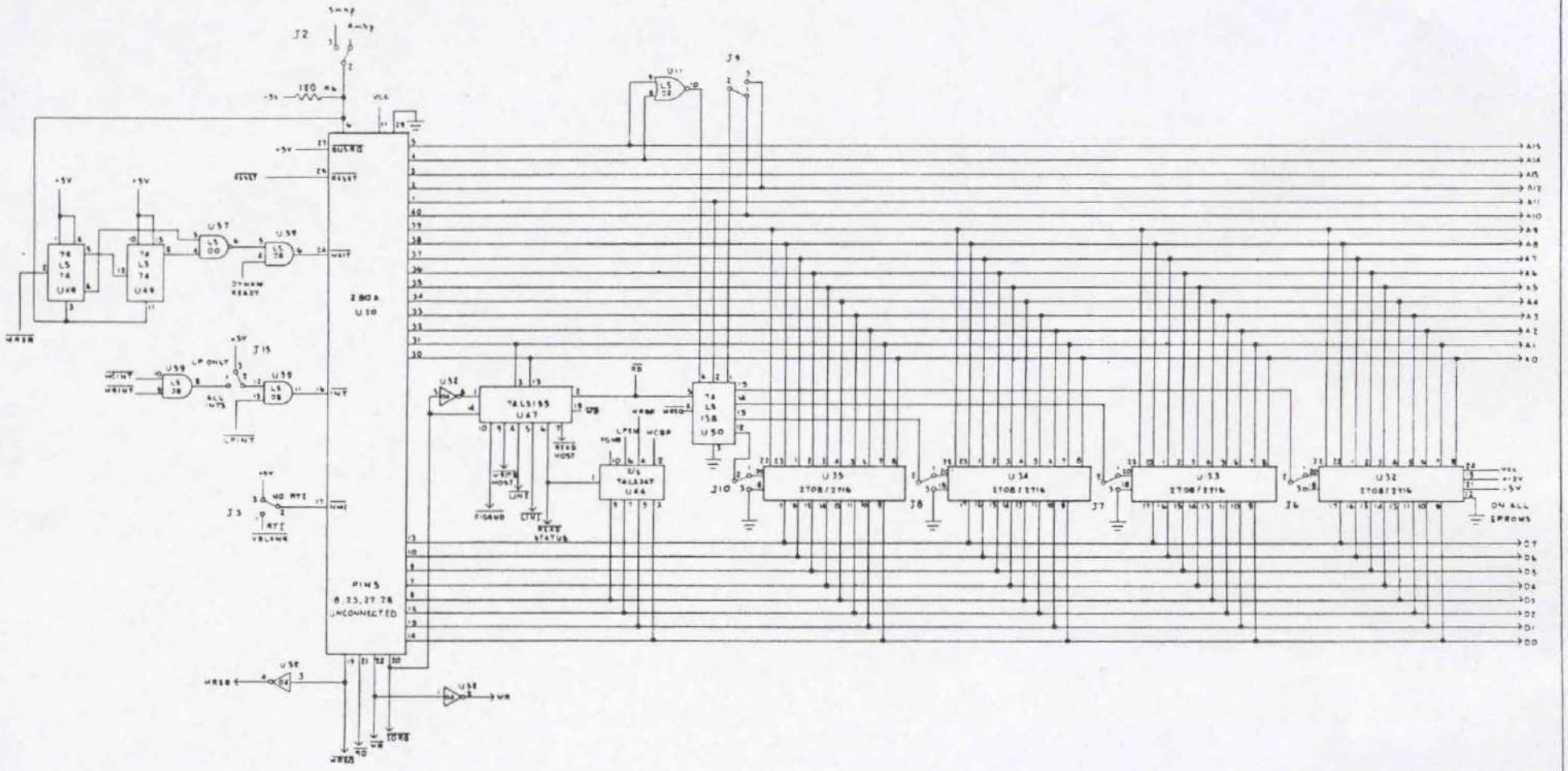
exit: none

destroys: all

description: call here to invoke the