I R O N I C S

IV-1653

VMEbus CRT

Controller Board

User Guide

IRONICS
Incorporated

IV-1653

VMEbus CRT

Controller Board

IV-1653-MAN-1.2
5/10/85

This manual has been carefully checked for accuracy. We can, however, assume no responsibility for errors, nor can we assume any liability which arises from the use or application of this board.

As part of our continuing effort to improve the quality of the Ironics VMEbus productivity series, we solicit comments, criticism and suggestions of our customers. We would greatly appreciate your taking the time to provide us with any feedback. Your comments, positive or negative, about the manual, hardware or software are especially welcome. Send your remarks to:

Ironics Incorporated
Quality Assurance
798 Cascadilla Street
Ithaca, New York 14850

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. GENERAL INFORMATION

1.1 INTRODUCTION

The Ironics 'IV' application series utilizes state-of-the-art circuitry to provide the OEM systems designer with powerful, low-cost microcomputer system modules. The Ironics 'IV' productivity series is designed to take full advantage of the VMEbus concept. The use of high-density MOS-LSI makes it possible to reduce module size and provide a high degree of functionality.

1.2 GENERAL DESCRIPTION

THE IV-1653 is a character-oriented display controller capable of both color and monochrome operation. Its downloadable character set allows the IV-1653 to satisfy many graphics display requirements without the support overhead or cost of a fully bit-mapped display. All memory on the IV-1653 is dual ported between the VMEbus and the display controller. This allows rapid, transparent screen updates. Advanced features such as horizontal scroll, softscroll, and split screen may be implemented with a minimum of software overhead. Both parallel and serial keyboard interfaces are provided on the IV-1653 to complete the console interface in a workstation environment. A second full duplex asynchronous printer port is also provided. The IV-1653 may also be synchronized with the IV-1651 high resolution graphics board to provide alphanumeric overlay on graphics displays.

1.3 FEATURES OF THE IV-1653 DISPLAY CONTROLLER BOARD

- Character-oriented CRT Display Controller
- Supports up to 48 lines of 80 characters noninterlaced; up to 66 lines of 102 characters with interlaced scan
- Dot addressable mosaic graphics using a downloadable character set
- 16Kx16 display memory dual ported with the bus
- 256 RAM (downloadable) or EPROM characters, 6x7 to 9x16 pixels per character
- Transparent screen memory updating
- 8 foreground colors and 8 background colors at each location
- Blink and underline attributes on character basis; double wide attribute on row basis in color mode
- Provides alphanumeric overlay for the IV-1651 at up to 640 x 480 resolution
- Softscroll, splitscreen, window capability
- Parallel keyboard port provided
- Two full duplex, asynchronous serial ports provided
- VMEbus interrupter provided for keyboard, serial I/O, and CRT interfaces

2. FUNCTIONAL DESCRIPTION

2.1 INTRODUCTION

A block diagram of the IV-1653 CRT display controller is shown in Figure 1. The board consists of a bus interface, a display refresh controller, a 16Kx16 memory which contains the display image in terms of character codes, a 4Kx9 character generator memory which expands character codes into parallel video dot rows, a 68230 PI/T used as a control port, a keyboard input port and independent counter/timer, and a 2681 dual UART used to provide access to serial keyboards and other ancilliary devices.

The bus interface includes an address decoder, an interrupter, and two 8-bit data buffers. The address decoder recognizes a contiguous block of addresses corresponding in turn to the 16Kx16 image memory, the 4Kx9 character generator memory, the SC2674 CRT controller, the MC68230L8 PI/T and the SC2681 DUART. Standard address assignments are shown in Figure 2. To change them,either or both of two FPLA's must be reprogrammed. The interrupter interfaces the two interrupt request outputs from the PI/T to the VMEbus. The board may interrupt on any one interrupt level using the jumper options provided. The PI/T provides a timer interrupt, a keyboard interrupt, and a vector for the CRT controller's and DUART's interrupt outputs. All of the on-board I/O utilizes the lower data bus.

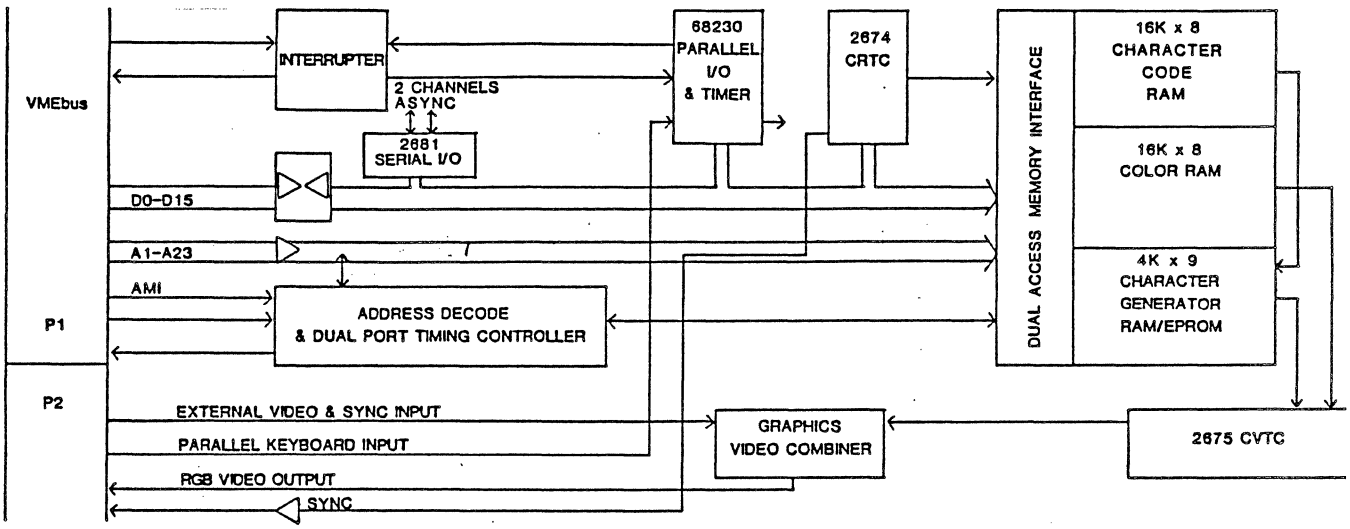
Both the image memory and the character generator memory are dual ported with the VMEbus. Accesses to them are synchronized to the CRT display refresh clock and controlled so that the current bus master may access the RAM at any time with no visible disturbance to the display. Two accesses to the RAM are made each character period. This reduces the latency for transparent access to less than two character clocks - or about 1 microsecond. With this timing, a 2000-character screen may be completely redrawn in less than 3 ms.

The character generator memory is implemented with 6116 2Kx8 static RAMs plus a 2147 4Kx1 static RAM. 120 ns access time RAMs are provided allowing dual access operation at the maximum display resolution. Bipolar PROMs or 250 ns access time 2716 EPROMs may be substituted if a hard character set is desired.

The display control circuitry is built around the Signetics 2674 and 2675 chip set. The 2674 provides the vertical and horizontal timing signals as well as the display refresh address, while the 2675 functions as a parallel to serial converter and color or monochrome attributes controller. Display format is fully programmable via the 2674's registers, control port outputs, and

strapping options on the board (provided the correct dot clock oscillator is employed). Several advanced features may also be implemented by suitably programming the 2674. These include automatic split screen, automatic soft scroll and double high and/or double wide characters.

Figure 2-1. Block Diagram of the IV-1653



2.2 VIDEO TIMING

The Signetics 2674 Advanced Video Timing Controller (AVDC) provides all timing and coordination for the video display. It generates the sync pulses, controls the cursor and supplies the addresses to the display RAMs for the character generator and the video controller.

The 2674 AVDC contains 15 Initialization Registers which define display parameters affecting screen format and monitor timing. A register pointer, cleared by the Reset command, is incremented after each Initialization Register reference. It may also be written by a command to the AVDC. These registers require initialization by the CPU for: characters per line, lines per screen, cursor style, blink rates, double-height character rows, split screen control and address pointers into the the display RAMs. The AVDC also responds to commands for: master reset, internal register addressing, display on/off, cursor on/off, and interrupt enable/disable. The read and write display memory operations mentioned in the 2674's data sheet are not effective on the IV-1653 because direct access to the display memory from the VMEbus has been provided.

An interrupt and status register within the AVDC indicates busy, vertical blanking interval, line zero of character row, split screen1,ready and split2. Any or all of these signals may be used to generate an interrupt. The interrupt vector is supplied by the PI/T chip. This is affected by connection of an active low interrupt request from the AVDC to the H1 pin of the 68230 PI/T and programming the PI/T to generate an interrupt when it is low. The interrupting condition is then determined by reading the status register of the AVDC. Other possible causes of an interrupt from the PI/T are from the keyboard port and from the serial I/O interrupt connected to the H2 pin.

For complete information on programming the AVDC, consult the Signetics 2674 Programmable Video Timing Controller data sheet and applications notes supplied with this document.

TABLE 2-1. I/O Port Addresses for the 2674 AVDC

HEX ADDR		READ	WRITE
FCA040+	1	interrupt register	initialization register
	3	status register	command register
	5	screen start lower address register	screen start lower address register
	7	screen start upper address register	screen start upper address register
	9	cursor address lower register	cursor address lower register
	11	cursor address upper register	cursor address upper register
	13	screen start 2 lower register	screen start 2 lower register
	15	screen start 2 upper register	screen start 2 upper register

2.3 DISPLAY RAM

The display RAM consists of 16K 16-bit words of dynamic RAM. The devices used have an access time of 100 ns. and a cycle time of 160 ns. This allows two accesses per character clock period at character clock rates up to 3.1 MHz and supports pixel clock rates above the 25 MHz limit of the 2675. The display RAM is dual ported with the VMEbus. It may be accessed at any time without visible disturbance of the display. The maximum latency for bus accesses is two character clock periods plus one dot clock period plus 100 ns. The minimum latency is .5 character clock plus 1 dot clock plus 100 ns. The average latency is approximately 1.2 character clocks. Both word and byte accesses may be performed. The VMEbus address modifier field is checked and only data accesses are permitted.

The lower byte of the display memory word is the character code and is used as an index into the character generator RAM. The upper byte of the display memory word contains the color codes and attribute bits. The definition of this byte is shown in the table below.

TABLE 2-2. Attribute Byte Definition

	Monochrome Mode:	Color Mode:
bit 8	General purpose output	Background blue
bit 9	Reverse video attribute	Background green
bit 10	General purpose output	Background red
bit 11	Blank attribute	Foreground blue
bit 12	Background intensify	Foreground green
bit 13	Highlight attribute	Foreground red
bit 14	Underline attribute	Underline attibute
bit 15	Blink attribute	Blink attribute

Note: 1. The attribute byte is active low.
2. In monochrome mode, the general purpose outputs mentioned abov are available on what would be the blue and red video outputs in the color mode.

