Omni-Programmer Technical Manual

Release 1.0

March 1, 1984

Varix Corporation
122 Spanish Village #608
Dallas, Texas 75248
(214)620-0925

COPYRIGHT

TRADEMARKS

Table of Contents

	Page
Overview:	
o How to use this manual	1
o Programmable Devices by Catagory	3
Theory of Operations:	
o Hardware	4
o Software	5
Specifications:	
o Hardware Interface Specifications	6
o Software Interface Specifications	22
o Electrical Specifications	27
o Mechanical Specifications	29
User Modifications:	
o User Specified Programming Algorithms	30
Omni-Programmer Index File	32
Method I	34
Method II	52
Assembly Language Assist Routines	
For input to Stream	64
For input to Mapword	65
Assemble Word	67
o Porting Procedure	68
Trouble Shooting:	
o Procedure	69
o Problem Reporting	71
o Problem Report Form	
o Customer Response Form	
o Schematics	73
Logic Board Schematics	
Socket Board Schematics	
o User Diagnostics	
BW Instructions with IBM PC	74
BW Instructions with Kaypro	75

Overview

The Varix Corporation Omni-Programmer product line is a combination of hardware and software which is designed to generate the necessary control voltages to place data into programmable integrated circuits. The hardware manipulates the currents and voltages for the creation of programming waveforms and the software controls the sequence of hardware events required for chip programming. Since the hardware has no chip dependent circuitry in it, it remains unchanged while software configures the socket pins appropriately for the targeted programmable device. The software runs on a microcomputer and issues commands to the Omni-Programmer hardware via a high speed parallel interface. During the programming process, the microcomputer must be 100% dedicated to the programming function since much of the critical algorithm timing is controlled by instruction timing. When not involved in programming, the attached computer is available for data down load and up load to larger computers, data printing, or general development activities.

There are two major categories of programmable integrated circuits -- read only memory (ROM) and programmable logic devices (PLD). ROM's represent the larger group and use one of two data storage techniques: fuse link, in which a tiny fuse inside the chip is destroyed when data is inserted and left intact otherwise; and charge storage, in which a small capacitor on the chip is either charged or discharged to represent the binary state of each bit. There are four technologies generally used to manufacture programmable integrated circuits. They are TTL bipolar, NMOS, CMOS, and ECL.

TTL bipolar is the oldest of the technologies and uses fuse link techniques for storing data. TTl bipolar's only advantage is high speed. The chips using this technology use a lot of power and are fairly expensive compared to MOS. NMOS is the highest density technology, is fairly low power, and is inexpensive. Charge storage technology is always used. Up to 512,000 bits can be put on a single chip with today's technology. EPROMs and EEPROMs are in this group. The greatest disadvantage of NMOS is its low speed. CMOS is similar to NMOS but is extremely low power. It uses either charge storage or fuse link technology tends to be fairly expensive and ranges from slow to fairly fast. The most rapid new developments are in this area. ECL is an infrequently used technology reserved for those applications in which blazing speed is the only consideration. They are fuse link devices which eat power, store only small amounts of data, and are exorbitantly expensive. Still, they offer a 10 times speed improvement over other device types.

While the major use of ROM devices is the permanent storage of data, PLD's are oriented toward the control of data flow. The data storage elements in a PLD are typically configured in a pattern to replace logic devices in a circuit board. Consequently, PLD's are always fuse link devices and are generally TTL bipolar technology. There is ongoing development of PLD's in CMOS, ECL, and charge storage (EEPLA) devices, but none are currently in production.

Due to the large number of programmable devices available on the market Varix has divided them into categories delineated by function and technology type. Each of these categories is called a license device group (LDG). There are currently 10 LDG's in the Omni product line.

The Omni-Programmer is designed to program the entire universe of programmable devices through software programs. These programs are ordinarily supplied by Varix Corporation, but they can be written by the user if so desired. Guidance on user written programming algorithms is in the user modification section of this manual. There is an occasional device whose required voltages are outside the specifications of the Omni-Programmer which are detailed in section 3, in which case the device cannot be programmed. Varix is committed to providing the software necessary for almost all standard production devices in the most timely manner possible.

This manual provides the information necessary to modify that software, port Omni-Programmer to a new microcomputer, and trouble shoot the Omni software or hardware if problems should arise.

Overview con't.

Programmable Devices By Catagory:

Varix Software	Description	Technology	Storage Mechanism
LDG01	EPROMs (UV erasable programmable read only memory) most common, highest density, low speed	NMOS	Charge Storage
LDG02	TTL Bipolar PROMs High speed, medium density	TTL Bipolar	Fuse Link
LDG03	TTL Bipolary PROMs (Current programmed) same as LDG02 except current is used for programming the fuses with data	TTL Bipolar	Fuse Link
LDG04	Obsclete Devices These devices have been discontinued by the manufacturer for new designs	All Types	All Types
LDG05	Microprocessors These are really standard Microprocessor chips with some EPROM or EEPROM on board	NMOS CMOS	Charge Storage
LDG06	PALs (programmable array logic) a small scale, limited PLD	TTL Bipolar	Fuse Link
LDG07	PLAs (programmable logic arrays) General PLDs plus a few special application units like programmable multiplexors	TTL Bipolar	Fuse Link
LDG08	CMOS	CMOS	Charge Storage Fuse Link
LDG09	EAROMs (electrically alterable read only memory) special byte by byte alterable devices	NMOS	Charge Storage
LDG10	EEPROMs (electrically erasable ROM) can be erased and reprogrammed electrically	NMOS CMOS	Charge Storage

Theory of Operations

Hardware

The Omni-Programmer is a dumb box whose purpose is to provide controlled voltages and currents to the pins of an integrated circuit under test. All of the level settings, ramp rates, pulse widths, and durations of voltages are controlled in real time by the software in the attached microcomputer.

There are three programmable voltage supplies in the Omni whose output voltage can be selectively set by software. Their current limit can also be controlled. There are 256 voltage steps and 256 current steps for each of the three supplies. The supplies are referred to as v1, v2, and v3. The current being delivered by each supply can be measured in steps of 256. An 8 bit D/A converter is used to adjust the voltage + current limit on each supply. Current measurement is accomplished by comparing a variable threshhold voltage against the voltage drop of a series resistor in each power supply circuit. In addition to the three voltages, any pin can be set to TTL high or TTL low.

The three power supply voltages are multiplexed to various pins on the programming sockets and are selectively enabled with transistor switches. Each switch is controlled by a bit from an octal latch on the Omni logic board. Voltage control bits on the Omni logic board are arranged as an array of sixteen 8 bit output ports and six 8 bit input ports. Voltage and current D/As, LEDs, and control bits are contained in the balance of the I/O ports. In all, 32 I/O ports are assigned to an SP0300 Omni-Programmer, 20 for output, 8 for input.

The interface between the Omni-Programmer and its attached microcomputer is designed to allow several configurations so that it can be readily adapted through cable wiring to most easily match the microcomputer. Basically, there is an 8 bit address, 8 bits of data out, 8 bits of data in, control lines, and ground reference in the cable. These buses can be run separately, requiring 16 bits of output, 8 bits of input and only one strobe to indicate when the next data transfer should be made. The input data in this mode will asynchronously represent the data from the input port selected by the address bus. At the other extreme, the entire bus may be multiplexed on 8 data lines. In this case, an address strobe is used to indicate valid address on the bus, a data strobe is used to indicate valid data, and an input strobe is used to turn the bus around for reading. Both the address and data are latched by the Omni in this configuration. NOTE: due to the high data transfer rates required for real time manipulations of voltages on pins, it is impractical to use any kind of serial interface scheme. The required transfer rate exceeds 1 Mbaud.

Software

The software for the Omni-Programmer runs exclusively on the attached microcomputer. With the exception of critical timing code, it is written in the high level language "C" to promote portability from one microcomputer to another. The software is divided into three sections:

> One: the hardware interface is 1000 bytes of assembly language code which communicates between the microcomputer/Omni hardware and the software. Only this section is modified when porting Omni from one microcomputer to another so long as both are using the same microprocessor and operating system.
>
> Two, the programming algorithm is 4000 bytes of "C" and assembler which creates the wavefroms and controls the programming algorithm for the device being programmed.
>
> Three, the user interface is 52,000 bytes of "C" code which generates the displays, accepts and executes user commands, and handles all system functions.

Both the hardware interface and programming algorithm software can be modified by the user. Techniques for doing this are described in the user modification section. Communications between the various modules is achieved through fixed jump vectors and status words in each module. This allows each module to operate entirely on its own and offers no restrictions on what language the module is written in so long as it adheres to the fixed interface specifications.

In addition to the actual programming code, there are a group of special utilities which enhance the programming environment. Omni comm supports communications between the microcomputer and a host machine so that chip data can be received from a remote source. Monolithic Memory's PALASM allows the creation of programmable logic device patterns from Boolean logic equations. Statistics prints historical information on Omni programming activity. BW offers diagnostic testing on the hardware.

One of the major advantages of the Varix Omni-Programmer is the flexibility it gives the user in writing software -- either as utilities, or as part of the programming environment, to manage the programming process for maximum efficiency.

Hardware Interface Specifications

The I/O port is sent via the address bus from the microcomputer. The data on the data bus is used to select bits within the I/O port. The output ports are divided into primary, secondary, and auxiliary ports. The primary ports are assigned one bit per socket pin where that bit indicates a reset condition (zero volts) when low ("0") and some positive voltage when high ("1"). The secondary ports control which voltage is active when a primary port bit is high. In the absence of a secondary port for a particular pin, the high state creates a logic level "1" on the socket through a 10K pull-up resistor to 5v. This 10K resistor also provides the pull-up for open collector output devices. The state in which the 10K resistor to 5v is active is called the "disable" condition. Any output pin from a chip should be "disabled" prior to reading data for it.

The secondary ports assign either one or two bits to a socket pin depending on how many voltages are available for that pin. In the case of a single secondary bit, a "0" indicates disable condition and "1" selects a voltage (typically v2). In the two bit decode case, 00 is disable, 01 is v1, 10 is v2, and 11 is either v3 or clk. The clk is a 3MHz driven square wave for use on microprocessor chips.

The auxiliary ports are D/A data, LED control, and calibration logic control. The read ports read the state (0 or 1) of each pin as compared to the threshold voltage. Consequently, the actual voltage on a pin can be determined with .118 volts by adjusting the threshold voltage until a transition occurs on the relevant pin.

Write Ports (primary)

I/O Port	Bit	Logic Level 0	Logic Level 1	48 Pin	28 Pin	24 Pin
00	0	Reset	*	21	11	14
	1	Reset	*	22	12	13
	2	Reset	*	23	13	15
	3	Reset	*	25	15	11
	4	Reset	*	26	16	17
	5	Reset	*	27	17	19
	6	Reset	*	28	18	16
	7	Reset	*	29	19	3
01	0	Reset	*	20	10	10
	1	Reset	*	19	9	9
	2	Reset	*	18	8	18
	3	Reset	*	17	7	7
	4	Reset	*	16	6	6
	5	Reset	*	15	5	5
	6	Reset	*	14	4	4
	7	Reset	*	13	24	2
02	0	Reset	*	35		
	1	Reset	*	34	28	
	2	Reset	*	31	21	23
	3	Reset	*	33	27	1
	4	Reset	*	12	23	
	5	Reset	*	36	25	22
	6	Reset	*	37		
	7	Reset	*	30	20	8
03	0	Reset	Disable	5		
	1	Reset	*	6		
	2	Reset	*	7		
	3	Reset	Disable	8		
	4	Reset	*	9		
	5	Reset	Disable	10		
	6	Reset	*	44	2	21
	7	NOT USED				

NOTE: * - pin controlled by secondary I/O port with corresponding pin # shown on the following pages.