desired previously created macro

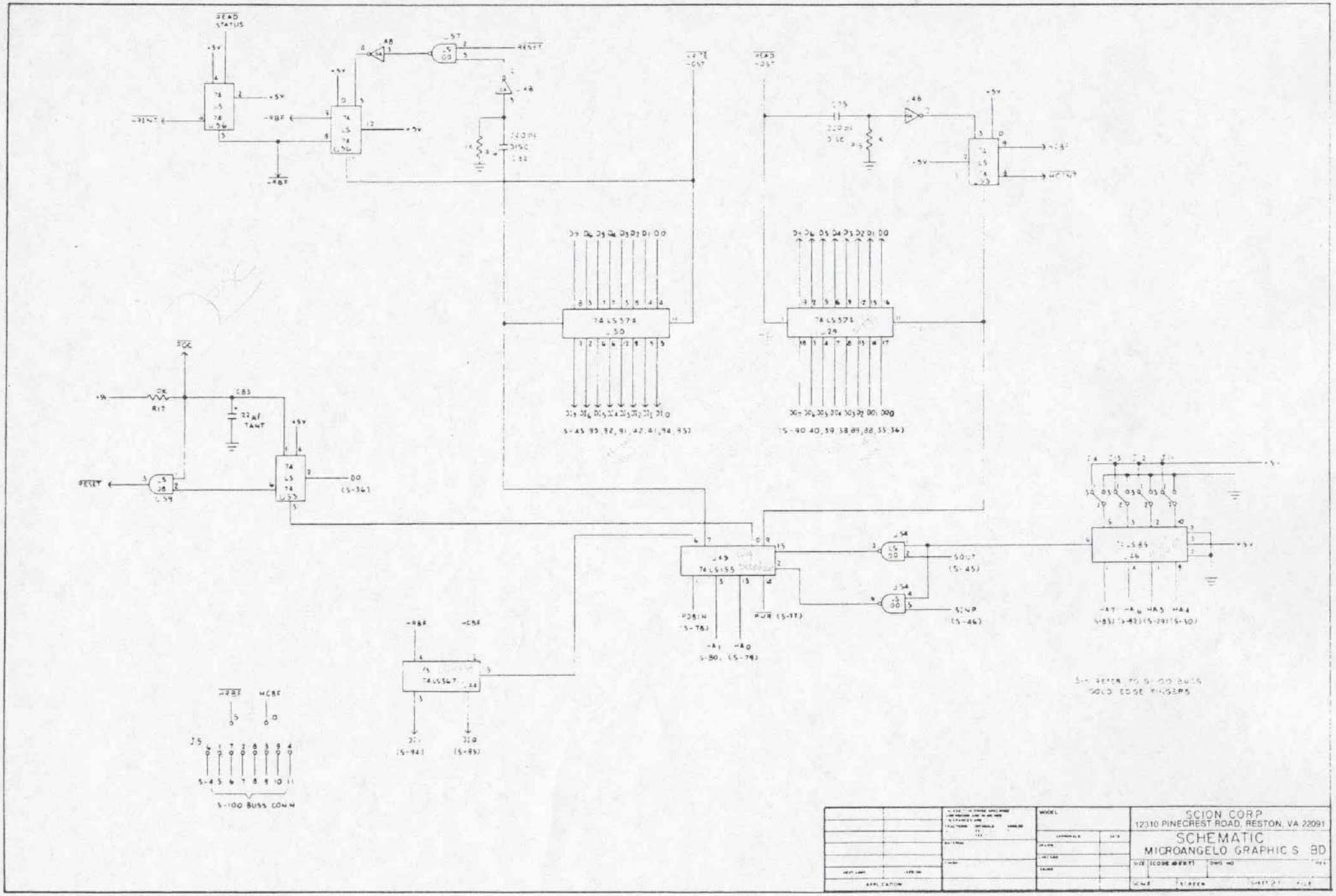
Variables and Parameters:

VARIABLE	ADDR	#BYTES	DESCRIPTION
CX	FFFB	2	the current graphics X coordinate
CY	FFF9	2	the current graphics Y coordinate
NEWCX	FFE9	2	[see the VECTOR entry point]
NEWCY	FFE7	2	[see the VECTOR entry point]
AX	FFD0	2	the current ALPHA screen X coordinate
AY	FFCE	2	the current ALPHA screen Y coordinate
AR	FFCD	1	the current ALPHA row number
AC	FFCC	1	the current ALPHA column number
ALPHSCRL	FFCB	1	the current ALPHA scroll parameter
ALPHMODE	FFCA	1	the current ALPHA mode bits
CHARMODE	FFC9	1	the current GRAPHICS character mode bits
TX	FFD4	2	the current tracking cross X coordinate
TY	FFD2	2	the current tracking cross Y coordinate
TSTAT	FFD6	1	1 if the tracking cross is visible, 0 otherwise [The DRAWCROSS entry point does not maintain this cell - you should do it manually when calling DRAWCROSS]
LPX	FFFE	1	the [X coordinate/2] of the last light pen interrupt
LPY	FFFF	1	the [Y coordinate/2] of the last light pen interrupt
LPSTAT	FFFD	1	0 if no light pen interrupt has occurred, 1 otherwise [you should reset to 0 to aknowledge a light pen interrupt]
HX	FFBF	2	the current X coordinate of the crosshair
HY	FFC1	2	the current Y coordinate of the crosshair
HSTAT	FFBE	1	1 if the crosshairs are visible, 0 otherwise [the DRAWHAIRS entry point does not maintain this cell - you should do it manually when calling DRAWHAIRS
ROMCHAR	11DB	-	the beginning of the ROM character generator



SCION CORP.
12310 PINECREST ROAD, RESTON, VA 22091

SCHEMATIC
MICROANGELO GRAPHICS BD.