COLOR CODES

B13 B12 B11	FOREGROUND COLOR
B10 B9 B8	BACKGROUND COLOR
0 0 0	WHITE
0 0 1	YELLOW
0 1 0	MAGENTA
0 1 1	RED
1 0 0	CYAN
1 0 1	GREEN
1 1 0	BLUE
1 1 1	BLACK

TABLE 2-3. Color Code Assignments

COLOR CODE EXAMPLES

FOREGND COLOR	WHITE BKGD	RED BKGD	GREEN BKGD	BLUE BKGD	BLACK BKGD
WHITE	C0	C3	C5	C6	C7
RED	D8	DB	DD	DE	DF
GREEN	E8	EB	ED	EE	EF
BLUE	F0	F3	F5	F6	F7
YELLOW	C8	CB	CD	CE	CF
CYAN	E0	E3	E5	E6	E7
MAGENTA	D0	D3	D5	D6	D7
BLACK	F8	FB	FD	FE	FF

TABLE 2-4. Examples of Combined Foreground and Background Color Codes

Notes:
1. "AND" the above hex codes with 07Fh to enable blinking.
2. "AND" the above codes with 0DFh to enable underlining.
3. Although not shown, yellow, cyan, and magenta backgrounds may also be programmed.

2.4 CHARACTER GENERATOR MEMORY

The character generator memory is implemented with fast static RAMs and dual ported with the bus to allow transparent access at any time without visible display disturbance. The timing and latency for character generator access is the same as for display memory access. The character generator memory is 4K words deep and 9 bits wide on bits 0-8 of the data lines. This is sufficient for 256 characters of up to 16 scan lines per character. The memory for each character is addressed in the order in which it is scanned for display refresh. The row scan count addresses occupy the four least significant address lines of the RAM. From the VMEbus, character 0 is found at addresses 0-F relative to the character generator's base address, character 1 is found at 10-1F, character N at N0 to NF and so on. This is true independent of the number of scan lines per character that has been programmed. The dot row output of the character generator is clocked out to the video monitor MSB first so that D8 of the memory word corresponds to the leftmost pixel on the screen. When the characters are less than 9 dots wide, the active video dot row data must be left justified (i.e. start at D8 of the word). If 8-bit wide proms are used, jumpers J5 and J2 allow D0 and D8 to be tied together. The PROM table can be generated in the usual fashion, then each byte rotated right once to adjust for this.

2.5 COLOR / MONOCHROME ATTRIBUTES CONTROLLER

The composition of the video signal is determined by the 2675 Video Attribute Controller. It accepts data from the Character generator and attribute RAMs, as well as signals from the AVDC to produce the video signal. In the table of section 2.3, the indicated attribute condition is activated by writing a zero to the appropriate bit. Control port outputs determine the border color, the cursor mode, and whether the 2675 operates with dot stretching or dot modulation. Jumpers connected to the 2675 select monochrome or color mode operation and video timing with 7,8,9, or 10 dots per character. Note that when the 2675 operates with 10 dots per character, the first and last pixels in each dot row of a character block are identical. This is, of course, not suitable for graphics applications. When PROMs are used for the character generator, only 8 bits of storage per character dot row are available. Substituting a 2675C for the 2675B allows 6 dots per character in place of the 7 dot option. This is appropriate for applications which alternate between 80 and 132 characters per row.

More detailed information about the 2675 may be found in its data sheet supplied with this document.

2.6 MEMORY AND I/O ADDRESSES

A contiguous block of VMEbus memory is assigned to the VMEcrt board. The standard base address is FC0000. The block extends to FCA05F. The figure below shows individual assignments within the block.

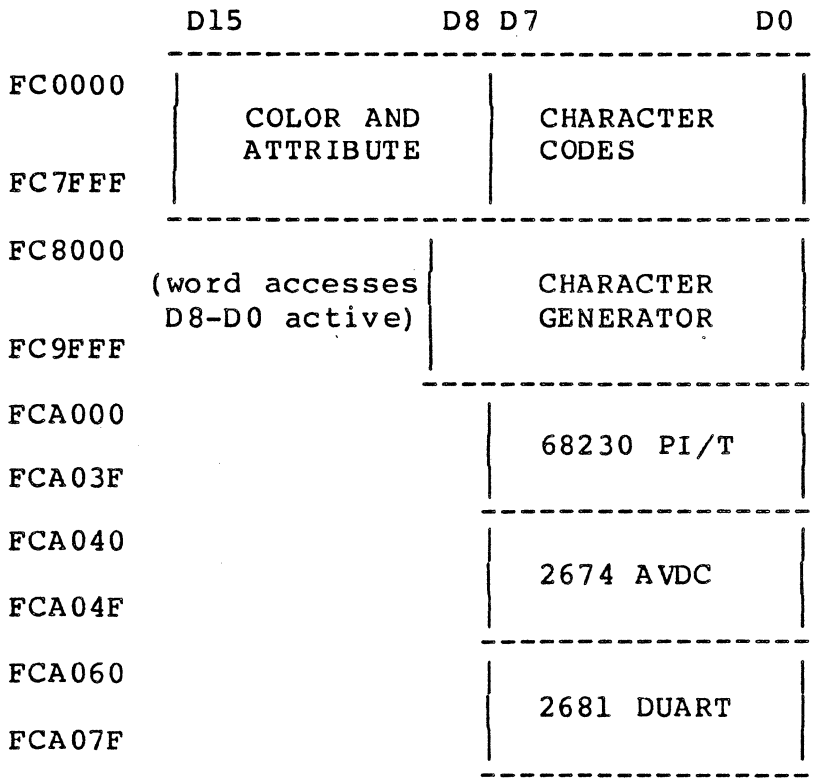


Figure 2-2. VMEcrt Memory Map

2.7 KEYBOARD INPUT PORT

An 8-bit parallel I/O port is provided for access to an encoded keyboard. The interface includes a strobe which may be programmed to be active high or active low, an acknowledge output with similar programmability, 8 data lines and an active low reset. This port is the B port of the 68230 PI/T chip addressed at FCA000. The user may consult its data sheet for programming details.

2.8 SERIAL I/O

Two full duplex asynchronous serial I/O channels are provided using the Signetics 2681 DUART. Its base address is FCA050, and its I/O lines are brought directly to P2 without buffering. P2 pin numbers are given in section 4 of this manual. Ironics' standard SIO adapter modules are available for connection of line drivers and receivers. (Consult the 2681's data sheet for further details.)

2.9 CONTROL PORT DEFINITION

The "A" port of the 68230 PI/T provides 7 control outputs and a general purpose status input. Bits 7,6, and 5 select the border color as specified below. Bits 4, 3, 2, and 1 are connected to pins 24, 9, 31, and 10 respectively of the 2675. These select the dot stretching or dot modulation mode of the 2675 and determine the cursor mode. The truth tables below define their effects.

BORDER COLOR SELECTION

B7	B6	B5	COLOR
0	0	0	BLACK
0	0	1	BLUE
0	1	0	GREEN
0	1	1	CYAN
1	0	0	RED
1	0	1	MAGENTA
1	1	0	YELLOW
1	1	1	WHITE

TABLE 2-5. Border Color Selection Codes

CURSOR MODE

B3=1	Cursor is white
B3=0	Cursor is reverse video, RGB outputs are inverted.

2675 VIDEO MODE

ADOTM	DOTM	DOTS	MODE DESCRIPTION
B4	B2	B1	
0	0	0	Each dot is active for a full dot clock period. Chip operates at full dot rate.
0	0	1	Dots or pixels are stretched to a minimum width of 2 dot clocks.
0	1	0	Each pixel is 2 dot clocks wide; the chip operates at half the dot clock frequency.
0	1	1	Not allowed.
1	0	0	Not allowed.
1	0	1	Not allowed.
1	1	0	Each pixel is two dot clocks wide. Video is active for only half of the pixel time.
1	1	1	Not allowed.

2.10 VIDEO FREQUENCY CALCULATION

Whenever the physical screen format (the number of characters per row, dots per character, or rows per screen) changes, the video frequency must also change. This section details the calculation of the video frequency. To affect the change, the crystal oscillator, U46, must be replaced with an oscillator of the appropriate frequency and the AVDC's initialization registers must be reprogrammed to reflect the new format. A jumper option to divide the crystal oscillator's output by 2 is provided. Selection of the dot modulation mode in the 2675 also has the effect of dividing the dot clock frequency by two. The need for a different dot clock frequency can sometimes be avoided by increasing the blanking periods or selecting a different number of dots per character. The dot clock frequency should always be chosen in conjunction with the screen format to result in a frame rate, or vertical frequency, within .01% of 60 Hz (30 Hz, if interlaced). If not, "swimming" of the display may result when the monitor is subject to external 60 Hz magnetic fields. Note that the 2674 has an AC line lock input which can be used to provide

exact synchronization to the AC line rate.

The horizontal frequency is the total number of scan lines multiplied by the vertical frame rate. The latter parameter is usually made equal to the AC line frequency- 50 or 60 Hz. The calculation of horizontal frequency is outlined below:

10	scan lines per character row	IR0
x 48	character rows per screen	IR4
480	active scan lines	
+ 4	scan lines vertical front porch	IR3
+ 3	scan lines vertical sync	fixed
+ 13	scan lines vertical back porch	IR3
500	total scan lines per frame	
x 60	vertical frame rate (Hz.)	
30.00	Khz horizontal frequency	

The video frequency is the product of the horizontal frequency, the number of character periods per scan lines, and the number of pixels per character. This calculation is outlined below:

80	characters of active video	IR5
+ 5	characters horizontal front porch	IR1
+ 8	characters horizontal sync width	IR2
+ 9	characters horizontal back porch	IR2
102	total character periods per scan line	
x30.00	khz horizontal frequency	
3.06	MHz character clock frequency	
x 8	pixels per character	J31,J32
24.48	MHz video frequency	

In the calculations above, the rightmost entries indicate, with two exceptions, which of the initialization registers of the 2674 AVDC must be programmed with data derived from each parameter. The vertical sync width is fixed at 3 scan lines. The number of dots per character is dependent on the strapping of pins 34 and 35 of the 2675. The horizontal front porch is not entered directly into IR1 but is instead used to calculate an equalizing constant. Refer to figure 15 of the 2674 data sheet for detailed

definition of the initialization registers' contents. Refer to your monitor's documentation for minimum sync requirements, even at this maximum there is still some leaway for sync timing adjustments.