Write Ports (primary)

I/O Port	Bit	Logic Level 0	Logic Level 1	48 Pin	28 Pin	24 Pin
04	0	Reset	Disable	43		
	1	Reset	Disable	42		
	2	Reset	Disable	41		
	3	Reset	Disable	40		
	4	Reset	Disable	39		
	5	Reset	*	11	22	20
	6	Reset	*	38	1	
	7	Reset	*	32	26	
05	0	Reset	Disable	45		
	1	Reset	Disable	46		
	2	Reset	Disable	47		
	3	Reset	*	48	3	24
	4	Reset	Disable	1		
	5	Reset	Disable	2		
	6	Reset	Disable	3		
	7	Reset	Disable	4		
06	NOT USED					
07	NOT USED					

Write Ports (secondary)

I/O Port	Bit	Logic Level 0	Logic Level 1	48 Pin	28 Pin	24 Pin
08	0	Disable	clk	6		
	1	Disable	clk	7		
	2	Disable	v2	9		
	3	Disable	v2	35		
	4	Disable	v2	32	26	
	5	Disable	v2	37		
	6	Disable	v3	38	1	
	7	Disable	v3	48	3	24
09	0	Disable	v2	31	21	23
	1	Disable	v2	18	8	18
	2	Disable	v2	17	7	7
	3	Disable	v2	16	6	6
	4	Disable	v2	15	5	5
	5	Disable	v2	14	4	4
	6	Disable	v2	34	28	
	7	Disable	v2	12	23	

NOTE: If pin is "RESET", logic level here is don't care.

Write Ports (secondary)

I/O Port	Bit	Logic Level 00	Logic Level 01	Logic Level 10	Logic Level 11	48 Pin	28 Pin	24 Pin
OA	1, 0	Disable	v1	v2	NC	21	11	14
	3, 2	Disable	v1	v2	NC	22	12	13
	5, 4	Disable	v1	v2	CLK	23	13	15
	7, 6	Disable	v1	v2	v3	25	15	11
Ob	1, 0	Disable	v1	v2	NC	26	16	17
	3, 2	Disable	v1	v2	NC	27	17	19
	5, 4	Disable	v1	v2	NC	28	18	16
	7, 6	Disable	v1	v2	NC	29	19	3
OC	1, 0	Disable	v1 1.5	v2	v3	11	22	20
	3, 2	Disable	v1	v2	NC	20	10	10
	5, 4	Disable	v1	v2	NC	19	9	9
	7, 6	Disable	v1	v2	NC	30	20	8
OD	1, 0	Disable	v1	v2	-5v	33	27	1
	3, 2	Disable	NC	v2	v3	36	25	22
	5, 4	Disable	NC	v2	v3	44	2	21
	7, 6	Disable	v1	v2	NC	13	24	2

NOTE: If pin is reset, logic levels are don't care.

NC - No connection (effect is diable).

Write Ports

I/O Port	Bits	
OE	NOT USED	
OF	NOT USED	
10	0-7	v1 Voltage Set
11	0-7	v2 Voltage Set
12	0-7	v3 Voltage Set
13	0-7	v1 Current Limit Set
14	0-7	v2 Current Limit Set
15	0-7	Threshold Set
16	0	LED 48 Pin Socket ("D" = ON)
	1	LED 28 Pin Socket
	2	LED 24 Pin Socket
	3	LED "Connected"
	4	LED "Programming"
	5	LED "Error"
	6	"1" = v1 Current Load Enable
	7	"1" = v2 Current Load Enable

Read Ports

I/O Port	Bit	48 Pin	28 Pin	24 Pin
0	0	21	11	14
	1	22	12	13
	2	23	13	15
	3	25	15	11
	4	26	16	17
	5	27	17	19
	6	28	18	16
	7	29	19	3
1	0	20	10	10
	1	19	9	9
	2	18	8	8
	3	17	7	7
	4	16	6	6
	5	15	5	5
	6	14	4	4
	7	13	24	2
2	0	35		
	1	34	28	
	2	31	21	23
	3	33	27	1
	4	12	23	
	5	36	25	22
	6	37		
	7	30	20	8
3	0	5		
	1	6		
	2	7		
	3	8		
	4	9		
	5	10		
	6	44	2	21
	7	GND	GND	GND
4	0	43		
	1	42		
	2	41		
	3	40		
	4	39		
	5	11	22	20
	6	38	1	
	7	32	26	

Read Ports ccn't.

I/O Port	Bit	48 Pin	28 Pin	24 Pin
5	0	45		
	1	46		
	2	47		
	3	48	3	24
	4	1		
	5	2		
	6	3		
	7	4		
6	NOT USED			
7	NOT USED			
8	NOT USED			
9	NOT USED			
A	NOT USED			
B	NOT USED			
C	NOT USED			
D	0-7	Reads Prcgrammed Threshcld Set		
E	NOT USED			
F	0	v1 Current Sense		
	1	v2 Current Sense		
	2	v3 Current Sense		

Chip Map

The table on the following page shows the relationship between the pins on the 48 pin socket and the pins on the other two sockets. The voltages available on those pins are also shown.

The first four columns contain pin numbers for socket #1 (6/10"). The first column relates to a 48 pin device, the second column corresponds to what a 40 pin device orientation would be in the same socket. The next three columns are the assignments that are hardware set for the pins specified. All pin # parameters are specified relative to the 48 pin socket.

CHIPMAP

```
                     1                                    48   v3
                     2                                    47
                     3                                    46
                     4                                    45
                     5    1                          40   44        v2   v3
            clk      6    2                          39   43
            clk      7    3                          38   42
                     8    4                          37   41
            v2       9    5                          36   40
                     10   6                          35   39
v3    v2    v1/1.5   11   7    1                28   34   38   v3
            v2       12   8    2                27   33   37   v2
      v2    v1       13   9    3    1      24   26   32   36        v2   v3
            v2       14   10   4    2      23   25   31   35   v2
            v2       15   11   5    3      22   24   30   34   v2
            v2       16   12   6    4      21   23   29   33   v1   v2   -5
            v2       17   13   7    5      20   22   28   32   v2
            v2       18   14   8    6      19   21   27   31   v2
      v2    v1       19   15   9    7      18   20   26   30   v1   v2
      v2    v1       20   16   10   8      17   19   25   29   v1   v2
      v2    v1       21   17   11   9      16   18   24   28   v1   v2
      v2    v1       22   18   12   10     15   17   23   27   v1   v2
clk   v2    v1       23   19   13   11     14   16   22   26   v1   v2
            gnd      24   20   14   12     13   15   21   25   v1   v2   v3


            v3    38   1                                    28   34   v2
v3    v2          44   2    1                          26   27   33   v1    v2     -5
            v3    48   3    2    1                24   25   26   32   v2
            v2    14   4    3    2    1      22   23   24   25   36         v2    v3
            v2    15   5    4    3    2      21   22   23   24   13   v1    v2
            v2    16   6    5    4    3      20   21   22   23   12   v2
            v2    17   7    6    5    4      19   20   21   22   11   v1/1.5       v2   v3
            v2    18   8    7    6    5      18   19   20   21   31   v2
      v2    v1    19   9    8    7    6      17   18   19   20   30   v1    v2
      v2    v1    20   10   9    8    7      16   17   18   19   29   v1    v2
      v2    v1    21   11   10   9    8      15   16   17   18   28   v1    v2
      v2    v1    22   12   11   10   9      14   15   16   17   27   v1    v2
clk   v2    v1    23   13   12   11   10     13   14   15   16   26   v1    v2
            gnd   24   14   13   12   11     12   13   14   15   25   v1    v2     v3


            v2    33   1                                    24   48   v3
      v2    v1    13   2                                    23   31   v2
      v2    v1    29   3    1                          20   22   36         v2    v3
            v2    14   4    2    1                18   19   21   44         v2    v3
            v2    15   5    3    2    1      16   17   18   20   11   v1/1.5       v2   v3
            v2    16   6    4    3    2      15   16   17   19   27   v1    v2
            v2    17   7    5    4    3      14   15   16   18   18   v2
      v2    v1    30   8    6    5    4      13   14   15   17   26   v1    v2
      v2    v1    19   9    7    6    5      12   13   14   16   28   v1    v2
      v2    v1    20   10   8    7    6      11   12   13   15   23   v1    v2     clk
v3    v2    v1    25   11   9    8    7      10   11   12   14   21   v1    v2
            gnd   24   12   10   9    8       9   10   11   13   22   v1    v2
```

CHIPMAP

48-pin	40-pin	28-pin	24-pin	Pri-mary	1st Alt	2nd Alt	28-pin	24-pin	COMMENTS
1									
2									
3									
4									
5	1								
6	2			CLK					3-MHZ
7	3			CLK					3-MHZ
8	4								
9	5			V2					
10	6								
11	7	1		V1	V2	V3	22		
12	8	2		V2			23		
13	9	3	1	V1	V2		24	2	
14	10	4	2	V2			4	4	
15	11	5	3	V2			5	5	
16	12	6	4	V2			6	6	
17	13	7	5	V2			7	7	
18	14	8	6	V2			8	18	
19	15	9	7	V1	V2		9	9	
20	16	10	8	V1	V2		10	10	
21	17	11	9	V1	V2		11	14	
22	18	12	10	V1	V2		12	13	
23	19	13	11	V1	V2	CLK	13	15	3-MHZ
24	20	14	12	GND			14	12	
25	21	15	13	V1	V2	V3	15	11	
26	22	16	14	V1	V2		16	17	
27	23	17	15	V1	V2		17	19	
28	24	18	16	V1	V2		18	16	
29	25	19	17	V1	V2		19	3	
30	26	20	18	V1	V2		20	8	
31	27	21	19	V2			21	23	
32	28	22	20	V2			26		
33	29	23	21	V1	V2	-5V	27	1	
34	30	24	22	V2			28		
35	31	25	23	V2					
36	32	26	24		V2	V3	25	22	
37	33	27		V2					
38	34	28		V3			1		
39	35								
40	36								
41	37								
42	38								
43	39								
44	40				V2	V3	2	21	
45									
46									
47									
48				V3			3	24	