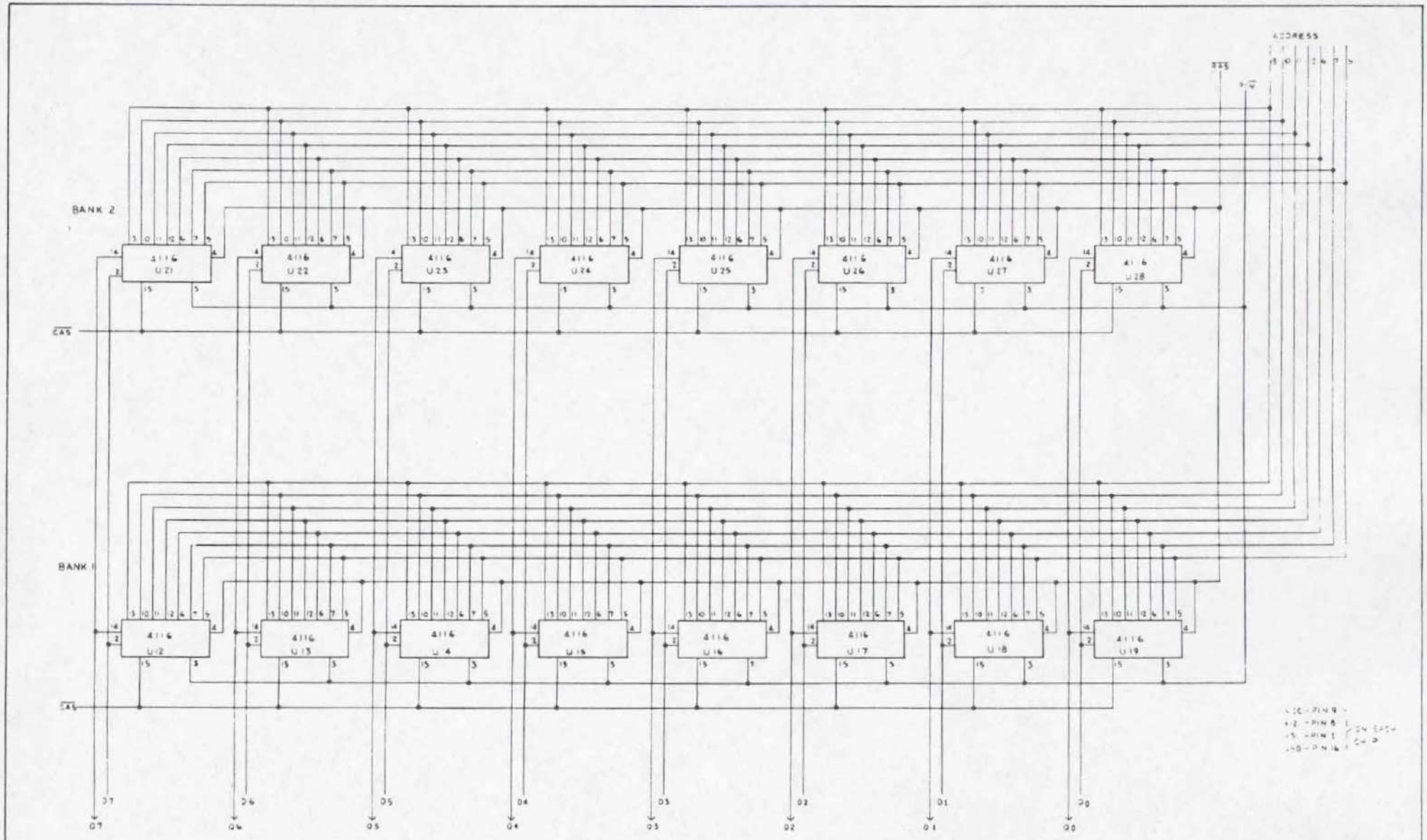

READ STATUS
RESET
74LS374
74LS373
74LS155
74LS367
74LS85
S-100 BUSS CONN
SCION CORP
12310 PINECREST ROAD, RESTON, VA 22091
SCHEMATIC
MICROANGELO GRAPHICS BD

SCION CORP
12310 PINECREST ROAD, RESTON, VA 22091
SCHEMATIC
MICROANGELO GRAPHICS BD

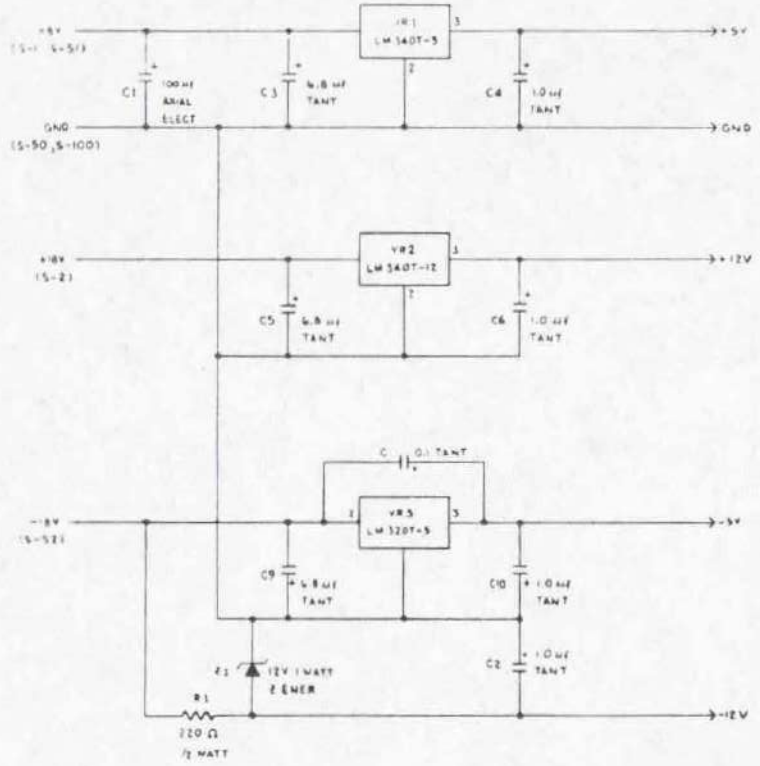
SCION CORP.
12310 PINECREST ROAD, RESTON, VA 22091

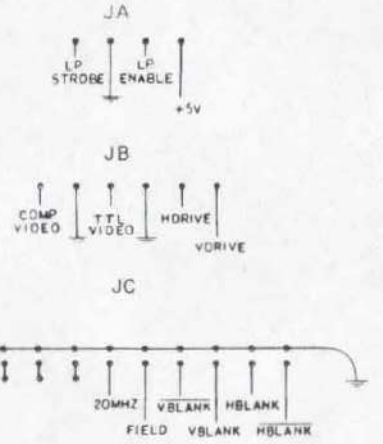
SCHEMATIC

MICROANGELO GRAPHIC'S BD



SCION CORP.
12310 PINECREST ROAD, RESTON, VA 22091
SCHEMATIC
MICROANGELO GRAPHICS BD.





				SCION CORP 12310 PINECREST ROAD, RESTON, VA 22091
				SCHEMATIC MICROANGELO GRAPHIC'S BD
				SIZE / DRAWING NUMBER / REV
				SCALE / SCREEN / SHEET 6 OF 6 / FILE