VIDEO TIMING APPROXIMATIONS FOR VARIOUS FORMATS

34X80 at 9X12	(34+2)X12X60=25920 horizontal X 100 chars = 2.592 MHZ CCLK; 193 ns memory cycle time for dual access; 20.736 MHz at 8 dots per char; 23.328 MHz video at 9 dots.
48X80 at 8X8	(48+2) X 8 scan lines per char row X 60 = 24 khz horizontal; 2.4 MHz CCLK; 208 ns memory cycle time for dual access; 19.2 MHz dots at 8 dots per character.
48X80 at 8X10	(48+2) X 10 X 60 = 30 khz horizontal; 3.125 MHz CCLK; 167 ns memory cycle time for dual access; 25 MHz video; note: this is fastest CCLK allowed. corresponds to 640 X 480 graphics format
64X80 at 9X12	66X60X12=47520 Hz horizontal; interlace to get 23.76 khz 2.376 MHz CCLK; 210 ns memory cycle time Dotck=21.384 MHZ
66X102 at 9x12	23.76 khz horizontal when interlaced; 2.851 MHz CCLK; 175 ns memory cycle time 25.659 MHz dot clock; note: fastest dot clock allowed.

3. CONFIGURATION JUMPERS

3.1 INTERRUPT LEVEL

J32-J37 select the level of interrupt acknowledge. One of J39-J44 should be installed to select the corresponding interrupt request level.

LEVEL	REQUEST JUMPER	J32	J33	J34	J35	J36	J37
7	J39	IN	OUT	IN	OUT	IN	OUT
6	J38	IN	OUT	IN	OUT	OUT	IN
5	J44	IN	OUT	OUT	IN	IN	OUT
4	J43	IN	OUT	OUT	INT	OUT	IN
3	J42	OUT	IN	IN	OUT	IN	OUT
2	J41	OUT	IN	IN	OUT	OUT	IN
1	J40	OUT	IN	OUT	IN	IN	OUT

TABLE 3-1. Interrupt Jumpers Configuration

3.2 KEYBOARD RESET

J24 may be inserted to connect the buffered bus RESET* signal to the keyboard input connector section of P2.

3.3 DISPLAY RAM TIMING

J14 is IN and J18 is OUT when 100 ns RAMs (IMS2620-10) are installed. This is the standard configuration of the board. With J14 OUT and J13 IN, lower cost 150 ns (TMS4416-15) RAMs may be used, providing the character clock rate is limited to 2 MHz.

3.4 EXTERNAL VERTICAL SYNC

Inserting J31 allows connection of an external video sync or line rate signal to the AC line lock input of the 2674. If the signal at P2-C2 is a square wave derived from the power line frequency, the 2674 will synchronize its vertical frame rate to it by adding horizontal scan lines to the vertical front porch until the input goes high. To make use of this feature, the 2674 should be programmed to a vertical frequency slightly higher than the synchronizing signal. J31 should also be inserted when the VMEcrt board display is overlaid on the VMEgraf graphics board display.

3.5 INTERLACED OPERATION

Jumpers J15-J23 adapt the IV-1653 for interlaced sync and video operation by shifting the row scan count address inputs to the character generator. For non-interlaced operation, J16, J18, J20 and J22 should be inserted. For interlaced operation, J17, J19, J24 and J23 should be inserted. These jumpers are located in the center of the board. With the bezel up, there will be one pin unconnected at the bottom of the column; this is the non-interlaced configuration. For interlaced operation, merely shift these shunts to leave one pin open at the top of the column.

Bit 7 of IR1 in the 2674 should be cleared for non-interlaced and set for interlaced operation.

3.6 EPROM CHARACTER GENERATOR

When J7 is IN and J8 is OUT, a 2716 EPROM may be installed in U7. This allows 128 non-volatile characters to be defined, such as a standard ASCII set, with 128 additional downloadable characters available. U41 stores data for characters 0-127.

3.7 STATUS INPUT

Bit 0 of the 68230's "B" port may be programmed as a general purpose input. It is normally pulled high. Inserting J9 shorts it to ground.

3.8 CHARACTER WIDTH SELECTION

J3 and J4 select the number of dots per character. Installing them grounds pins 35 and 34 respectively of the 2675. The available options are shown below.

		WIDTH	
J3	J4	2675B	2675C
out	out	9	9
out	in	8	8
in	out	7	6
in	in	10	10

TABLE 3-2. Character Width Selection Jumpers

3.9 COLOR / MONOCHROME MODE

J6 IN selects color mode. J6 OUT selects monochrome mode. This jumper shorts pin 30 of the 2675 to ground. The mode change affects attribute and video output signal definitions as described in the 2675's data sheet.

3.10 GRAPHICS BOARD OVERLAY

The IV-1653 CRT board can be used to provide an alphanumeric overlay for the IV-1651 graphics board. The interconnections required to accomplish this are listed in Section 4.2.

In this mode, the IV-1651 board supplies a dot clock plus horizontal and vertical sync signals to the IV-1653. The graphics video is sent through a multiplexer on the CRT board. The multiplexer's outputs are then sent either directly to a TTL RGB monitor for an 8-color display or back to the graphics board to be routed through the color lookup table for a 16-color display. The CRT board synchronizes with the signals from the graphics board, and then replaces the graphics video with its foreground color whenever its video output is at the foreground color. The alphanumeric display overlays the graphics.

In this mode, the two boards must be programmed to display compatible formats. They must display the same number of pixels vertically and horizontally and be programmed so that their blanking signals are aligned. An exception to this is that the CRT board may be set up to work at half the dot clock frequency of the graphics board or half the number of pixels horizontally.

If the graphics board is setup for a display of 640 H x 480 V pixels with a 60Hz refresh, then the CRT board can be programmed to display 48 lines of 80 characters. Each character should span 8 pixels horizontally and 10 pixels vertically. Both boards would require a 25 MHZ dot clock which would be provided by the IV-1651. Since 25 MHZ is the maximum pixel rate for the IV-1653, if a higher resolution graphics display is desired, the IV-1653 would have to be configured to display at half the graphics board's dot clock frequency.

In addition to the board-to-board interconnect listed in section 4.2, the following jumper configuration must be established:

1. J26 and J27 IN and J25 and J28 OUT to select dot clock or J26 and J28 IN and J25 and J27 OUT to select 1/2 external dot clock

2. J11 IN for positive CSYNC to graphics board if sync on green output is desired

3. J38 IN to enable graphics overlay

4. J31 IN for vertical sync to alignment

5. J48(47) in to send received clock signal to the crt board's synchronizing circuit. If J47 is used the CRT display is shifted to the left two dot position

3.11 SYNC POLARITY

The HSYNC output is active high when J12 is IN and active low when J12 is OUT. The VSYNC/CSYNC output is active high when J11 is IN and active low when J11 is OUT.

3.12 DOT CLOCK SELECTION

Four alternate dot clock sources are provided and are selected as shown below:

SOURCE	J26	J25	J27	J28
internal	out	in	in	out
internal/2	out	in	out	in
external	in	out	in	out
external/2	in	out	out	in

TABLE 3-3. Clock Source/Rate Selection

One of the external clocks should be selected when doing a graphics overlay.

3.13 BORDER WIDTH ADJUSTMENT ENABLE

When J1 is IN, trimpots VR1 and VR2 may be used to adjust the width of the display's border. When J1 is OUT, the selected border color is active for the entire blanking interval. In order to adjust the border width at both ends of the screen, it is necessary to understand the timing intervals controlled by potentiometers VR1 and VR2. In the timing diagram below, note that the blanking interval is divided into three sub-intervals. The first interval (I) is controlled by the adjustment of VR1, and directly adjusts the border width at the right side of the display. The

second interval (II) is the retrace interval and is directly controlled by VR2. The remaining time (III) in the blanking interval is proportional to the border width at the left side of the screen.

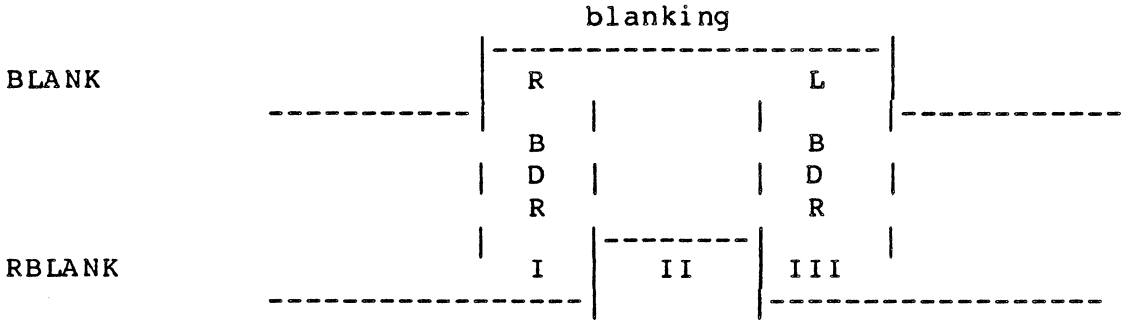


Figure 3-1. Border Adjustment Timing

3.14 STANDARD CONFIGURATION (Stand alone operation)

J14	100ns RAMS
J16,J18,J20,J22	non-interlaced display
J4,J5	8 bits per character
J11	positive HSYNC
J12	positive VSYNC/CSYNC output
J6	color mode of 2675
J7	RAM (not Eprom in U7)
J25, J27	internal dot clock
J29	non overlay mode
J48	reset gate/synchronizing signal.
J10 out!	Must be out for stand-alone mode.

3.15 OVERLAY CONFIGURATION

Start from standard configuration (section 3.14)
Remove: J25, J29
Insert: J26, J30, J12, J31, J10, J46

4. P2 CONNECTIONS

4.1 SIGNAL DEFINITIONS

In keeping with Ironics' practice of using P2 rows "A" and "C" for I/O, the video outputs, serial I/O lines, parallel keyboard inputs, and interboard synchronization signals are terminated on these rows of P2. The pins used are shown below.