PINMAP

pin	reset	disable	set	alt1	alt2
1	port 5 bit 4 0	port 5 bit 4 1			
2	port 5 bit 5 0	port 5 bit 5 1			
3	port 5 bit 6 0	port 5 bit 6 1			
4	port 5 bit 7 0	port 5 bit 7 1			
5	port 3 bit 0 0	port 3 bit 0 1			
6	port 3 bit 1 0	port 3 bit 1 1 port 8 bit 0 0	port 3 bit 1 1 port 8 bit 0 1 clk		
7	port 3 bit 2 0	port 3 bit 2 1 port 8 bit 1 0	port 3 bit 2 1 port 8 bit 1 1 clk		
8	port 3 bit 3 0	port 3 bit 3 1			
9	port 3 bit 4 0	port 3 bit 4 1 port 8 bit 2 0	port 3 bit 4 1 port 8 bit 2 1 v2		
10	port 3 bit 5 0	port 3 bit 5 1			
11	port 4 bit 5 0	port 4 bit 5 1 port C bit 1 0 port C bit 0 0	port 4 bit 5 1 port C bit 1 0 port C bit 0 1 v1/1.5	port 4 bit 5 1 port C bit 1 1 port C bit 0 0 v2	port 4 bit 5 1 port C bit 1 1 port C bit 0 1 v3
12	port 2 bit 4 0	port 2 bit 4 1 port 9 bit 7 0	port 2 bit 4 1 port 9 bit 7 1 v2		
13	port 1 bit 7 0	port 1 bit 7 1 port D bit 7 0 port D bit 6 0	port 1 bit 7 1 port D bit 7 0 port D bit 6 1 v1	port 1 bit 7 1 port D bit 7 1 port D bit 6 0 v2	

pin	reset	disable	set	alt1	alt2
14	port 1 bit 6 0	port 1 bit 6 1 port 9 bit 5 0	port 1 bit 6 1 port 9 bit 5 1 v2		
15	port 1 bit 5 0	port 1 bit 5 1 port 9 bit 4 0	port 1 bit 5 1 port 9 bit 4 1 v2		
16	port 1 bit 4 0	port 1 bit 4 1 port 9 bit 3 0	port 1 bit 4 1 port 9 bit 3 1 v2		
17	port 1 bit 3 0	port 1 bit 3 1 port 9 bit 2 0	port 1 bit 3 1 port 9 bit 2 1 v2		
18	port 1 bit 2 0	port 1 bit 2 1 port 9 bit 1 0	port 1 bit 2 1 port 9 bit 1 1 v2		
19	port 1 bit 1 0	port 1 bit 1 1 port C bit 5 0 port C bit 4 0	port 1 bit 1 1 port C bit 5 0 port C bit 4 1 v1	port 1 bit 1 1 port C bit 5 1 port C bit 4 0 v2	
20	port 1 bit 0 0	port 1 bit 0 1 port C bit 3 0 port C bit 2 0	port 1 bit 0 1 port C bit 3 0 port C bit 2 1 v1	port 1 bit 0 1 port C bit 3 1 port C bit 2 0 v2	
21	port 0 bit 0 0	port 0 bit 0 1 port A bit 1 0 port A bit 0 0	port 0 bit 0 1 port A bit 1 0 port A bit 0 1 v1	port 0 bit 0 1 port A bit 1 1 port A bit 0 0 v2	
22	port 0 bit 1 0	port 0 bit 1 1 port A bit 3 0 port A bit 2 0	port 0 bit 1 1 port A bit 3 0 port A bit 2 1 v1	port 0 bit 1 1 port A bit 3 1 port A bit 2 0 v2	
23	port 0 bit 2 0	port 0 bit 2 1 port A bit 5 0 port A bit 4 0	port 0 bit 2 1 port A bit 5 0 port A bit 4 1 v1	port 0 bit 2 1 port A bit 5 1 port A bit 4 0 v2	port 0 bit 2 1 port A bit 5 1 port A bit 4 1 clk

pin	reset	disable	set	alt1	alt2
24	port 3 bit 7 X ground				
25	port 0 bit 3 0	port 0 bit 3 1 port A bit 7 0 port A bit 6 0	port 0 bit 3 1 port A bit 7 0 port A bit 6 1 v1	port 0 bit 3 1 port A bit 7 1 port A bit 6 0 v2	port 0 bit 3 1 port A bit 7 1 port A bit 6 1 v3
26	port 0 bit 4 0	port 0 bit 4 1 port B bit 1 0 port B bit 0 0	port 0 bit 4 1 port B bit 1 0 port B bit 0 1 v1	port 0 bit 4 1 port B bit 1 1 port B bit 0 0 v2	
27	port 0 bit 5 0	port 0 bit 5 1 port B bit 3 0 port B bit 2 0	port 0 bit 5 1 port B bit 3 0 port B bit 2 1 v1	port 0 bit 5 1 port B bit 3 1 port B bit 2 0 v2	
28	port 0 bit 6 0	port 0 bit 6 1 port B bit 5 0 port B bit 4 0	port 0 bit 6 1 port B bit 5 0 port B bit 4 1 v1	port 0 bit 6 1 port B bit 5 1 port B bit 4 0 v2	
29	port 0 bit 7 0	port 0 bit 7 1 port B bit 7 0 port B bit 6 0	port 0 bit 7 1 port B bit 7 0 port B bit 6 1 v1	port 0 bit 7 1 port B bit 7 1 port B bit 6 0 v2	
30	port 2 bit 7 0	port 2 bit 7 1 port C bit 7 0 port C bit 6 0	port 2 bit 7 1 port C bit 7 0 port C bit 6 1 v1	port 2 bit 7 1 port C bit 7 1 port C bit 6 0 v2	
31	port 2 bit 2 0	port 2 bit 2 1 port 9 bit 0 0	port 2 bit 2 1 port 9 bit 0 1 v2		
32	port 4 bit 7 0	port 4 bit 7 1 port 8 bit 4 0	port 4 bit 7 1 port 8 bit 4 1 v2		

pin	reset	disable	set	alt1	alt2
33	port 2 bit 3 0	port 2 bit 3 1 port D bit 1 0 port D bit 0 0	port 2 bit 3 1 port D bit 1 0 port D bit 0 1 v1	port 2 bit 3 1 port D bit 1 1 port D bit 0 0 v2	port 2 bit 3 1 port D bit 1 1 port D bit 0 1 -5
34	port 2 bit 1 0	port 2 bit 1 1 port 9 bit 6 0	port 2 bit 1 1 port 9 bit 6 1 v2		
35	port 2 bit 0 0	port 2 bit 0 1 port 8 bit 3 0	port 2 bit 0 1 port 8 bit 3 1 v2		
36	port 2 bit 5 0	port 2 bit 5 1 port D bit 3 0 port D bit 2 0		port 2 bit 5 1 port D bit 3 1 port D bit 2 0 v2	port 2 bit 5 1 port D bit 3 1 port D bit 2 1 v3
37	port 2 bit 6 0	port 2 bit 6 1 port 8 bit 5 0	port 2 bit 6 1 port 8 bit 5 1 v2		
38	port 4 bit 6 0	port 4 bit 6 1 port 8 bit 6 0	port 4 bit 6 1 port 8 bit 6 1 v3		
39	port 4 bit 4 0	port 4 bit 4 1			
40	port 4 bit 3 0	port 4 bit 3 1			
41	port 4 bit 2 0	port 4 bit 2 1			
42	port 4 bit 1 0	port 4 bit 1 1			
43	port 4 bit 0 0	port 4 bit 0 1			
44	port 3 bit 6 0	port 3 bit 6 1 port D bit 5 0 port D bit 4 0		port 3 bit 6 1 port D bit 5 1 port D bit 4 0 v2	port 3 bit 6 1 port D bit 5 1 port D bit 4 1 v3

pin	reset	disable	set	alt1	alt2
45	port 5 bit 0 0	port 5 bit 0 1			
46	port 5 bit 1 0	port 5 bit 1 1			
47	port 5 bit 2 0	port 5 bit 2 1			
48	port 5 bit 3 0	port 5 bit 3 1 port 8 bit 7 0	port 5 bit 3 1 port 8 bit 7 1 v3		

Specifications con't.

Hardware Interface Specifications

Interface

	min	max	units
Strobe pulse widths (address,data)	100		nanoseconds
Data in stable	.3	1	microseconds
Data setup time	100		nanoseconds
Data hold time	10		nanoseconds
Delay from data into output strobe	100		nanoseconds

Software Interface Specification

Under the OMNI operating system, memory is allocated as follows:

0103H thru 0502H	Hardware Control Code (HCC)
0503H thru 1602H	Programming Algorithm (PA)/ User Defined PA (UDP)
1603H thru FFFFH	OMNI Operating System (OS)

The Programming Algorithm is the software that is normally loaded to program or read a desired device such as a 2716 EPROM. It consists of a series of instructions that jump to various address pointers in the HCC which in turn carry out hardware set-up and execution operations like "set pin 11 to voltage 1". These addresses are called Function Pointers. They are described in the following paragraphs using this format: Address in hex, Function name, functional description, and where applicable, range and typical default condition for a 2716 EPROM. When a jump to a function is performed the stack must contain appropriate information for the function: the proper sequence would be first the function pointer address, then the function parameter(s) (abbreviated "param #").

0103H resall

Reset all pins to initializationn state. This function causes all pins to be reset.

0106H disall

Disable all pins; with pins in this state they can neither be written to nor read.

0109H respin

Resets the pin number in Param 1: range of pins 1-48. Reset sets the pin to 0 volts.

010CH dispin

Disables the pin number in Param 1; range of pins 1-48. Disabling a pin sets the pin voltage to 5 volts through a 10 Kohm pullup resistor.

010FH setpin

This function sets the pin number in Param 1 to its primary voltage or clock assignment; range of pins 1-48. Pin assignments can be found in the CHIPMAP found at the end of this section.

0112H alt1pin

This function sets the pin number in Param 1 to its first alternate voltage or clock assignment; range of pins 1-48 (not all pins have alternate conditions). Pin assignments can be found in the CHIPMAP.

0115H alt2pin

This function sets the pin number in Param 1 to its second alternate voltage or clock assignment; range of pins 1-48 (not all pins have alternate conditions). Pin assignments can be found in the CHIPMAP.

0127H rdpin

Reads the pin number in Param 1; range of pins 1-48. A read must be preceeded by a disable to the disired pin or the read will return what the pin has been set to.

012AH setv1

This function sets the primary voltage, referenced above, to the voltage determined by : .118 volts multiplied by (X) the value in Param 1; range of Param 1 = 0 to 255, range of voltage value=0 to 28 volts. Default value for a 2716 is Param 1 = 226 (27 volts); this value is usually not changed while programming the 2716.

012DH setv2

This function works just like setv1 does. Its value may change many times while programming a 2716 depending on the other functions it is being used with. It is the first alternate voltage.

0130H setv3

This function works just like setv1 and setv2 do. It is used like setv2 while programming but may not be changed as much because it is nct available on many pins (see CHIPMAP). It is the second alternate voltage. When using the alt2pin function on pin 33 this voltage sets pin 33 to -5 volts for use with the AMD part number 27S27.

0133H setcv1

This function sets the current limit for the primary voltage, v1, to the current determined by .004 amperes X the value in Param l; range of Param 1 = 0-255; range of current =0-1000 milliamperes. The default condition for the 2716 is Param 1=255.

0136H Setv2

This function works just like setcv1 for the first alternate voltage but it limits the current to 255 milliamperes and each increment has a value of .001 amperes; range of Param = 0 - 255; range of current = 0 - 255 milliamperes. The default condition for a 2716 is Param 1 = 255.

0139H setthr

This function sets the threshold voltage which is used while reading a pin to determine a high or low state. It's value is set just like setv1 where each increment represents .118 volts; range of Param 1 = 0 - 255; range of voltage = 0 - 28 volts.

0145H connect

This function is used only to see if the OMNI Programming Unit is attached to the Kaypro or other controlling processor. When this function is called a routine is started that either returns an error message or returns to the control program if no connect problems are detected.

014BH delay

This function creates a delay equal to the value of Param 1 X 100 microseconds. Range of Param 1 is 0-255; range of delay is 0 to 25500 microseconds. This function is used to control the amount of time a programming voltage is applied to a particular pin.

The code that becomes the UDP replaces the standard Programming Algorithm and therefore becomes the interface between the Operating System and the device for which the UDP is being written. Thus, some precautions are necessary to ensure no harm is done to the device and that the desired functions can be performed by the user. Within the section of memory known as the UDP, there are some pre-assigned locations that must be left untouched for the operating system to access them properly. The UDP can be looked at as having three sections as follows:

0503H thru 0512H	Setup Conditions
0513H thru 051FH	Device Type Definition
0520H thru 0520H	User Written Program Algorithm

The setup conditions are spaces in memory that set up the device for initialization, reading, writing, error reporting and configuration as follows:

```
0503H devsetup (chip index)
0506H devread  (chip start address, memory start address,
                  count)
0509H devwrite (chip start address, memory start address,
                  count)
050CH error condition code reporting
050EH error condition address
```

The jumps to these addresses are done by the OS; they are filled by the OS from inputs to prompt the user answers on initialization. The parameters will be loaded onto the stack in the order listed (param1, param2, param3).