PIN #	FUNCTION
C7	external dot clock
C3	external horizontal sync
C2	external video sync/ ac line lock
C8	graphics video 0 input
C9	graphics video 1 input
C11	graphics video 2 input
C13	graphics video 3 input
C10	overlayed graphics video output 4
C12	overlayed graphics video output 1
C14	overlayed graphics video output 2
C15	overlayed graphics video output 3
C16	vertical 1 composite sync output
C24	Luminance video output
C25	Vertical sync or composite sync
C26	Horizontal sync
C27	Blue video output
C29	Red video output
C31	Green video output
C32	Vertical sync or composite sync
A1	TTL channel A serial receive data
A2	TTL channel A control line transmit data
A3	TTL output 0 control line
A5	TTL channel B transmit data
A4	TTL channel B receive data
A6	TTL output 1 control line
A7	TTL input 2 control line
A13	Keyboard input strobe
A27	Keyboard data 7
A26	" " 6
A25	" " 5
A24	" " 4
A28	" " 3
A29	" " 2
A30	" " 1
A31	" " 0
A32	Keyboard RESET*

4.2 BACKPLANE INTERCONNECT FOR GRAPHICS OVERLAYED WITH ALPHANUMERIC

CRT Board IV-1653		Graphics Board IV-1651
P2-C2		P2-C2
-C3		-C3
-C7		-C7
-C8		-C8
-C9		-C9
-C11		-C11
-C13		-C13
-C29	(C14*)	-C14
-C31	(C15*)	-C15
-C27	(C10*)	-C10
-C24	(C12*)	-C12
P2-C25		P2-C25

* Connections required for Rev. 1.2

5. SOFTWARE

5.1 INTRODUCTION

The following code is provided as an illustration of the software necessary to drive the IV-1653 CRT card. The first section to follow illustrates a C language structure definition of the registers on the VMEcrt card. The sections which follow contain code to initialize the video controller chips, sample font data, the color/monochrome attributes controller chip, and a simple type and display driver for the VMEcrt card.

A VT100 terminal emulator is available for those Ironics users running UNIX on our VME development system. Contact your local representative for more information.

5.2 C LANGUAGE STRUCTURE DEFINITION OF VMEcrt CARD

```
/* srtucture of the Ironics IV1653 VME CRT display controller */

struct crt {
        short
           display[16384],/*Display Ram,as [<attributes><char. code>] */
           chargen[4096]; /* Character Generator Ram/Rom, as:        */
                          /*   16 locations by 9 bits per char.      */

/* Note: All registers below are low (odd) bytes in 16 bit locations,*/
/*       they are therefore defined with a dummy name for each       */
/*       high (even) byte associated.                                */
/*                                                                   */

/* structure of 68230 PI/T chip */
        unsigned char
                dum00,  pgcr,   /*port general control reg. */
                dum01,  psrr,   /**/
                dum02,  padir,  /*port A direction*/
                dum03,  pbdir,  /*port B direction*/
                dum04,  pcdir,  /*port C direction*/
                dum05,  pivr,   /*interupt vector*/
                dum06,  pacr,   /*port A control reg. */
                dum07,  pbcr,   /*port B control reg. */
                dum08,  padat,  /*port A data*/
                dum09,  pbdat,  /*port B data*/
                dum10,  paar,   /**/
                dum11,  pbar,   /**/
                dum12,  pcdat,  /**/
                dum13,  psr,    /**/
                pitresv0[4],    /**/
                dum14,  tcr,    /**/
                dum15,  tivr,   /**/
                dum16,  dum17,  /**/
```

```
                dum18,  cprh,   /**/
                dum19,  cprm,   /**/
                dum20,  cprl,   /**/
                dum21,  dum22,  /**/
                dum23,  cntrh,  /**/
                dum24,  cntrm,  /**/
                dum25,  cntrl,  /**/
                dum26,  tsr,    /**/
                pitresvl[10];   /**/

/* 2674 Display Controller */
        unsigned char
                dum27,  irp,    /* Initialization/Interupt register*/
                dum28,  stc,    /* Command/Status register         */
                dum29,  ssl,    /* Screen Start address low byte   */
                dum30,  ssh,    /* Screen Start address high byte  */
                dum31,  cul,    /* Cursor address low byte         */
                dum32,  cuh,    /* Cursor address high byte        */
                dum33,  pal,    /* Pointer address low byte        */
                dum34,  pah,    /* Pointer address high byte       */
                vdcresv0[16];

/* 2681 Dual UART */
        unsigned char
                dum35,  mra,    /* Mode register 1/2 (A)          */
                dum36,  sra,    /* Status register (A)            */
                dum37,  cmda,   /* Command register (A)           */
                dum38,  adat,   /* Data register (A)              */
                dum39,  acr,    /* Aux. Control register          */
                dum40,  ireg,   /* Interupt register              */
                dum41,  cthi,   /* Count high byte                */
                dum42,  ctlo,   /* Count low byte                 */
                dum43,  mrb,    /* Mode register 1/2 (B)          */
                dum44,  srb,    /* Status register (B)            */
                dum45,  cmdb,   /* Command register (B)           */
                dum46,  bdat,   /* Data register (B)              */
                dum47,  res1,   /* RESERVED                       */
                dum48,  opcr,   /* Port register                  */
                dum49,  strt,   /* Start + output port ctl        */
                dum50,  stop;   /* Stop + output port ctl         */
};
```

5.3 FONT DATA FILE

```
/* 5 x 7 character font, left top justified in 8x8 */

static char font[] ={
        0x00, 0xf8, 0x88, 0x88, 0x88, 0x88, 0x88, 0xf8, /* 0x0 */
        0x00, 0xf8, 0x80, 0x80, 0x80, 0x80, 0x80, 0x80, /* 0x1 */
        0x00, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0xf8, /* 0x2 */
        0x00, 0x08, 0x08, 0x08, 0x08, 0x08, 0x08, 0xf8, /* 0x3 */
        0x00, 0x40, 0x20, 0x10, 0x78, 0x20, 0x10, 0x08, /* 0x4 */
        0x00, 0xf8, 0x88, 0xd8, 0xa8, 0xd8, 0x88, 0xf8, /* 0x5 */
        0x00, 0x00, 0x08, 0x10, 0xa0, 0xc0, 0x80, 0x00, /* 0x6 */
        0x00, 0x70, 0x88, 0x88, 0xf8, 0x50, 0x50, 0xd8, /* 0x7 */
        0x00, 0x20, 0x40, 0xf0, 0x48, 0x28, 0x08, 0x08, /* 0x8 */
        0x00, 0x00, 0x20, 0x10, 0xf8, 0x10, 0x20, 0x00, /* 0x9 */
        0x00, 0xf8, 0x00, 0x00, 0xf8, 0x00, 0x00, 0xf8, /* 0xa */
        0x00, 0x00, 0x20, 0x20, 0xa8, 0x70, 0x20, 0x00, /* 0xb */
        0x00, 0x20, 0xa8, 0x70, 0x20, 0xa8, 0x70, 0x20, /* 0xc */
        0x00, 0x00, 0x20, 0x40, 0xf8, 0x40, 0x20, 0x00, /* 0xd */
        0x00, 0x70, 0x88, 0xd8, 0xa8, 0xd8, 0x88, 0x70, /* 0xe */
        0x00, 0x70, 0x88, 0x88, 0xa8, 0x88, 0x88, 0x70, /* 0xf */
        0x00, 0xf8, 0x88, 0x88, 0xf8, 0x88, 0x88, 0xf8, /* 0x10 */
        0x00, 0x70, 0xa8, 0xa8, 0xb8, 0x88, 0x88, 0x70, /* 0x11 */
        0x00, 0x70, 0x88, 0x88, 0xb8, 0xa8, 0xa8, 0x70, /* 0x12 */
        0x00, 0x70, 0x88, 0x88, 0xe8, 0xa8, 0xa8, 0x70, /* 0x13 */
        0x00, 0x70, 0xa8, 0xa8, 0xe8, 0x88, 0x88, 0x70, /* 0x14 */
        0x00, 0x00, 0x28, 0x10, 0xa8, 0xc0, 0x80, 0x00, /* 0x15 */
        0x00, 0x70, 0x50, 0x50, 0x50, 0x50, 0x50, 0xd8, /* 0x16 */
        0x00, 0x08, 0x08, 0x08, 0xf8, 0x08, 0x08, 0x08, /* 0x17 */
        0x00, 0xf8, 0x88, 0x50, 0x20, 0x50, 0x88, 0xf8, /* 0x18 */
        0x00, 0x20, 0x20, 0x70, 0x70, 0x20, 0x20, 0x20, /* 0x19 */
        0x00, 0x70, 0x88, 0x80, 0x40, 0x20, 0x00, 0x20, /* 0x1a */
        0x00, 0x70, 0x88, 0x88, 0xf8, 0x88, 0x88, 0x70, /* 0x1b */
        0x00, 0xf8, 0xa8, 0xa8, 0xe8, 0x88, 0x88, 0xf8, /* 0x1c */
        0x00, 0xf8, 0x88, 0x88, 0xe8, 0xa8, 0xa8, 0xf8, /* 0x1d */
        0x00, 0xf8, 0x88, 0x88, 0xb8, 0xa8, 0xa8, 0xf8, /* 0x1e */
        0x00, 0xf8, 0xa8, 0xa8, 0xb8, 0x88, 0x88, 0xf8, /* 0x1f */
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0x20 */
        0x20, 0x20, 0x20, 0x20, 0x20, 0x00, 0x20, 0x00, /* 0x21 */
        0x50, 0x50, 0x50, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0x22 */
        0x50, 0x50, 0xf8, 0x50, 0xf8, 0x50, 0x50, 0x00, /* 0x23 */
        0x20, 0x78, 0xa0, 0x70, 0x28, 0xf0, 0x20, 0x00, /* 0x24 */
        0xc0, 0xc8, 0x10, 0x20, 0x40, 0x98, 0x18, 0x00, /* 0x25 */
        0x40, 0xa0, 0xa0, 0x40, 0xa8, 0x90, 0x68, 0x00, /* 0x26 */
        0x20, 0x20, 0x20, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0x27 */
        0x20, 0x40, 0x80, 0x80, 0x80, 0x40, 0x20, 0x00, /* 0x28 */
        0x20, 0x10, 0x08, 0x08, 0x08, 0x10, 0x20, 0x00, /* 0x29 */
        0x20, 0xa8, 0x70, 0x20, 0x70, 0xa8, 0x20, 0x00, /* 0x2a */
        0x00, 0x20, 0x20, 0xf8, 0x20, 0x20, 0x00, 0x00, /* 0x2b */
        0x00, 0x00, 0x00, 0x00, 0x20, 0x20, 0x40, 0x00, /* 0x2c */
        0x00, 0x00, 0x00, 0xf8, 0x00, 0x00, 0x00, 0x00, /* 0x2d */
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x20, 0x00, /* 0x2e */
```

```
0x00, 0x08, 0x10, 0x20, 0x40, 0x80, 0x00, 0x00, /* 0x2f */
0x70, 0x88, 0x98, 0xa8, 0xc8, 0x88, 0x70, 0x00, /* 0x30 */
0x20, 0x60, 0x20, 0x20, 0x20, 0x20, 0x70, 0x00, /* 0x31 */
0x70, 0x88, 0x08, 0x30, 0x40, 0x80, 0xf8, 0x00, /* 0x32 */
0xf8, 0x08, 0x10, 0x30, 0x08, 0x88, 0x70, 0x00, /* 0x33 */
0x10, 0x30, 0x50, 0x90, 0xf8, 0x10, 0x10, 0x00, /* 0x34 */
0xf8, 0x80, 0xf0, 0x08, 0x08, 0x88, 0x70, 0x00, /* 0x35 */
0x38, 0x40, 0x80, 0xf0, 0x88, 0x88, 0x70, 0x00, /* 0x36 */
0xf8, 0x08, 0x10, 0x20, 0x40, 0x40, 0x40, 0x00, /* 0x37 */
0x70, 0x88, 0x88, 0x70, 0x88, 0x88, 0x70, 0x00, /* 0x38 */
0x70, 0x88, 0x88, 0x78, 0x08, 0x10, 0xe0, 0x00, /* 0x39 */
0x00, 0x00, 0x20, 0x00, 0x20, 0x00, 0x00, 0x00, /* 0x3a */
0x00, 0x00, 0x20, 0x00, 0x20, 0x20, 0x40, 0x00, /* 0x3b */
0x10, 0x20, 0x40, 0x80, 0x40, 0x20, 0x10, 0x00, /* 0x3c */
0x00, 0x00, 0xf8, 0x00, 0xf8, 0x00, 0x00, 0x00, /* 0x3d */
0x40, 0x20, 0x10, 0x08, 0x10, 0x20, 0x40, 0x00, /* 0x3e */
0x70, 0x88, 0x10, 0x20, 0x20, 0x00, 0x20, 0x00, /* 0x3f */
0x70, 0x88, 0xa8, 0xb8, 0xb0, 0x80, 0x78, 0x00, /* 0x40 */
0x20, 0x50, 0x88, 0x88, 0xf8, 0x88, 0x88, 0x00, /* 0x41 */
0xf0, 0x88, 0x88, 0xf0, 0x88, 0x88, 0xf0, 0x00, /* 0x42 */
0x70, 0x88, 0x80, 0x80, 0x80, 0x88, 0x70, 0x00, /* 0x43 */
0xf0, 0x88, 0x88, 0x88, 0x88, 0x88, 0xf0, 0x00, /* 0x44 */
0xf8, 0x80, 0x80, 0xf0, 0x80, 0x80, 0xf8, 0x00, /* 0x45 */
0xf8, 0x80, 0x80, 0xf0, 0x80, 0x80, 0x80, 0x00, /* 0x46 */
0x78, 0x80, 0x80, 0x80, 0x98, 0x88, 0x78, 0x00, /* 0x47 */
0x88, 0x88, 0x88, 0xf8, 0x88, 0x88, 0x88, 0x00, /* 0x48 */
0x70, 0x20, 0x20, 0x20, 0x20, 0x20, 0x70, 0x00, /* 0x49 */
0x08, 0x08, 0x08, 0x08, 0x08, 0x88, 0x70, 0x00, /* 0x4a */
0x88, 0x90, 0xa0, 0xc0, 0xa0, 0x90, 0x88, 0x00, /* 0x4b */
0x80, 0x80, 0x80, 0x80, 0x80, 0x80, 0xf8, 0x00, /* 0x4c */
0x88, 0xd8, 0xa8, 0xa8, 0x88, 0x88, 0x88, 0x00, /* 0x4d */
0x88, 0x88, 0xc8, 0xa8, 0x98, 0x88, 0x88, 0x00, /* 0x4e */
0x70, 0x88, 0x88, 0x88, 0x88, 0x88, 0x70, 0x00, /* 0x4f */
0xf0, 0x88, 0x88, 0xf0, 0x80, 0x80, 0x80, 0x00, /* 0x50 */
0x70, 0x88, 0x88, 0x88, 0xa8, 0x90, 0x68, 0x00, /* 0x51 */
0xf0, 0x88, 0x88, 0xf0, 0xa0, 0x90, 0x88, 0x00, /* 0x52 */
0x70, 0x88, 0x80, 0x70, 0x08, 0x88, 0x70, 0x00, /* 0x53 */
0xf8, 0x20, 0x20, 0x20, 0x20, 0x20, 0x20, 0x00, /* 0x54 */
0x88, 0x88, 0x88, 0x88, 0x88, 0x88, 0x70, 0x00, /* 0x55 */
0x88, 0x88, 0x88, 0x88, 0x88, 0x50, 0x20, 0x00, /* 0x56 */
0x88, 0x88, 0x88, 0xa8, 0xa8, 0xd8, 0x88, 0x00, /* 0x57 */
0x88, 0x88, 0x50, 0x20, 0x50, 0x88, 0x88, 0x00, /* 0x58 */
0x88, 0x88, 0x50, 0x20, 0x20, 0x20, 0x20, 0x00, /* 0x59 */
0xf8, 0x08, 0x10, 0x20, 0x40, 0x80, 0xf8, 0x00, /* 0x5a */
0xf8, 0xc0, 0xc0, 0xc0, 0xc0, 0xc0, 0xf8, 0x00, /* 0x5b */
0x00, 0x80, 0x40, 0x20, 0x10, 0x08, 0x00, 0x00, /* 0x5c */
0xf8, 0x18, 0x18, 0x18, 0x18, 0x18, 0xf8, 0x00, /* 0x5d */
0x00, 0x00, 0x20, 0x50, 0x88, 0x00, 0x00, 0x00, /* 0x5e */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xf8, /* 0x5f */
0x40, 0x20, 0x10, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0x60 */
0x00, 0x00, 0x00, 0x70, 0x08, 0x78, 0x88, 0x78, /* 0x61 */
```

```
0x00, 0x80, 0x80, 0xb0, 0xc8, 0x88, 0xc8, 0xb0, /* 0x62 */
0x00, 0x00, 0x00, 0x70, 0x88, 0x80, 0x88, 0x70, /* 0x63 */
0x00, 0x08, 0x08, 0x68, 0x98, 0x88, 0x98, 0x68, /* 0x64 */
0x00, 0x00, 0x00, 0x70, 0x88, 0xf8, 0x80, 0x70, /* 0x65 */
0x00, 0x10, 0x28, 0x20, 0x70, 0x20, 0x20, 0x20, /* 0x66 */
0x00, 0x68, 0x98, 0x98, 0x68, 0x08, 0x88, 0x70, /* 0x67 */
0x00, 0x80, 0x80, 0xb0, 0xc8, 0x88, 0x88, 0x88, /* 0x68 */
0x00, 0x20, 0x00, 0x60, 0x20, 0x20, 0x20, 0x70, /* 0x69 */
0x00, 0x08, 0x00, 0x08, 0x08, 0x08, 0x88, 0x70, /* 0x6a */
0x00, 0x80, 0x80, 0x90, 0xa0, 0xc0, 0xa0, 0x90, /* 0x6b */
0x00, 0x60, 0x20, 0x20, 0x20, 0x20, 0x20, 0x70, /* 0x6c */
0x00, 0x00, 0x00, 0xd0, 0xa8, 0xa8, 0xa8, 0xa8, /* 0x6d */
0x00, 0x00, 0x00, 0xb0, 0xc8, 0x88, 0x88, 0x88, /* 0x6e */
0x00, 0x00, 0x00, 0x70, 0x88, 0x88, 0x88, 0x70, /* 0x6f */
0x00, 0xb0, 0xc8, 0x88, 0xc8, 0xb0, 0x80, 0x80, /* 0x70 */
0x00, 0x68, 0x98, 0x88, 0x98, 0x68, 0x08, 0x08, /* 0x71 */
0x00, 0x00, 0x00, 0xb0, 0xc8, 0x80, 0x80, 0x80, /* 0x72 */
0x00, 0x00, 0x00, 0x78, 0x80, 0x70, 0x08, 0xf0, /* 0x73 */
0x00, 0x20, 0x20, 0xf8, 0x20, 0x20, 0x28, 0x10, /* 0x74 */
0x00, 0x00, 0x00, 0x88, 0x88, 0x88, 0x98, 0x68, /* 0x75 */
0x00, 0x00, 0x00, 0x88, 0x88, 0x88, 0x50, 0x20, /* 0x76 */
0x00, 0x00, 0x00, 0x88, 0x88, 0xa8, 0xa8, 0x50, /* 0x77 */
0x00, 0x00, 0x00, 0x88, 0x50, 0x20, 0x50, 0x88, /* 0x78 */
0x00, 0x88, 0x88, 0x88, 0x78, 0x08, 0x88, 0x70, /* 0x79 */
0x00, 0x00, 0x00, 0xf8, 0x10, 0x20, 0x40, 0xf8, /* 0x7a */
0x00, 0x10, 0x20, 0x20, 0x40, 0x20, 0x20, 0x10, /* 0x7b */
0x00, 0x20, 0x20, 0x20, 0x00, 0x20, 0x20, 0x20, /* 0x7c */
0x00, 0x40, 0x20, 0x20, 0x10, 0x20, 0x20, 0x40, /* 0x7d */
0x00, 0x40, 0xa8, 0x10, 0x00, 0x00, 0x00, 0x00, /* 0x7e */
0x00, 0x50, 0xa8, 0x50, 0xa8, 0x50, 0xa8, 0x50, /* 0x7f */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0x80 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0x81 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0x82 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0x83 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0x84 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0x85 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0x86 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0x87 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0x88 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0x89 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0x8a */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0x8b */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0x8c */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0x8d */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0x8e */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0x8f */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0x90 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0x91 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0x92 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0x93 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0x94 */
```

```
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0x95 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0x96 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0x97 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0x98 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0x99 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0x9a */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0x9b */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0x9c */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0x9d */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0x9e */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0x9f */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xa0 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xa1 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xa2 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xa3 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xa4 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xa5 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xa6 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xa7 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xa8 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xa9 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xaa */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xab */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xac */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xad */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xae */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xaf */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xb0 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xb1 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xb2 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xb3 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xb4 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xb5 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xb6 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xb7 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xb8 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xb9 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xba */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xbb */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xbc */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xbd */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xbe */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xbf */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xc0 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xc1 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xc2 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xc3 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xc4 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xc5 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xc6 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xc7 */
```

```
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xc8 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xc9 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xca */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xcb */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xcc */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xcd */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xce */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xcf */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xd0 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xd1 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xd2 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xd3 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xd4 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xd5 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xd6 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xd7 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xd8 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xd9 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xda */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xdb */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xdc */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xdd */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xde */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xdf */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xe0 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xe1 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xe2 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xe3 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xe4 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xe5 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xe6 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xe7 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xe8 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xe9 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xea */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xeb */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xec */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xed */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xee */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xef */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xf0 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xf1 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xf2 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xf3 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xf4 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xf5 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xf6 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xf7 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xf8 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xf9 */
0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xfa */
```

```
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xfb *
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xfc *
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xfd *
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, /* 0xfe *
        0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00  /* 0xff *
};
```