The next three bytes are also filled by the OS.

0510H	number of pinns on device (24 for 2714)	
0511H	socket used for this device (1=6/10", 2=4/10", 3=3/10")	
0512H	device type (1=PROM, 2=PLA, 3=PGA, 4=PAL)	

The section that begins with 0513H varies in length and interpretation by device type. For PROMs and addresses contain the following information:

0513H	promlen	Max number of words to be addressed (for 2716-2048)
0515H	promwid	Word length (2716 word length = 8 bits)
0517H	promblk	Initial state of bits (OFFH for 2716)

For FPLAs the addresses contain the following information:

0513H	plalenn	Number of P-terms possible
0515H	plaiwid	Number of input bits possible
0516H	placwid	Number of output bits possible
0517H	plabias	Blank state of output bias
0518H	plaand	Blank state of AND matrix
0519H	placr	Blank state of OR matrix
051AH	plaistart	Starting input number
051BH	placstart	Starting output number

For PALs the addresses contain the following information:

0513H	palinp	Number of PAL inputs
0514H	palprod	Number of PAL products
0515H	palbln	Number of blown links
0516H	palilst	Pointer to bit array of inputs
0517H	palplst	Pointer to bit array of products
0518H	palblst	Coordinate array of blown links

For FPGA (Field Programmable Gate Arrays) the addresses contain the following information:

0513H	pgalen	PGA length
0515H	pgawid	PGA width
051x	tbd	to be defined

Electrical Specifications

Voltage Supply Circuits:

	min	max	units
Output voltage range	0	30	volts
Max output voltage	28	30	volts
Voltage steps	.118		volts/steps
Current limit (v1)	0	1.02	amps
(v2)	0	255	mA
(v3)		1.2	amps
Current limit steps (v1)		4	mA/step
(v2)		1	mA/step
Current step error		5%	
Slew rate (v1)	.2	.3	v/microsecond
(v2)	.4	.5	v/microsecond
(v3)	.2	.3	v/microsecond
Current sense (v1, v3)	0	1.2	amps
(v2)	0	250	mA

Socket pin switches:

	min	max	units
Current on v2 switches	0	.5	amp
Current on v1, v3 switches	0	3	amps
Saturation voltage change v1, v3	.2	.4	volts
(minimum current to max current) v2	.2	.3	volts
GND	.1	.3	volts
Current limit on gnd switch	6	15	mA
Slew rate	20	40	v/microsecond
-5v circuit current		-100	mA

Electrical Specifications con't.

Voltage Calibration Circuit

	min	max	units
Reference voltage	9.95	10.05	volts
Threshold voltage	0	30.1	volts
Threshold voltage error		.1%	
Threshold voltage steps		.118	volts/step

Mechanical Specifications

Overall Dimensions	16" W x 11" D x 5" H
Table space required	16" W x 14" D
Weight	8 lbs.
Operating Environment:	
Temperature	32 - 95° F 0 - 35° C
Humidity	20%-80% relative
Voltage	110VAC ± 10%
Power Requirements	50 watts

User Specified Programming Algorithms

The Omni-Programmer algorithm area is ordinarily used for loading the software algorithm used to program a particular group of chips. The Omni user interface and operating system uses the index file OMNI.IDX to find the program which should be loaded into the available algorithm space. The available space is only 4000 bytes so any user program must be no larger than that.

The user algorithm can be written in any language. It must adhere to the software interfaces specified in the previous section, be absolute binary Z80 machine code targeted for a bottom address of 503 (Hexadecimal), and reside in a file with the first byte of the file being the first byte of the program. If the filename is then placed appropriately in OMNI.IDX, the user algorithm will execute when called.

Varix programming algorithms are all written in the high level language "C". In order to improve the speed of programming, the innermost loops have often been replaced with assembly language assist routines. Functions like incrementing to the next chip address and assembling the data output into a single word are examples of these situations.

There are two basic approaches to writing device algorithms used by Varix Corporation. The first uses calls to the hardware interface routines described in the previous section to set each pin on the chip in the desired manner. EXAMPLE: Placing a 10 bit address on the address bus of a PROM requires a sequence of 10 calls to either DISPIN or RESPIN dependent on whether the pin is TTL high or TTL low. While the resulting code is very straight forward, it is also somewhat slow.

The second approach involves a combination of high level code and tables which describe the chip. In the above example with a 10 bit address, a table would be created which listed the chip pins making up the address bus. A single call to MAPWORD would then cause the data pattern to be placed on the appropriate pins.

Once the code and tables are created, the following system sequence should be used to create the programming algorithm.

```
A>cii -m  -X300  b:<filename>.c

A>m80  =b:<filename>.asm

A>l80  /p:503, hprom, b:<filename>, omnilib/s, libc/s, b:<filename>/n/x/e

A>zsid
  *i<filename>.hex
  *r-403
  *^c

  A> SAVE 16 B:<filename>.000
```

The link process (L80) will complete with a message in the following format:

```
[0 503 1439]
```

if the third number exceeds 1503, your programming algorithm is too large and must be reduced in size.

M80 and L80 are the Z80/8080 macro assembler and linker from Microsoft, Inc. CII is the 'Aztec C Compiler from Manx Software Systems, Inc. ZSID is the symbolic debugger from Digital Research, Inc.

Due to the stand alone nature of the programming algorithm module, it is possible to use any compiler, assembler, linker so long as the interface specifications are adhered to. A development package can be purchased from Varix which contains the development software customarily used by Varix along with command files.

Once the algorithm is written, it must be listed in the OMNI.IDX file in order to be invoked. The format of the file is shown on the following page. Typically, only the filename with a pound sign in front followed by the name (such as chip device number) with which you wish to invoke the file. The name must be preceded by a zero, which is ordinarily used as an index into a programming algorithm which may program several chips. EXAMPLE: To invoke the file CALIB.000 by typing CALIBRATE under the Omni-Programmer software add the following entries to OMNI.IDX:

```
#CALIB.000
0CALIBRATE          ; Special note: all invocation names must use upper case
                      letters.
```

The entire OMNI.IDX file looks like:

(see following page)

```
;               Omni-Programmer Index File              10/5/83
;
; Special characters are:
;       # Family name (general device type)
;       @ User interface filename
;       % Command table filename
;       : Personality module filename
;       ; Comment line (blank lines are comments also)
;
; A personality name is preceeded by the index (in hex) of its module
; in the personality file.

;------------------------------ Proms ------------------------------
#EPROM
@OMPROM.OVR
%OMPROM.TBL
:OMEPRM.000
02516 12532 42564 02716 22732 32732A 52764
:OMEPRM.001
027128 127256
:OMEPRM.002
068732-0
168732-1
268764 268766

;------------------- Gang-programmer Proms -------------------------
#GANG

@OMGANG.OVR
%OMGANG.TBL

:OMGPRM.000
02716 12732 22732A

:OMGPRM.001
02764 127128 227256
```

User Specified Programming Algorithms con't.

It is valuable to note that the OMNI.IDX file can be modified by using any editor to change the names of the chips which must be typed in for programming. Suppose, for example, that one wished to use internal 10 digit part numbers to identify chips. A simple change from 02716 12732 to 0300-0462-001 1300-0462-002 would cause the Intel 2716 and 2732 chips to be called up by their 10 digit number. You can even put both forms in if desired. If you put:

```
#TEPROM.001
02716    0300-0462-001    02KEPROM    0A
```

into your OMNI.IDX file, then entering any of the four names, 2716, 300-0462-001, 2KEPROM, or A, would result in programming a 2716.

USER SPECIFIED ALGORITHMS

User specified algorithms METHOD I:

(see following pages)

TEPROM1.C

METHOD I Example:

```
#include        "a:stdio.h"

#include        "a:econfig.h"

/*
        production version
*/

/*
        Setup the function pointers for the hardware interface routines
*/

#include        "a:omhdw.h"

/*
        set up variables to hold pin numbers
*/

int     ce;
int     oe;
int     vpp;
int     pgm;
int     pdpgm;
int     cepgm;
int     cevpp;
int     cs1;
int     cs2;
int     vcc;
int     vcca;

/*
        address pin maps
*/

int     maxaddr,                /* maximum address pin number   */

                                /* 2516/2716 address pins       */
char    amap1[11] = {20, 19, 18, 17, 16, 15, 14, 13, 35, 34, 31};

                                /* 2532 address pins            */
char    amap2[12] = {20, 19, 18, 17, 16, 15, 14, 13, 35, 34, 31, 30};

                                /* 2732/2732a address pins      */
char    amap3[12] = {20, 19, 18, 17, 16, 15, 14, 13, 35, 34, 31, 33};

                                /* 2564 address pins            */
char    amap4[13] = {20, 19, 18, 17, 16, 15, 14, 13, 35, 34, 31, 30, 33};

                                /* 2764 address pins            */
char    amap5[13] = {20, 19, 18, 17, 16, 15, 14, 13, 35, 34, 31, 33, 12};

char    *adrtab;

/*
        define voltages needed by the eproms
```

```
*/

#define VCC     46              /*  5.52 volts */
#define VIH     46              /*  5.52 volts */
#define VPP1    211             /* 25.32 volts */
#define VPP2    179             /* 21.48 volts */
#define THRESH  17              /*  2.04 volts */
#define CHPCHK  25              /*  2.95 volts */
#define CURRV1  255             /* maximum current limit */
#define CURRV2  255             /* maximum current limit */

int     vccv;
int     vihv1;
int     vihv2;
int     vpp1v1;
int     vppiv2;
int     vpp2v1;
int     vpp2v2;

#define V1      1
#define V2      2
#define V3      3

/*
        define  the run and reset buttons
*/

#define RNBUTTON        42              /* run button   */
#define RSBUTTON        43              /* reset button */

/*
        define the programming intervals
*/

#define TP1     3               /*   .3 milliseconds */
#define TP2     9               /*   .9 milliseconds */
#define TP3     36              /*  3.6 milliseconds */
#define TP4     500             /* 50.0 milliseconds */

/*
        set up led bits
*/

#define CNECT   0x08
#define PRGRAM  0x20
#define PGERR   0x10
#define SKT3    0x04
#define SKT2    0x02
#define SKT1    0x01

/*
        temporary variables for code optimization
*/

int     setup = NO;     /* successful devsetup flag     */
int      rtncode;            /* function return code          */
int      status;             /* procpin status temp           */
int      address;            /* chip address                  */
int      oldaddr;            /* last address accessed         */
int      tempaddr;           /* temporary address             */
int      index;              /* for loop index                */
int      gotdata;            /* temp for verify               */
int      data;               /* data byte                     */
char     *buffer;            /* pointer into data buffer      */
int      saddr;              /* temp for addr parameter       */
char     *sauxbuf;           /* temp for auxbuf parameter     */
int      slength;            /* temp for length parameter     */
int      stp;                /* current programming interval */
int      erra;               /* error programming for 1 msec */
int      errb;               /* error programming for 3 msec */
int      tdevnbr = 0;        /* temp for device number index */
int      mask;               /* address bit mask              */
int      (*dr) ();           /* pointer to devread function  */
int      (*dw) ();           /* pointer to devwrite function */
int      (*pv) ();           /* pointer to progverf function */
int      (*cd) ();           /* pointer to check data funct. */
/*
         set up pointer to device error number and configuration table
*/

int      *deverror = 0x50c;
int      *erraddrs = 0x50e;
CCONFIG *config    = 0x510;

/*
         devsetup - initial hardware setup

         entry:  devnbr - the index number of the eprom
                         0 - 2516/2716
                         1 - 2532
                         2 - 2732
                         3 - 2732a
                         4 - 2564
                         5 - 2764

         exit:   return the following:
                         ERROR - omniprogrammer not connected or devnbr out
                                 of range
                         OK - no problems

         algorithm:      initialize the hardware
                         disable all pins
                         check devnbr to make sure it is in range
                         set up the voltages for the target eprom
                         map the pins to their socket relative values
                         return

         notes:  all pins are disabled on exit
*/
```

```
int     devsetup        (devnbr)
int     devnbr;
{
        register        int     i;

        int     dr2716 ();
        int     dw2716 ();
        int     pv2716 ();
        int     cd2716 ();
        int     dr2532 ();
        int     dw2532 ();
        int     pv2532 ();
        int     cd2532 ();
        int     dr2732 ();
        int     dw2732 ();
        int     pv2732 ();
        int     cd2732 ();
        int     dwa2732 ();
        int     dr2564 ();
        int     dw2564 ();
        int     pv2564 ();
        int     cd2564 ();
        int     dr2764 ();
        int     dw2764 ();
        int     pv2764 ();

        tdevnbr = devnbr;

        (*init) ();
        (*disall) ();
        *deverror = 0;
        *erraddrs = 0;
        rtncode = OK;
        setup = NO;
        (*setcv1) (CURRV1);
        (*setcv2) (CURRV2);

/*
        is anybody out there?
*/

        if (!(*connect) ()) {
                *deverror = CNCTERR;
                rtncode   = ERROR;
        }
        else {
                (*ledon) (CNECT);
        }

        (*ledoff) (PRGRAM | PGERR | SKT1 | SKT2 | SKT3);

/*
        make sure the socket is empty
*/
```

```
        if (rtncode == OK) {
                (*setthr) (CHPCHK);
                if (!empty ()) {
                        *deverror = NOTEMPTY;
                        rtncode = ERROR;
                }
        }

/*
        set up voltages for the target device
*/

        if (rtncode == OK) {
                vccv   = calibrate (VCC, V3);
                vihv1  = calibrate (VIH, V1);
                vihv2  = calibrate (VIH, V2);
                vpp1v1 = calibrate (VPP1, V1);
                vpp1v2 = calibrate (VPP1, V2);
                vpp2v1 = calibrate (VPP2, V1);
                vpp2v2 = calibrate (VPP2, V2);

                (*disall) ();
                (*setv1)  (vihv1);
                (*setv2)  (vihv2);
                (*setv3)  (vccv);
                (*setthr) (THRESH);
        }

        switch (tdevnbr) {
        case 0:
        case 1:
        case 2:
        case 3:
                config->pincnt = 24;
                break;
        case 4:
        case 5:
                config->pincnt = 28;
                break;
        default:
                *deverror = PARMERR;
                rtncode = ERROR;
        }

/*
        map the pins to their socket relative values
*/

        switch (tdevnbr) {
        case 0:
                cepgm  = 30;                    /* pin 18 */
                oe     = 32;                    /* pin 20 */
                vpp    = 33;                    /* pin 21 */
                vcc    = 36;                    /* pin 24 */
                adrtab = amap1;
                config->cinfo.cprom.promlen = 2048;
                mask    = 0x07ff;
                dr      = dr2716;
                dw      = dw2716;
                pv      = pv2716;
                cd      = cd2716;
                break;
        case 1:
                pdpgm   = 32;                   /* pin 20 */
                vpp     = 33;                   /* pin 21 */
                vcc     = 36;                   /* pin 24 */
                adrtab = amap2;
                config->cinfo.cprom.promlen = 4096;
                mask   = 0x0fff;
                dr     = dr2532;
                dw     = dw2532;
                pv     = pv2532;
                cd     = cd2532;
                break;
        case 2:
        case 3:
                cepgm   = 30;                   /* pin 18 */
                oevpp   = 32;                   /* pin 20 */
                vcc     = 36;                   /* pin 24 */
                adrtab = amap3;
                config->cinfo.cprom.promlen = 4096;
                mask   = 0x0fff;
                dr     = dr2732;
                pv     = pv2732;
                cd     = cd2732;
                if (devnbr == 2)
                        dw = dw2732;
                else
                        dw = dwa2732;
                break;
        case 4:
                vpp     = 11;                   /* pin  1 */
                cs1     = 12;                   /* pin  2 */
                pdpgm   = 32;                   /* pin 22 */
                vcca    = 36;                   /* pin 26 */
                cs2     = 37;                   /* pin 27 */
                vcc     = 38;                   /* pin 28 */
                adrtab = amap4;
                config->cinfo.cprom.promlen = 8192;
                mask    = 0x1fff;
                dr      = dr2564;
                dw      = dw2564;
                pv      = pv2564;
                cd      = cd2564;
                break;
        case 5:
                vpp     = 11;                   /* pin  1 */
                ce      = 30;                   /* pin 20 */
                oe      = 32;                   /* pin 22 */
                pgm     = 37;                   /* pin 27 */
                vcc     = 38,                          /* pin 28 */
                adrtab = amap5;
                config->cinfo.cprom.promlen = 8192;
                mask    = 0x1fff;
                dr      = dr2764;
                dw      = dw2764;
                pv      = pv2764;
                cd      = cd2716;
                break;
        }

        if (rtncode != OK) {
                (*ledon) (PGERR);
        }
        else {
                (*ledon) (SKT1);
                setup = YES;
        }

        return (rtncode);
}

/*
        devread - read the 8 bit byte addressed

        entry:  addr - the address of the byte to read

        exit:   return the following:
                ERROR - pin number out of range or invalid parameter
                the value of the addressed byte if successful

        algorithm:      check the range of addr, if out of range
                                return ERROR
                        power on the chip
                        apply the address to the address pins
                        read the output pins
                        return the value read from the output

        notes:  it is assumed that the pins are disabled on entry.
                all pins are disabled on exit.
*/

int     devread (addr, auxbuf, length)
int     addr;
char    *auxbuf;
int     length;
{
        int     readbyte ();

        saddr   = addr;
        sauxbuf = auxbuf;
        slength = length;

/*
        check to see if we are reading the gang buttons
```

```
*/

        if ((*connect) ()) {
                if (saddr == RUNBUTTON) {
                        if ((*rdpin) (RNBUTTON) == NO)
                                return (YES);
                        else
                                return (NO);
                }

                if (saddr == RESETBUTTON) {
                        if ((*rdpin) (RSBUTTON) == NO)
                                return (YES);
                        else
                                return (NO);
                }
        }

/*
        are we ready to go?
*/

        if (promsetup () < 0) {
                (*ledon) (PGERR);
                return (ERROR);
        }

        (*dr) ();
        procpin (readbyte);

        (*disall) ();
        return (slength);
}

int     promsetup ()
{

/*
        is the hardware set up?
*/

        if ((setup == NO) || ((*getthr) () == 0)) {
                if (devsetup (tdevnbr) < 0)
                        return (ERROR);
        }

/*
        is anybody out there?
*/

        if (!(*connect) ()) {
                *deverror = CNCTERR;
                return (ERROR);
        }
        (*ledoff) (PGERR);
```

```
          (*ledon) (CNECT);

/*
          is the chip in the socket and in the right place?
*/

          (*setthr) (CHPCHK);
          if (empty ()) {
                    *deverror = NOCHIP;
                    return (ERROR);
          }

          if ((*rdpin) (vcc) != 0) {
                    *deverror = ORIENTATION;
                    return (ERROR);
          }

          (*setthr) (THRESH);

          return (OK);
}

static
int       dr2716 ()
{

          (*alt2pin) (vcc);
          (*setpin) (vpp);
          (*respin) (cepgm);
          (*respin) (oe);
          return;
}

static
int       dr2532 ()
{

          (*alt2pin) (vcc);
          (*setpin) (vpp);
          (*respin) (pdpgm);
          return;
}

static
int       dr2732 ()
{

          (*alt2pin) (vcc);
          (*respin) (cepgm);
          (*respin) (oevpp);
          return;
}

static
int       dr2564 ()
```

```
{

        (*setpin) (vcc);
        (*alt2pin) (vcca);
        (*alt2pin) (vpp);
        (*respin) (cs1);
        (*respin) (cs2);
        (*respin) (pdpgm);
        return;
}

static
int     dr2764 ()
{

        (*setpin) (vcc);
        (*setpin) (vpp);
        (*respin) (ce);
        (*respin) (oe);
        return;
}

/*
        devwrite - write a value to a 8 bit byte

        entry:  data - the value of the byte to write
                addr - the address of the byte to write

        exit:   return the following:
                        ERROR - invalid parameter or unsuccessful write
                        data - return the data if successful

        algorithm:

        notes:  it is assumed that all pins are disabled on entry.
                all pins are disabled on exit.
*/



int     devwrite        (addr, auxbuf, length)
int     addr;
char    *auxbuf;
int     length;
{

        saddr   = addr;
        sauxbuf = auxbuf;
        slength = length;

        if (promsetup () < 0) {
                (*ledon) (PGERR);
                return (ERROR);
        }
        (*ledon) (PRGRAM);

        (*dw) ();

        (*disall) ();
        (*ledoff) (PRGRAM);

        if (rtncode < 0) {
                *deverror = PROGERR;
                (*ledon) (PGERR);
        }
        else
                *deverror = 0;

        return (rtncode);
}

static
int     dw2716 ()
{
        int     progpin ();

        (*setvl)  (vpplvl);
        (*respin) (cepgm);
        (*alt2pin) (vcc);
        (*setpin) (vpp);

        rtncode = procpin (progpin);

        (*setvl)  (vihvl);
        (*dispin) (vpp);
        return;
}

static
int     dw2532 ()
{
        int     progpin ();

        (*setvl)  (vpplvl);
        (*alt2pin) (vcc);
        (*setpin) (vpp);

        rtncode = procpin (progpin);

        (*setvl)  (vihvl);
        (*dispin) (vpp);
        return;
}

static
int     dw2732 ()
{
        register int    i;
        int     progpin ();

        (*resall) ();
        for (i = 0; i <= vpp1v2; i++) {
                (*delay)  (2);
                (*setv2)  (i);
        }
        (*disall) ();
        (*alt2pin) (vcc);
        (*setpin) (oevpp);

        rtncode = procpin (progpin);

        (*dispin) (oevpp);
        (*setv2)  (vihv2);
        return;
}

static
int     dwa2732 ()
{
        register int    i;

        int     progpin ();

        (*resall) ();
        for (i = 0; i <= vpp2v2; i++) {
                (*delay)  (2);
                (*setv2)  (i);
        }
        (*disall) ();
        (*alt2pin) (vcc);
        (*setpin) (oevpp);

        rtncode = procpin (progpin);

        (*dispin) (oevpp);
        (*setv2)  (vihv2);
        return;
}

static
int     dw2564 ()
{
        int     progpin ();

        (*setv1)  (vpp1v1);
        (*setpin) (vcc);
        (*alt2pin) (vcca);
        (*setpin) (vpp);
        (*respin) (cs1);
        (*respin) (cs2);

        rtncode = procpin (progpin);
        (*setv1)  (vihv1);
        (*dispin) (vpp);
        return;
}

static
int     dw2764 ()
{
        int     progpin ();

        (*setv1)  (vpp2v1);
        (*setpin) (vcc);
        (*setpin) (vpp);
        (*respin) (ce);

        rtncode = procpin (progpin);

        (*setv1)  (vihv1);
        (*dispin) (vpp);
        return;
}

int     procpin (function)
int     (*function) ();
{

        oldaddr = 0xFFFF;
        address = saddr;
        buffer  = sauxbuf;

        for (index = 0; index < slength; index++) {

                addrinc ();

                if ((*function) () < 0) {
                        *erraddrs = address;
                        return (ERROR);
                        }

                oldaddr = address;
                address++;
                buffer++;

        } /* end of for */

        return (slength);

}

static
int     progpin ()
{

        (*cd) ();
        if (gotdata == *buffer)
```

```
                return (OK);
        stp = TP1;
        erra = progverf ();
        stp = TP2;
        errb = progverf ();
        if (erra == NO) {
                stp = TP3;
                progverf ();
        }
        if (errb == NO) {
                stp = TP4;
                if (progverf () == NO) {
                        *deverror = PROGERR;
                        return (ERROR);
                }
        }
        return (OK);
}

int     progverf ()
{
        (*pv) ();
        if (gotdata != *buffer)
                return (NO);
        else
                return (YES);
}

static
int     pv2716 ()
{

        data    = *buffer;

        data1set ();

        (*dispin) (cepgm);
        (*delay) (stp);
        (*respin) (cepgm);

        data1dis ();

        cd2716 ();
}

static
int     cd2716 ()
{

        (*respin) (oe);
        gotdata = data1get ();
        (*dispin) (oe);

}
```

```
static
int       pv2532 ()
{

          data      = *buffer;

          data1set ();

          (*respin) (pdpgm);
          (*delay) (stp);
          (*dispin) (pdpgm);

          data1dis ();

          cd2532 ();

}

static
int       cd2532 ()
{

          (*setvl)  (vihvl);
          (*delay)  (9);
          (*respin) (pdpgm);
          gotdata = data1get ();
          (*dispin) (pdpgm);
          (*setvl)  (vpplvl);

}

static
int       pv2732 ()
{

          data      = *buffer;

          data1set ();

          (*respin) (cepgm);
          (*delay) (stp);
          (*dispin) (cepgm);

          data1dis ();

          cd2732 ();

}

static
int       cd2732 ()
{

          (*respin) (oevpp);
          (*respin) (cepgm);
```

```
        gotdata = data1get ();
        (*dispin) (cepgm);
        (*setpin) (nevpp);

}

static
int     pv2564 ()
{

        data     = *buffer;

        data1set ();

        (*respin) (pdpgm);
        (*delay) (stp);
        (*dispin) (pdpgm);

        data1dis ();

        cd2564 ();

}

static
int     cd2564 ()
{

        (*setvl)  (vihvl);
        (*delay)  (9);
        (*respin) (pdpgm);
        gotdata = data1get ();
        (*dispin) (pdpgm);
        (*setvl)  (vpp1vl);

}

static
int     pv2764 ()
{

        data     = *buffer;

        data1set ();

        (*respin) (pgm);
        (*delay) (stp);
        (*dispin) (pgm);

        data1dis ();

        cd2716 ();

}
```

```
static
int     readbyte ()
{
        *buffer = dataiget ();
        return (OK);
}

/* End of tepromi.c */
```