5.4 INITIALIZATION PROGRAM

```
#include <stdio.h>
#define CBASE 0xFC0000
#include "ivcrt.h"

main()
{
        register struct crt *cp = ((struct crt *)CBASE);
        register int i, j;
#include "font5x7d.h"

        if (phys(0,CBASE,65536,CBASE) == 0 ) {

                for (i = 0; i < 4096; i++) cp->chargen[i]=0xFFFF;
                for (i = 0; i < 128; i++) {
                  for (j = 0; j < 10; j++)
                  cp->chargen[(i*16)+j] =(~(font[(i*10)+j] >> 2));
                }
                for (i = 128; i < 256; i++) {
                  for (j = 0; j < 10; j++)
                  cp->chargen[(i*16)+j] =~font[(i*10)+j];
                }
                for (i = 0; i < 16384; i++)
                        cp->display[i]=0xC800 + (i & 0xFF);
                initpit();      /* init the pit */
                initvdc();      /* init the display controller */

        } else printf("cannot phys0);
} /* main */

initpit()
{
        register struct crt *cp = ((struct crt *)CBASE);
                cp->pgcr  = 0;          /**/
                cp->pacr  = 0xFF;       /**/
                cp->padir = 0;          /**/
                cp->pbcr  = 0x80;       /**/
                cp->pbdir = 0xFE;       /**/
                cp->pbdat = 0x30;       /**/
}

initvdc()
```

```
{
        register struct crt *cp = ((struct crt *)CBASE);
        register int i;
        char hbp, hact, hfp, hs, ec;
        char vbp, vs, vfp, vrows, lpcr;
        static char ir[15] ={
                0x4E,   /* ~Double; 10 Lines/Row; CSYNC; Shared */
                0x21,   /* ~Interlaced; EC = 34 cclk */
                0x1B,   /* ~Row Table; HS = 8 cclk; HBP = 11 cclk */
                0x24,   /* VFP = 8 Lines; VBP = 12 Lines */
                0xAE,   /* Blink Rate = 1/128 Vsync; 48 Rows/Screen */
                0x49,   /* 75 Characters/Row */
                0x09,   /* Cursor first line=0;Cursor last line=10 */
                0x39,   /* VSYNC=3 lines;cursor blink;cursor rate=1/64;
                                underline position = line 10 */
                0x00,   /* display buffer first address = 0 */
                0xF0,   /* display buffer last address = 16383 */
                0x00,   /* display pointer = 0 */
                0x00,   /* ~LZ down; ~LZ up */
                0x00,   /* ~scroll start; split reg 1 = row 1 */
                0x00,   /* ~scroll end; split reg 2 = row 1 */
                0x00    /* double 1 = normal; double 2 = normal;
                                1 lines to scroll */
        };

        waitcm();
        cp->stc = 0;
        waitcm();
        cp->stc = 0; /* reset vdc */

        hbp = 7; hact = 80; hfp = 7; hs = 10;
        vbp = 20; vrows = 48; lpcr = 10; vfp = 8; vs = 3;

        ec = ((hbp + hact + hs + hfp)/2) - (2 * hs);

        ir[0] = (ir[0]&0x87)|((lpcr-1)<<3);
        ir[1] = (ir[1]&0x80)|(ec-1);
        ir[2] = (ir[2]&0x80)|(((hs/2)-1)<<3)|(((hbp-3)/4)+1);
        ir[3] = (((vfp-4)/4)<<5)|((vbp-4)/2);
        ir[4] = (ir[4]&0x80)|(vrows-1);
        ir[5] = (hact-1);
        ir[7] = (ir[7] & 0x3F);
        switch (vs) {
                case 1: ir[7] |= 0x40; break;
                case 3: ir[7] |= 0x00; break;
                case 5: ir[7] |= 0x80; break;
                case 7: ir[7] |= 0xC0; break;
                default: break;
        }

        for (i =0; i < 15; i++) {
```

```
                cp->irp = ir[i];
                printf("%02x0,ir[i]);
        }
        cp->ssl = 0;
        cp->ssh = 0;
        cp->cul = 0;
        cp->cuh = 0;
        cp->pal = 0xFF;
        cp->pah = 0x07;

        waitcm();
        cp->stc = 0x29; /* turn display on */

}

waitcm()
{
        register struct crt *cp = ((struct crt *)CBASE);

        while (((cp->stc) & 0x20) == 0) ;
}
```

5.5 CHARACTER DISPLAY PROGRAM

```
/* Example display driver for IV-1653 CRT Board                    */
/*      Uses Brute Force Scrolling                                 */
#include <stdio.h>
#define CBASE 0xFC0000
#include "ivcrt.h"
#define LL      80      /* number of characters per line */
#define SL      24      /* number of rows per screen */

main()
{
        register struct crt *cp = ((struct crt *)CBASE);
        register int c, curs;

        if (phys(0,CBASE,65536,CBASE) == 0 ) {/* get the board from UNIX*/

        curs = LL*(SL-1);
        cp->stc = 0x31; /* turn on cursor */
        while((c = fgetc(stdin)) != EOF) {
        switch (c) {
case '0':       scroll();
                curs = LL*(SL-1);
                cp->cul = (curs&0xFF); cp->cuh = (curs&0xFF00)>>8;
                break;
default:        cp->display[curs++] = 0xC700 | (0xFF & c);
                if (curs >= (LL*SL)) {
```

```
                        curs = LL*(SL-1);
                        scroll();
                        cp->cul = (curs&0xFF);cp->cuh = (curs&0xFF00)>>8;
                        break;
                }
                waitcm();
                cp->stc = 0xA9; /* increment cursor */
        }
        }
        } else printf("display: cannot phys0);
} /* main */

scroll()
{
        register struct crt *cp = ((struct crt *)CBASE);
        register int i;

        for (i = 0; i < (LL*(SL-1)); i++) cp->display[i]=cp->display[i+LL
        for (i = (LL*(SL-1)); i < LL*SL; i++) cp->display[i]=0xC720;
}

waitcm()
{
        register struct crt *cp = ((struct crt *)CBASE);
        while (((cp->stc) & 0x20) == 0) ;
}
```

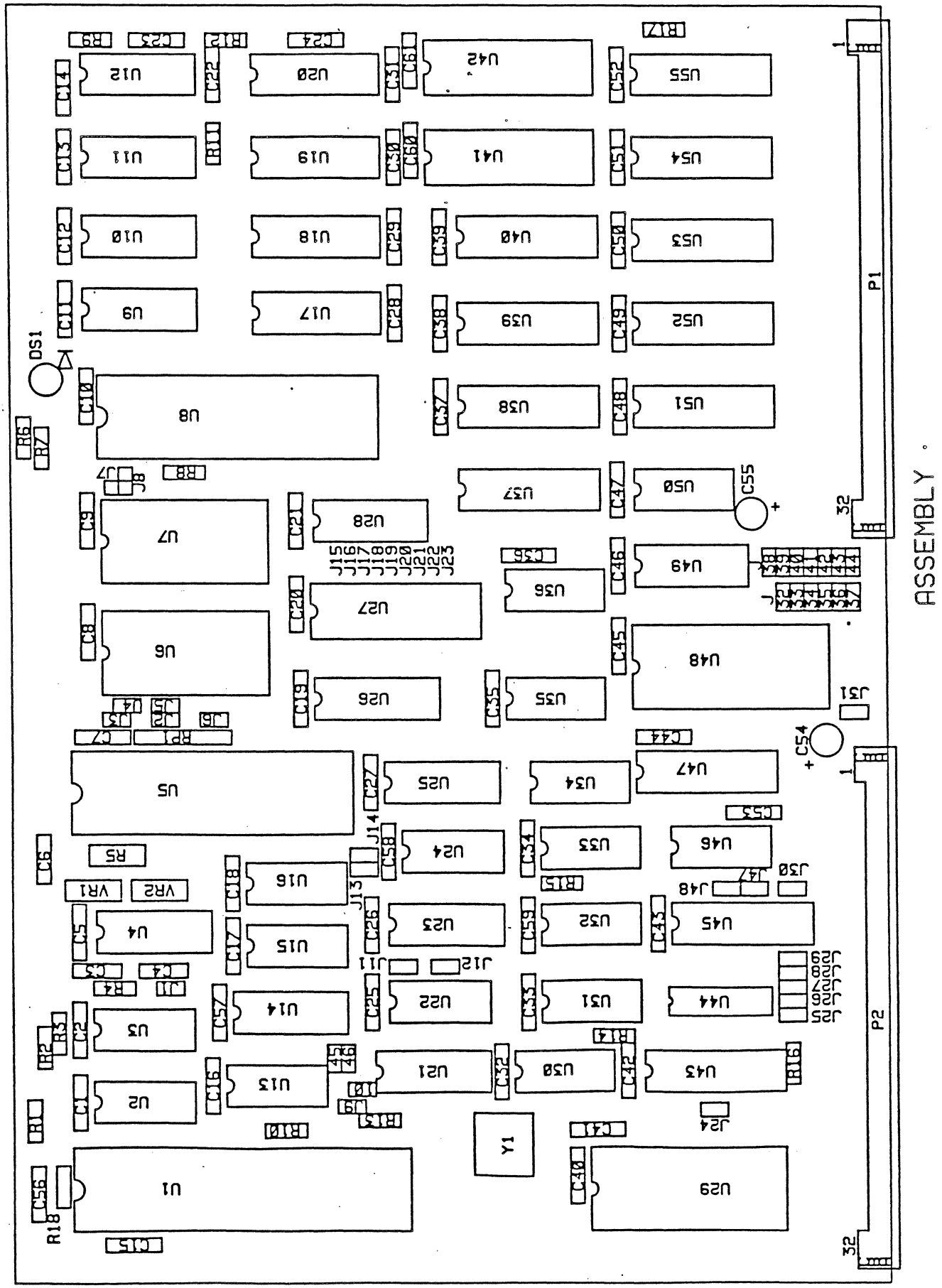
6. APPENDIX

6.1 PARTS LIST

PART	Description
U1	68230L8
	48-PIN SOCKET
U2	MLPSLDL-15
U3, 36, 50	74F04
U4	74LS221
U5	SC2675TC4N40
	40-PIN SOCKET
U6, 7	2716/AM9128
	24-PIN SOCKET
U8	SC2674BC4N40
U9-12	74LS258
U13,24,30,34	74F74
U14,23	74F112 "S" acceptable
U15	74F32
U16	TTLDM-150
U17-20	IMS2620P-10
U21	74F399
U22	74F86
U25	74F257
U26	6147P70 OR 2147D-2 NEC
U27,41,42	8X371 SIGNETICS
U28	74LS174
U29	2681 SIGNETICS
	28-PIN SOCKET
U31	74LS164
U32,35	74F00
U33	74F20
U37,51,52,43	74F244
U38	74F534
U39	74F373
U40	74LS533
U44	25MHz OSCILLATOR
	OSCILLATOR SOCKET
U45	74F374
U46	7438
U47	82S153 - 215.3
	20-PIN SOCKET
U48	82S100 - 216.2
	28-PIN SOCKET
U49	74LS375
U53	82S153 - 217.0
	20-PIN SOCKET
U54,55	74F245