USER SPECIFIED ALGORITHMS

User specified algorithms METHOD II:

(see following pages)

MMITWA.C/MMITWA.TAB

User Modifications   METHOD II Example:

```
/*                FILE = MMITWA.C                                              */


/*                              MMI Ti-W Bipolar PROMs                         */


/*                                          last modified:   3/30/84   by. SK   */


#include "a:pstdio.h"

#include          "a:ccunfig.h"

/*
        Setup the function pointers for the hardware interface routines
*/

#include          "a:amhdw.h"

/*
        set up variables to hold pin numbers
*/

int     ce;
int     e1;
int     e2;
int     vcc;

/*
        tables for chip programming
*/

#include  "b:mmitwa.tab"


/*
        define voltages needed by the proms
*/

#define VIL      0                /*  0    volts */
#define VCC      45               /*  5.31 volts */
#define VCCP     101              /* 11.92 volts */
#define VOUT1    97               /* 11.45 volts */
#define VOUT2    97               /* 11.45 volts */
#define THRESH   17               /*  2.01 volts */
#define CHPCHK   25               /*  2.95 volts */
#define CURRV1   255              /* maximum current limit */
#define CURRV2   255              /* maximum current limit */

int     vil;
int     vcch;
int     vccp;
int     vout1;
int     vout2;
```

```
#define V1      1
#define V2      2
#define V3      3

/*
        define the programming intervals
*/

#define TP      4                  /* 400 micro seconds */

/*
        set up led bits
*/

#define CNECT   0x08
#define PRGRAM  0x20
#define PGERR   0x10
#define SKT3    0x04
#define SKT2    0x02
#define SKT1    0x01
#define MAXTRY    10

/*
        temporary variables for code optimization
*/

int     setup = NO;     /* successful devsetup flag      */
int     dabit;          /* data bit                      */
int     rtncode;        /* function return code          */
int     status;         /* procpin status temp           */
int     address;        /* chip address                  */
int     oldaddr;        /* last address accessed         */
int     tempaddr;       /* temporary address             */
int     index;          /* for loop index                */
int     gotdata;        /* temp for verify               */
int     data;           /* data byte                     */
char    *buffer;        /* pointer into data buffer      */
int     saddr;          /* temp for addr parameter       */
char    *sauxbuf;       /* temp for auxbuf parameter     */
int     slength;        /* temp for length parameter     */
int     tdevnbr = 0;    /* temp for device number index */
int     mask;           /* address bit mask              */
char    *dptr;          /* pointer into data pin array   */
char    *paddrs;        /* pointer to address string     */
char    *pdatas;        /* pointer to data string        */
char    *pprerds;       /* pointer to pre-read string    */
char    *ppords;        /* pointer to post-read string   */
char    *pwrs;          /* pointer to write string       */
char    *prddata;       /* pointer to read data          */
char    *powrs;         /* pointer to offset in write string    */
int     promwid;        /* width of prom                 */


/*
        set up pointer to device error number and configuration table
```

```
*/

int     *deverror = 0x50c;
int     *erraddrs = 0x50e;
CCONFIG *config   = 0x510;

/*
        devsetup - initial hardware setup

        entry:  devnbr - the index number of the eprom

        exit:   return the following:
                        ERROR - omniprogrammer not connected or devnbr out
                                of range
                        OK - no problems

        algorithm:      initialize the hardware
                        disable all pins
                        check devnbr to make sure it is in range
                        set up the voltages for the target eprom
                        map the pins to their socket relative values
                        return

        notes:  all pins are disabled on exit
*/

int     devsetup        (devnbr)
int     devnbr;
{
        register        int     i;


        tdevnbr = devnbr;

        (*init) ();
        (*disall) ();
        *deverror = 0;
        *erraddrs = 0;
        rtncode = OK;
        setup = NO;
        (*setcv1) (CURRV1);
        (*setcv2) (CURRV2);

/*
        is anybody out there?
*/

        if (!(*connect) ()) {
                *deverror = CNCTERR;
                rtncode   = ERROR;
        }
        else {
                (*ledon) (CNECT);
        }
        (*ledoff) (PRGRAM | PGERR | SKT1 | SKT2 | SKT3);

/*
        make sure the socket is empty
*/

        if (rtncode == OK) {
                (*setthr) (CHPCHK);
                if (!empty ()) {
                        *deverror = NOTEMPTY;
                        rtncode = ERROR;
                }
        }

/*
        set up voltages for the target device
*/

        if (rtncode == OK) {
                vil  = VIL;
                vcch = calibrate (VCC, V3);
                vccp = calibrate (VCCP, V3);
                vout2 = calibrate (VOUT2, V2);
                vout1 = calibrate (VOUT1, V1);

                (*disall) ();
                (*setv1)  (vcch);
                (*setv2)  (vcch);
                (*setv3)  (vcch);
                (*setthr) (THRESH);
        }

/*
        map the pins to their socket relative values
*/

        paddrs = &addrlst[devnbr][0];
        pdatas = &datalst[devnbr][0];
        pprerds = &prerds[devnbr][0];
        ppords = &pords[devnbr][0];
        pwrs     = &wrstr[devnbr][0];
        prddata = &rddata[devnbr][0];
        powrs = &owrs[devnbr][0];
        config->pincnt = numpins[devnbr];
        config->cinfo.cprom.promlen = prleng[devnbr];
        config->cinfo.cprom.promwid = promwid = prwid[devnbr];
        config->socket = socksize[devnbr];
        config->cinfo.cprom.promblk = 0;
        if (devnbr = 3) {
                *(pwrs + (*(powrs+2))) = vout2;
        }
        else {
                *(pwrs + (*(powrs+2))) = vout1;
        }
        *(pwrs + (*pwrs)) = vccp;
        *(pwrs + (*(pwrs+4))) = vcch;
        *(pwrs + (*(pwrs+5))) = vil;

        if (rtncode != OK) {
                (*ledon) (PGERR);
        }
        else {
                setup = YES;
                if (socksize[devnbr] == 1)
                        (*ledon) (SKT1);
                else
                        (*ledon) (SKT3);
        }

        return (rtncode);
}

/*
        devread - read the 8 bit byte addressed

        entry:  addr - the address of the byte to read

        exit:   return the following:
                ERROR - pin number out of range or invalid parameter
                the value of the addressed byte if successful

        algorithm:      check the range of addr, if out of range
                                return ERROR
                        power on the chip
                        apply the address to the address pins
                        read the output pins
                        return the value read from the output

        notes:  it is assumed that the pins are disabled on entry.
                all pins are disabled on exit.
*/

int     devread (addr, auxbuf, length)
int     addr;
char    *auxbuf;
int     length;
{
        int     data;

        saddr   = addr;
        sauxbuf = auxbuf;
        slength = length;

/*
        are we ready to go?
*/

        if (promsetup () < 0) {
                (*ledon) (PGERR);
```

```
                        return (ERROR);
                }
                oldaddr = 0xFFFF;
                stream(opprerds);
                for (saddr=addr; saddr<(addr+length); saddr++) {
                        mapword(oldaddr,saddr,paddrs);
                        data = assemble(prddata);
                        *auxbuf++ = data;
                        oldaddr = saddr;
                };

                stream(ppords);
                (*disall) ();
                return (slength);
}

int             promsetup ()
{

/*
                is the hardware set up?
*/

                if ((setup == NO) || ((*getthr) () == 0)) {
                        if (devsetup (tdevnbr) < 0)
                                return (ERROR);
                }

/*
                is anybody out there?
*/

                if (!(*connect) ()) {
                        *deverror = CNCTERR;
                        return (ERROR);
                }
                (*ledoff) (PCERR);
                (*ledon) (CNECT);

/*
                is the chip in the socket and in the right place?
*/

                (*setthr) (CHPCHK);
                if (empty ()) {
                        *deverror = NOCHIP;
                        return (ERROR);
                }

                if ((*rdpin) (vcc) != 0) {
                        *deverror = ORIENTATION;
                        return (ERROR);
                }

                (*setthr) (THRESH);
```

```
        return (OK);
}

/*
        devwrite - write a value to a 8 bit byte

        entry:  data - the value of the byte to write
                addr - the address of the byte to write

        exit:   return the following:
                        ERROR - invalid parameter or unsuccessful write
                        data - return the data if successful

        algorithm:

        notes:  it is assumed that all pins are disabled on entry.
                all pins are disabled on exit.
*/



int     devwrite        (addr, auxbuf, length)
int     addr;
char    *auxbuf;
int     length;
{
        int     data;
        register  shftr;
        int     numtry;
        int     promdata;
        int     i;

        saddr   = addr;
        sauxbuf = auxbuf;
        slength = length;
        rtncode = length;
/*
        are we ready to go?
*/

        if (promsetup () < 0) {
                (*ledon) (PGERR);
                return (ERROR);
        }
        (*ledon) (PRGRAM);


/*      this is where the write algorithm goes      */

        oldaddr = 0xFFFF;
        for (saddr = addr; saddr < addr + length && rtncode >= 0, saddr++) {
                mapword(oldaddr,saddr,paddrs);
                oldaddr = saddr;
                stream (pprerds);
                promdata = assemble(prddata);
                stream (ppords);
                data = *auxbuf++;
                shftr = 1;
           if(data != promdata) {
                for (i=0; i<promwid; i++) {
                        if (data & shftr) {
                                cpytriad((pdatas + i*6), (pwrs + *(powrs+0)));
                                cpytriad((pdatas + i*6 + 3), (pwrs + *(powrs + 1)));
                                for (numtry = 0;(!(promdata & shftr)) && (numtry < MAXTRY
                                        stream(pwrs);
                                        stream(pprerds);
                                        promdata = assemble(prddata);
                                        stream(ppords);
                                        if (promdata & shftr) {
                                                stream(pwrs);
                                                send();
                                                send();
                                                send();
                                                send();
                                        }
                                }
                        }
                        shftr = shftr << 1;
                }
                (*setv3) (vcch);
                stream (pprerds);
                promdata = assemble(prddata);
                stream (ppords);
                if (promdata != data) {
                        *erraddrs = saddr;
                        rtncode = ERROR;
                }
           }
        }
        (*disall) ();
/*      end of write algorithm                                  */

        (*ledoff) (PRGRAM);

        if (rtncode < 0) {
                *deverror = PROGERR;
                (*ledon) (PGERR);
        }
        else
                *deverror = 0;

        return (rtncode);
}

/* End of template */
```

METHOD II Example con't.

```
/*                FILE = MMITWA.TAB                  */

/*                         MMI Ti-W pin tables                                     */

/*        devnum           device            size        socket   req   numpin

           0               63s141            256 X 4        3       n       16
           1               63s241            512 X 4        3       n       16
           2               63s841            2048 X 4       3       n       18
           3               63s1641           4096 X 4       3       n       20
*/

static int numpins[4] ={16,16,18,20};

static int socksize[4] = {3,3,3,3};

static int prleng[4] = {256.512,2048,4096};

static int prwid[4] = {4,4,4.4};

static char prerds[4][10] = {

{0X0C,0X03,0XFF,0X00,0X00,0XFF,0X01,0X00,0XFB,0xFF},

{0X0C 0X03,0XFF,0X00.0X00,0XEF.0xFF}.

{0X0D,0X30,0XFF.0X00,0X00 0XFD,0xFF},

{0X0D.0X0C,0XFF.0X00,0X00,0XF7,0X00,0X00,0XFD,0xFF},

};

static char perds[4][10] = {

{0X00,0X10,0XFF,0X01,0X04,0XFF,0X0C,0X00.0XFC,0xFF},

{0X00,0X10,0XFF.0X0C.0X00,0XFC,0xFF},

{0X00,0X02,0XFF,0X0D,0X00,0XCF,0xFF},

{0X00,0X01.0XFF.0X00,0X02,0XFF 0X0D,0X00,0XF8,0xFF},

};

static  char     swrs[4][6] = {
                              {8,9,14,24,22,35},
                              {5,6,11,21,29,32},
                              {5,6,11,21,29,32},
                              {8,9,14,42,50,53},
                           };

static char wrstr[4][50]= {

{0X01,0X00,0XFD,0X0C.0X03,0XFF,0X12,0XFF, 100,0XFF,0XFF,0XFF,0X10.0XFF,  94,
0X00,0X00,0XFF,0X00,0X00,0XEF,0X00,0X10,0XFF,0XFF,0XFF,0XFF,0X0C,0X00,0XFC,
0X12,0XFF,  44,0X10,0XFF,    0,0X01,0X04,0XFF,0xFF},

{0X0C,0X03,0XFF,0X12,0XFF, 100,0XFF,0XFF,0XFF,0X10,0XFF,  94,0X00,0X00,0XFF,
0X00,0X00,0XEF,0X00,0X10,0XFF,0XFF,0XFF,0XFF,0X0C,0X00,0XFC,0X12,0XFF,
  44,0X10,0XFF,    0,0xFF},

{0X0D,0X30,0XFF,0X12,0XFF, 100,0XFF,0XFF,0XFF,0X10,0XFF,  94,0X00,0X00,0XFF,
0X00,0X00,0XFD,0X00,0X02,0XFF,0XFF,0XFF,0XFF,0X0D,0X00,0XCF,0X12,0XFF,  44,
0X10,0XFF,    0,0xFF},

{0X00,0X00,0XF7,0X0D,0X0C,0XFF,0X12,0XFF, 100,0XFF,0XFF,0XFF,0X11,0XFF,  94,
0X00,0X00,0XFF,0X00,0X00,0XFF,0X00,0X00,0XFF,0X00,0X00,0XFF,0X00,0X00,0XFF,
0X00,0X00,0XFF,0X00,0X00,0XFF,0X00,0X00,0XFD,0X00,0X02,0XFF,0XFF,0XFF,0XFF,
0X00,0X00,0XF3,0X12,0XFF,  44,0X11,0XFF,    0,0X00,0X08,0XFF,0xFF},

};

static char addrlst[43][79] = {

{8,0X01,0X00,0XFD,0X01,0X02,0XFF,0X01,0X00,0XFE,0X01,0X01,0XFF,0X00,0X00,0XF7,
0X00,0X08,0XFF,0X02,0X00,0X7F,0X02,0X80,0XFF,0X01,0X00,0XF7,0X01,0X08,0XFF,
0X01,0X00,0XEF,0X01,0X10,0XFF,0X01,0X00,0XDF,0X01,0X20,0XFF,0X00,0X00,0XDF,
0X00,0X20,0XFF},

{9,0X01,0X00,0XFD,0X01,0X02,0XFF,0X01,0X00,0XFE,0X01,0X01,0XFF,0X00,0X00,0XF7,
0X00,0X08,0XFF,0X02,0X00,0X7F,0X02,0X80,0XFF,0X01,0X00,0XF7,0X01,0X08,0XFF,
0X01,0X00,0XEF,0X01,0X10,0XFF,0X01,0X00,0XDF,0X01,0X20,0XFF,0X00,0X00,0XDF,
0X00,0X20,0XFF,0X01,0X00,0XFB,0X01,0X04,0XFF},

{11,0X02,0X00,0X7F,0X02,0X80,0XFF,0X01,0X00,0XFD,0X01,0X02,0XFF,0X01,0X00,0XFE,
0X01,0X01,0XFF,0X01,0X00,0XF7,0X01,0X08,0XFF,0X01,0X00,0XEF,0X01,0X10,0XFF,
0X01,0X00,0XDF,0X01,0X20,0XFF,0X01,0X00,0XBF,0X01,0X40,0XFF,0X04,0X00,0XDF,
0X04,0X20,0XFF,0X00,0X00,0XDF,0X00,0X20,0XFF,0X01,0X00,0XFB,0X01,0X04,0XFF,
0X00,0X00,0XF7,0X00,0X08,0XFF},

{12,0X01,0X00,0XF7,0X01,0X08,0XFF,0X02,0X00,0X7F,0X02,0X80,0XFF,0X01,0X00,0XFD,
0X01,0X02,0XFF,0X01,0X00,0XEF,0X01,0X10,0XFF,0X01,0X00,0XDF,0X01,0X20,0XFF,
0X01,0X00,0XBF,0X01,0X40,0XFF,0X00,0X00,0X7F,0X00,0X80,0XFF,0X03,0X00,0XBF,
0X03,0X40,0XFF,0X04,0X00,0XDF,0X04,0X20,0XFF,0X00,0X00,0XDF,0X00,0X20,0XFF,
0X01,0X00,0XFE,0X01,0X01,0XFF,0X00,0X00,0XFE,0X00,0X01,0XFF},

};

static char datalst[43][24] = {

{0X0B,0X00,0XCF,0X0B,0X10,0XDF,0X0A,0X00,0XCF,0X0A,0X10,0XDF,0X0A,0X00,0XFC,
0X0A,0X01,0XFD,0X0A,0X00,0XF3,0X0A,0X04,0XF7},

{0X0B,0X00,0XCF,0X0B,0X10,0XDF,0X0A,0X00,0XCF,0X0A,0X10,0XDF,0X0A,0X00,0XFC,
0X0A,0X01,0XFD,0X0A,0X00,0XF3,0X0A,0X04,0XF7},

{0X0B,0X00,0XFC,0X0B,0X01,0XFD,0X0B,0X00,0XCF,0X0B,0X10,0XDF,0X0A,0X00,0XCF,
0X0A,0X10,0XDF,0X0A,0X00,0XFC,0X0A,0X01,0XFD},

{0X09,0X00,0XFD,0X09,0X02,0XFF,0X0B,0X00,0XFC,0X0B,0X02,0XFE,0X0B,0X00,0XCF,
0X08,0X20,0XEF,0X0A,0X00,0XEF,0X0A,0X20,0XEF},

};    /*  NOTE    6351641 USES V2 FOR PROGRAMMING VOLTAGE Vpp   */

static char rddata[4][9] = {

{4,0,0X02,0,0X01,0,0X04,0,0X40},

{4,0,0X02,0,0X01,0,0X04,0,0X40},

{4,0,0X01,0,0X04,0,0X40,0,0X10},

{4,0,0X04,0,0X40,0,0X10,1,0X04},

};
```

Assembly Language Assist Routines

For input to Stream

```
                        Production Assembly Routine Stream
;
;This routine takes a table of pin manipulation information which is
;very timing sensitive and massages it in the background so that it can
;be streamed to the hardware in it most efficient form.
;
;the input stream must be pointed to by the first parameter on the stack
;and be in the following format:
;
;                    port address  --  address of the omni port to be changed
;                    set mask      --  ones are bits to be set in that port
;                    clr mask      --  zeros are bits to be reset in that port
;                      .
;                      .           --  repeat as many times as needed
;                      .
;                      OXFF        --  end string with all ones
;
;
;the destination string will start at the local location "pumpdata" and
;will be in the following format:
;
;                    length        --  length of this data stream
;                    port address  --  port to be modified
;                    port data     --  data to be put at port
;                      .
;                      .           --  repeat to length specified in first byte
;
;the algorithm executes as follows:
;
;       calling sequence        stream(ptr to stream)
;
;          get port address whose data is to be modified
;          get current data at that port from iostate
;          modify bits indicated by set mask and clr mask
;          put port address into pump string
;          put desired data into pump string
;          repeat until zero encountered in stream
;          put count into first byte of pump string
;          call pump to send the pump string out at high speed
;          return
;
;          ON EXIT, register bc preserved; hl, de destroyed
;
;
'  Get the addresses for the hardware interface routines
;
;
```

For input to Mapword

```
;                                      Mapword
;
;
;    Mapword takes an 16 bit word and a map table and efficiently maps the bits
;onto the pins indicated by the map table
;
;the map table looks like this:
;
;                  #entries          the number of entries in this word
;                  port       ±
;                  setmask     >   hardware change if bit 0 of mapword is 0
;                  clrmask    /
;                  port       ±
;                  setmask     >   hardware change if bit 0 of mapword is 1
;                  clrmask    /
;                    .
;                    .             repeat for maximum number of mapped bits
;                    .             up to 16
;
;
;there are three entry parameters for mapword which are on the stack as per
;'C' conventions.
;
;
;       old word - integer containing the previous value mapped by mapword
;               this is used to optimize mapword to modify only those bits
;               which are different from one time to the next.  If data is
;               to be written for the first time, old word needs to equal
;               the exclusive or of map word.
;
;       map word - integer value containing bit pattern to be mapped
;
;       pointer to map table - pointer to list of masks to implement
;                    hardware changes to make under various conditions
;
;mapword calling sequence is:
;
;             mapword(old word, map word, pointer to map table);
;
;on return, the Omni pins will be set up with the new data pattern
;there are no return codes
;
;
;
tmpstream:   db 0ffh
             ds  48
```