VR1,VR2	CTS-375 20KB 20K TRIM POT
RPI	4.7K SIP 1-472 8X-1-472
Y1	5.6864 MHz. XTAL
R1,2,4,7,8,10	
R13,16,17	4.7K
R3,14,15	1K
R6	470
R9,11,12	120
R5	31.6 1% 1/2 w
R18	100
DS1	TIL 220
C1,2,5-21,25-40 42-53,57-61	.1 uf DIPGUARD
C22-24	200 pf.
C41	20 pf. CD
C54	47/16v.
C55	100/16v.
C56	150 pf. CD
C3	1 n
C4	2 n
CF	.1 DIP Cap.
P1,P2	ERNI 533.602 CONN
PC.	C01-1605-1.2
J1-14,16-48	DUAL .1" HEADER

6.2 COMPONENTS LOCATION DIAGRAM

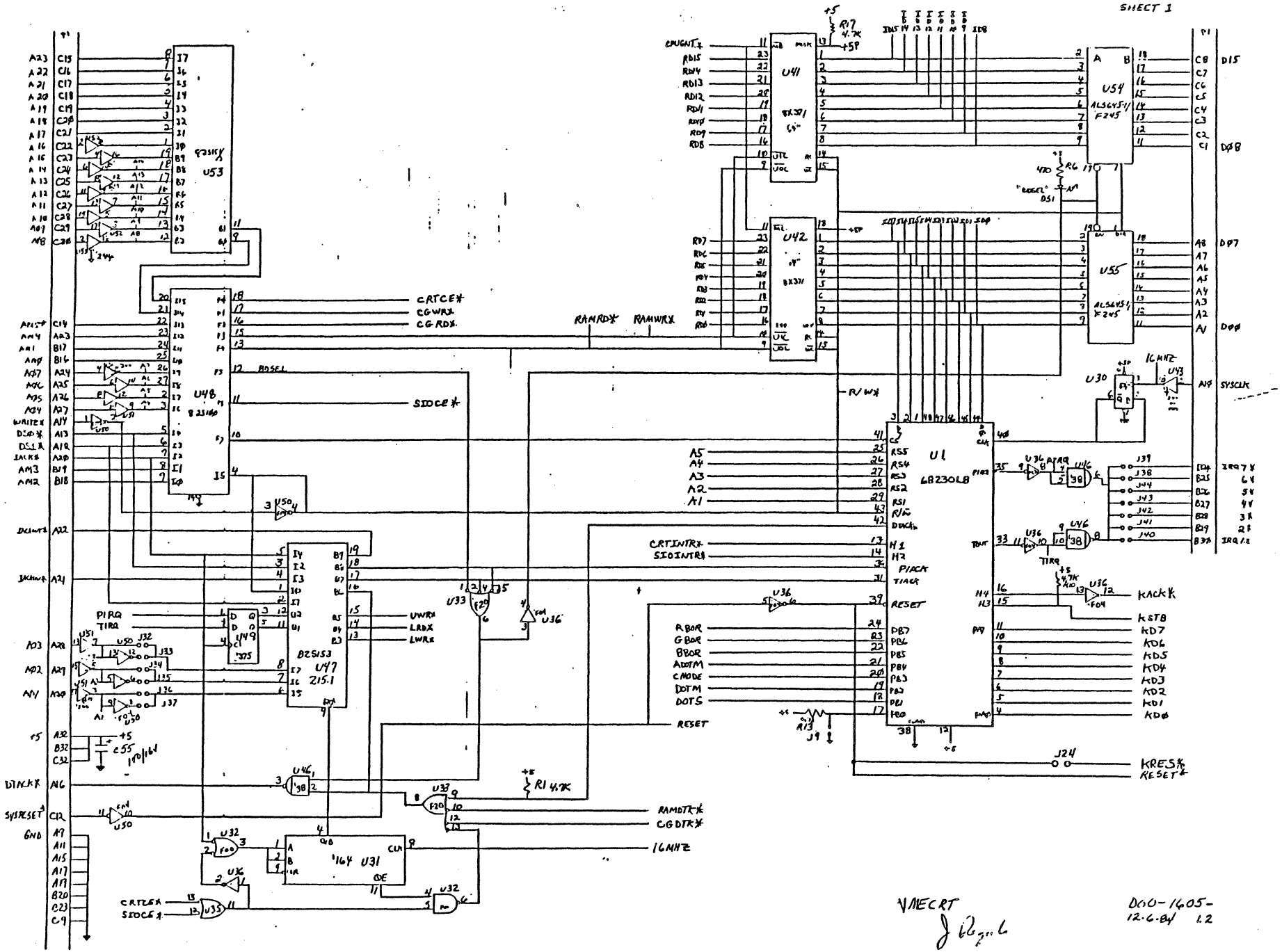


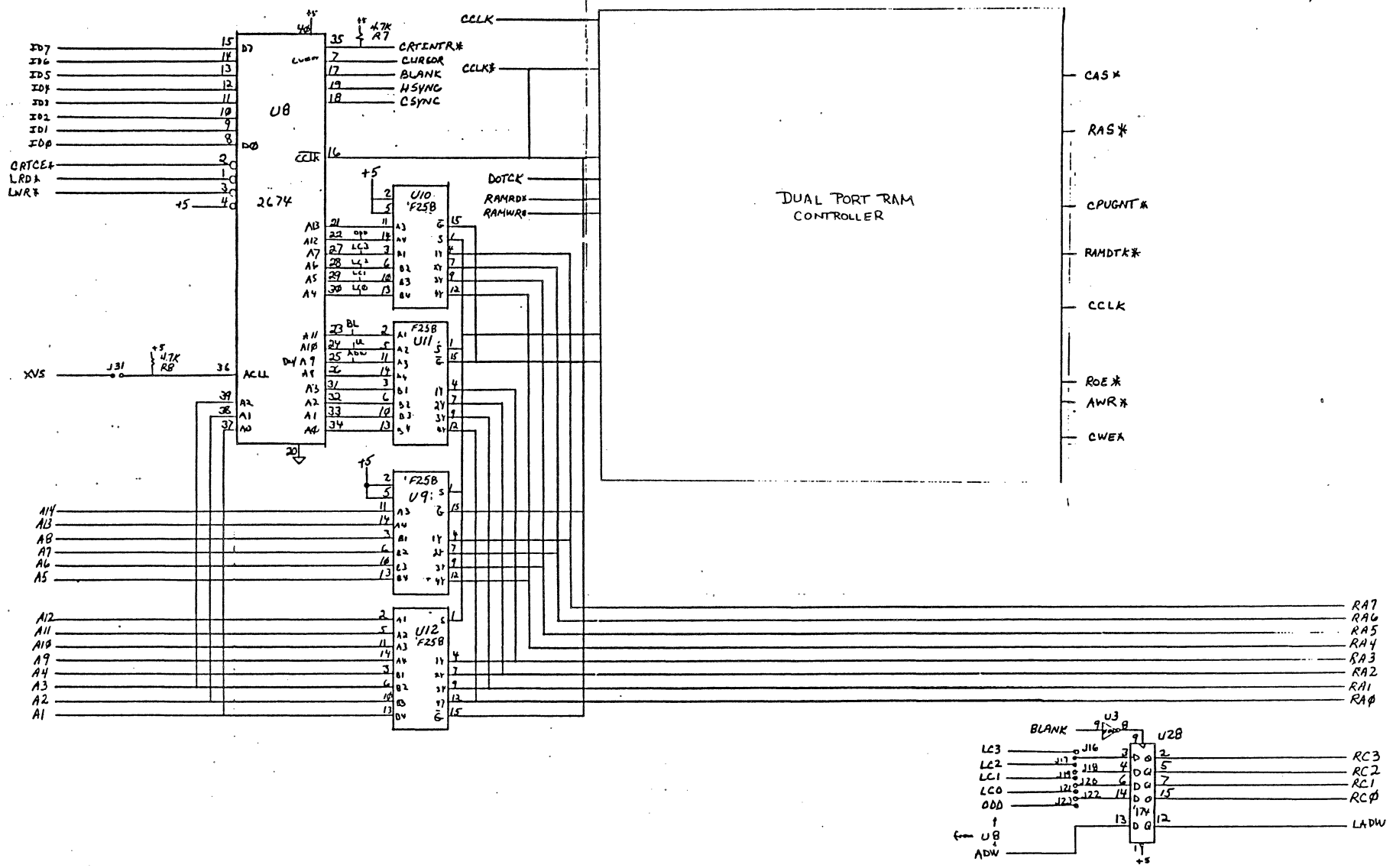
6.4 APPLICATION NOTES

The following data sheets and application notes are included at the courtesy of Signetics, Inc. and Motorola, Inc.

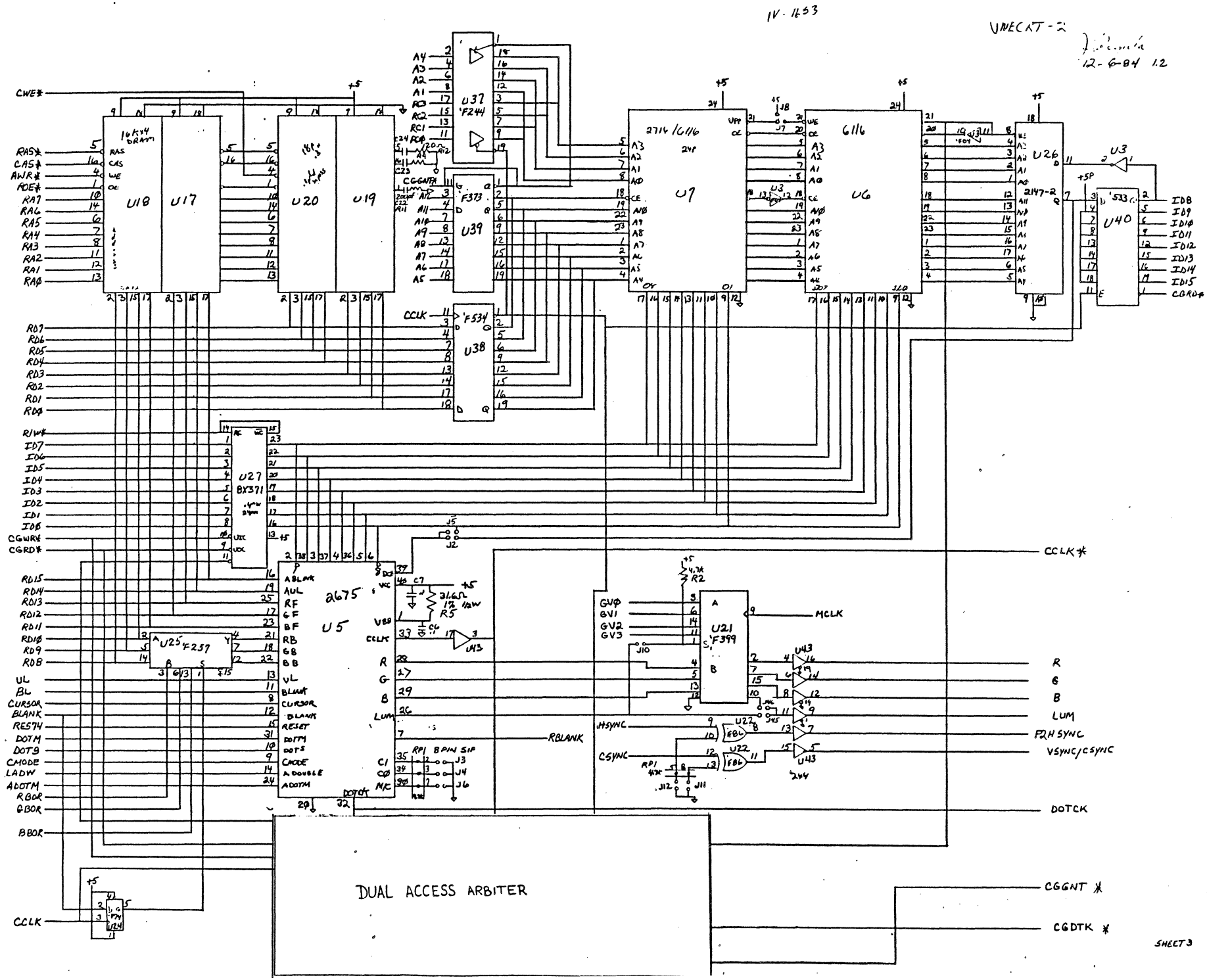
Signetics:
- 2674
- "SMOOTH SCROLL WITH THE 2674"
- 2675
- 2681

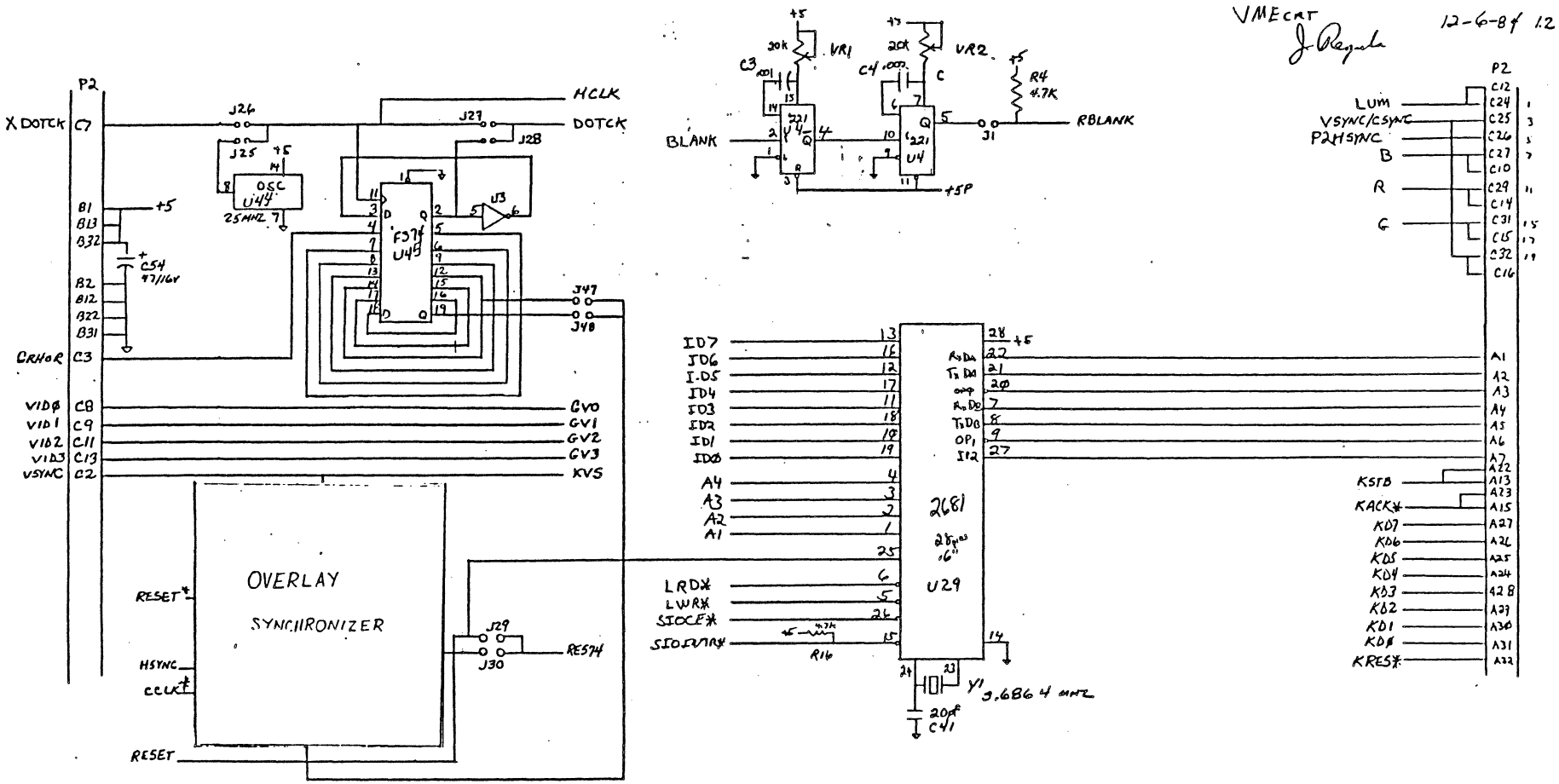
MOTOROLA:
- 68230

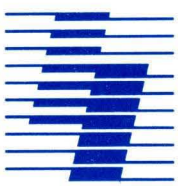

SHEET 1
U41
U42
U54
U55
U53
U48
U47
U31
U1
68230L8
CRTCE*
CGWR*
CGRD*
RAMRD*
RAMWR*
SIOCE*
R/W*
CRTINTR*
SIOINTR*
RESET
RAMDTR*
CGDTR*
16MHZ
SYSCLK
KACK*
KSTB
KRES*
RESET*
KD7
KD6
KD5
KD4
KD3
KD2
KD1
KD0
DO0-1605-12-6-84 1.2


U8
2674
U10 'F258
U11 F258
U9 'F258
U12 'F258
DUAL PORT RAM CONTROLLER
CCLK
CCLK*
DOTCK
RAMADR
RAMWR
CAS*
RAS*
CPUGNT*
RAMDT*
CCLK
ROE*
AWR*
CWE*
CRTINTR*
CURSOR
BLANK
HSYNC
CSYNC
CRTCE*
LRD*
LWR*
XVS
RA7
RA6
RA5
RA4
RA3
RA2
RA1
RA0
U3
U28
RC3
RC2
RC1
RC0
LADW

SHEET 2


IV-1653
UNECAT-2
12-6-84 1.2
CWE*
RAS*
CAS*
AWR*
RDE*
RA7
RA6
RA5
RA4
RA3
RA2
RA1
RA0
U18
U17
U20
U19
U37 'F244
U39 'F373
U38 'F534
CCLK
CGGNT*
U7 2716/6116 2KP
U6 6116
U26 2147-2
U40 '533
IDB
ID9
ID10
ID11
ID12
ID13
ID14
ID15
CGRD*
RD7
RD6
RD5
RD4
RD3
RD2
RD1
RD0
R/W*
ID7
ID6
ID5
ID4
ID3
ID2
ID1
ID0
CGWR*
U27 8X371
J5
J2
CCLK*
RD15
RD14
RD13
RD12
RD11
RD10
RD9
RD8
U25 'F257
U5 2675
C7
3.16Ω 1% 1/2W R5
U43
R
G
B
LUM
FRH SYNC
VSYNC/CSYNC
RBLANK
DOTCK
UL
BL
CURSOR
BLANK
RES74
DOTM
DOT9
CMODE
LADW
ADOTM
RBOR
GBOR
BBOR
CCLK
GV0
GV1
GV2
GV3
J10
U21 'F399
MCLK
4.7K R2
HSYNC
CSYNC
U22 F86
RP1 4.7K
J12
J11
RP1 B PIN SIP
J3
J4
J6
DUAL ACCESS ARBITER
CGGNT *
CGDTK *
SHEET 3


VMECRT
J. Reynolds
12-6-84 12
P2
X DOTCK
C7
HCLK
DOTCK
J26
J25
J29
J28
OSC
U49
25MHZ
+5
B1
B13
B32
B2
B12
B22
B31
C54
47/16V
CRHOR
C3
F374
U48
U3
J47
J48
VID0
VID1
VID2
VID3
VSYNC
C8
C9
C11
C13
C2
GV0
GV1
GV2
GV3
KVS
OVERLAY
SYNCHRONIZER
RESET
HSYNC
CCLK
RESET
J29
J30
RESET
BLANK
VR1
30K
C3
.001
VR2
20K
C4
.000
C
R4
4.7K
RBLANK
J1
+5P
ID7
ID6
ID5
ID4
ID3
ID2
ID1
ID0
A4
A3
A2
A1
LRD*
LWR*
SIOCE*
SIOCLK*
R16
2681
U29
Y1
3.6864 MHZ
20pF
C41
LUM
VSYNC/CSYNC
P2HSYNC
B
R
G
KSTB
KACK*
KD7
KD6
KD5
KD4
KD3
KD2
KD1
KD0
KRES*



IRONICS Incorporated

Computer Systems Division
798 Cascadilla Street
Ithaca, New York 14850
607-277-4060 Telex: 705-742