For input to Mapword con't.

```
;
;The algorithm uses both register sets to promote efficiency, and after
;the setup code, the following registers are used:
;
;       de = the differences between the old data and the new data
;       hl = the new data
;        a = the offset from the current position of hl' to current bit position
;       hl' = pointer to the current position in the map table
;       de' = pointer to current position in the temporary stream data
;        b' = a temporary offset holding register
;
```

For data collection from Assemble

```
;                               Assemble Word
;
;
;       This routine takes a list of port and bit mask assignments and
;assembles a word of data from them based on the state of the pins the
;list maps to.
;
;The assemble list has the following format:
;
;                    count
;                    port        ±     port address of the MSB to be read
;                    bit mask    /     position in port of bit to read
;                      .         ±
;                      .         /     repeat two entries at a time up to 8
;                                      ending with the least significant bit
;
;The calling sequence for assemble is:
;
;                data = assemble(rddatalst);
;
;on entry:
;
;                  rddatalst is a pointer to the table shown above
;
;on exit:
;
;                  data contains the assembled data
;
;
```

PORTING PROCEDURES

PORTING OMNI SOFTWARE TO A NEW HOST MICROCOMPUTER

Omni software is targeted for the Kaypro II or IBM PC. The procedure that follows is required for operating the Omni Programmer on a host machine other than those two mentioned. It is conservatively estimated to be a 2 week effort to completely modify and test the code. Varix will assist with technical support as necessary to answer questions.

REQUIREMENTS:

1. The processor of the target host must be Z-80 or 8088 and the operating system must be CPM/80 or MSDOS respectively.

2. The machine must support a 16 bit parallel port with the following characteristics:

   8 bits output only port (address)
   8 bits bidirectional (data)
   2 strobe signals

PROCEDURE FOR PORT OF OMNI:

1. Modify the following assembly language routines in OMHDW.MAC

   (a) Change read/write port addresses in init, pin, apin, pout, apout, pump.
   (b) Adjust timing routines for processor speed - delay, short delay. (code is written for 2.5 MHz Z-80, 5 MHz 8088)

2. Reassemble the OMHDW.MAC routine as follows:

```
M80  =  OMHDW.MAC
L80     OMHDW, OMHDW/n/x/e
ZSID
*L -103
*^C
A>SAVE 16 OMHDW.OVR
```

NOTE: Omni software is distributed in one of the following formats:
8" floppy, single sided, single density for CPM
5 1/4" floppy, IBM-PC MSDOS 2.1
5 1/4" floppy, Kaypro II format, CPM

PROCEDURE FOR PORT OF OMNI COM

1. Modify interrupt initialization to support communications vector on new machine.

2. I/O driver must be modified to support the communications controller if a port other than a Z-80 SIO.

PROCEDURES

- o Varix Warranty
- o Customer Problem Reporting

  Problems to be Resolved by:

  Customer Problem Report
  Communication by Telephone
  Written Correspondence

- o Customer Response Form

Assistance Policy

Varix will be of assistance to customer problems under warranty or granted through our maintenance contract. Please contact Varix to correct any problems you have regarding hardware, software, or service.

We also encourage our customers to forward any comments that may contribute to or enhance the use or our Varix products.

You may elect to use the Customer Problem Report, the Customer Response Form, or choose to pay us a visit. We take pride in serving our customers.

User reference to: **Varix Warranty**

For a period of one year after the purchase of the Omni-Programmer, Varix Corporation will provide the following at no charge to the customer:

Immediate replacement of defective hardware with operational equipment.

Immediate correction or replacement software to remedy any user identified programming error attributable to Varix.

Upgrade of software to meet commitments to the customer made at the time of purchase.

After the warranty period, software upgrades which will both correct existing problems and provide all current software enhancements can be purchased. Defective hardware can be replaced or repaired for a single fixed cahrge. There will be a 90 day warranty period for each of these services during which any problems found will be fixed free of charge.

Alternately, the customer may purchase a yearly maintenance contract which will provide extension of the hardware warranty for that year and provide regular updates of software, including both problem fixes and enhancements, for that year. Minimum of 3 times/year.

The foregoing warranties are in lieu of all other warranties express or implied including but not limited to warranties of merchantability and fitness for a particular purpose and are the only warranties made by Varix Corporation in connection with the equipment. Related documentation furnished by Varix Corporation and the installation assistance, if any, rendered by Varix Corporation. In no event will Varix Corporation be responsible for consequential damages.

Customer Problem Reporting

Customer reference to the **Customer Problem Report Form** may expedite matters in a problem solving situation.

Also, reference to the Varix **Omni-Programmer User Manual** may assist with your in-house problem solving. In some instances this manual may aid in the definition or description of your problem.

The **Customer Problem Report Form,** included in this section, will afford the customer a current status of his immediate problem and when resolution is achieved will give him a documented source for future reference.

When a problem exists the user should complete this form by:

1) identifying the type of hardware or software in question,

2) give a complete and thorough description of the problem, and

3) list any additional information that may be of assistance to our customer service in recreating the problem at Varix for further study.

The information given above will allow Varix personnel the opportunity to readily assist the customer.

Please see "Contacting Varix" in this section to secure further procedure.

Contacting Varix

Communication by **Telephone:**

a) have your Customer Problem Report ready to help you when you phone,

b) specify to our Varix operator whether your your problem is hardware or software related,

c) use your form to write down any response or advice the customer service engineer has given you. Use the "Resolution" area provided on the form, and

d) keep your completed form for return calls or future reference.

Written Correspondence:

a) send all correspondence to our mailing address:

Varrix Corporation
122 Spanish Village #608
Dallas, TX 75248

ATTN: Customer Service

b) enclose with your cover letter a copy of the Customer Problem Report that you have completed.

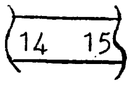
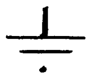
c) The resolution area of the form will then be completed by Varix and returned to you.

d) Keep your returned and completed form for further correspondence and/or future reference.

Omni-Programmer Schematics

The following pages contain the schematics for the Omni-Programmer models SP0300 and GP1140. They are provided as reference for further understanding of the interface specifications.

Nomenclature used in these schematics is:

J1 	indicates a part of a connector J1
p.6,9	are page references for the signal connection
	System ground

U31	IC	#31	Q1	transistor	#1
R15	resistor	#15	C5	capacitor	#5
D7	diode	#7	L4	LED	#4

User Diagnostics

BW is a Burn Test Program. It is a functional test of the Omni and will execute while loaded into either Drive A: or B:. It is available in any of the standard formats in which Omni software is distributed and a copy is included with this manual.

Bw working with an IBM-PC.

EXECUTABLE program.

Insert DOS Disk - (MSDOS must be running)

To indicate the default drive from A> to B>-
Type B: i.e.

```
              A>B: (press Enter)
```

BW procedure:

```
A>EXEC:BW  (press Enter ) or B>EXEC:BW (press Enter)
```

When the menu appears:

```
                   B = Continuous Burn In
                   W = Extended Wigpin Test
                   C = Current Limit Test  **Not used for functional test**
                   T = Test 1.5 Volt Circuit
                   E = Cross-effect Test
                   X = Exit
```

Depress B (Enter)

This will give you a visual test both on the computer screen and with the lights on the Omni-Programmer flashing continuously. While the Continuous Burn In file is running if there is a failure then depress any key and return to the menu.

Using the Menu continue to select the W, T, and E files to aid in locating the failure. When the failure is identified make a descriptive note as to the location and type of problem and call the Varix Customer Service Department for further assistance.

User Diagnostics: Burn Test Program BW con't.

BW working with a Kaypro computer.

```
A>COM:BW (press Return) or  B>COM:BW (press Return)
```

To access Drive B from Drive A :

```
A>B: (press Return)
```

The screen will require a y/n response to Continuous Burn In. Y will initiate the Continuous Burn In file and a N response will give you the menu.

While running the Continuous Burn In file should there be a failure then the RESET switch on the back of the Kaypro must be depressed. Bring up BW again and this time around answer N to get the BW menu.

When the menu appears:

```
W = Extended Wigpin Test
C = Current Limit Test  **Not used for functional test**
T = Test 1.5 Volt Circuit
E = Cross-effect Test
X = Exit
```

Using the menu as a guide select the W, T, and/or E file to aid in locating the failure. When the failure is identified make a descriptive note as to the location and type of problem and call the Varix Customer Service Department for further assistance.

```
        title   'production hardware interface for the kaypro ii'
        .z80
        cseg

;
;       set up the jump table
;

        jp      resall
        jp      disall
        jp      respin
        jp      dispin
        jp      setpin
        jp      alt1pin
        jp      alt2pin
        jp      qenable
        jp      qdispin
        jp      qsetpin
        jp      qalt1pin
        jp      pump
;       jp      qalt2pin
        jp      rdpin
        jp      setv1
        jp      setv2
        jp      setv3
        jp      setcv1
        jp      setcv2
        jp      setthr
        jp      getthr
        jp      ledon
        jp      ledoff
        jp      connect
        jp      init
        jp      delay
        jp      shortdelay
        jp      pin
        jp      apin
        jp      pout
        jp      apout

;
;       iostate contains the bit values written to each of the output pins.
;

iostate:        db      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h
                db      00h, 00h, 00h, 00h, 00h, 00h, 00h, 00h

;
;       pintab maps pin numbers to port addresses and bit positions.
;       the first entry is empty so that the indexes will be 1 relative.
;       each pin has the following information: primary port, primary port
;       bit mask, secondary output port, number of bits in secondary port,
;       bit mask in secondary port.

pintab: db      00h, 00h, 00h, 00h, 00h         ;dummy entry
```

```
          db        05h,  10h,  00h,  00h,  00h           ;pin # 1
          db        05h,  20h,  00h,  00h,  00h           ;pin # 2
          db        05h,  40h,  00h,  00h,  00h           ;pin # 3
          db        05h,  80h,  00h,  00h,  00h           ;pin # 4
          db        03h,  01h,  00h,  00h,  00h           ;pin # 5
          db        03h,  02h,  08h,  01h,  01h           ;pin # 6
          db        03h,  04h,  08h,  01h,  02h           ;pin # 7
          db        03h,  08h,  00h,  00h,  00h           ;pin # 8
          db        03h,  10h,  08h,  01h,  04h           ;pin # 9
          db        03h,  20h,  00h,  00h,  00h           ;pin # 10
          db        04h,  20h,  0ch,  02h,  01h           ;pin # 11
          db        02h,  10h,  09h,  01h,  80h           ;pin # 12
          db        01h,  80h,  0dh,  02h,  40h           ;pin # 13
          db        01h,  40h,  09h,  01h,  20h           ;pin # 14
          db        01h,  20h,  09h,  01h,  10h           ;pin # 15
          db        01h,  10h,  09h,  01h,  08h           ;pin # 16
          db        01h,  08h,  09h,  01h,  04h           ;pin # 17
          db        01h,  04h,  09h,  01h,  02h           ;pin # 18
          db        01h,  02h,  0ch,  02h,  10h           ;pin # 19
          db        01h,  01h,  0ch,  02h,  04h           ;pin # 20
          db        00h,  01h,  0ah,  02h,  01h           ;pin # 21
          db        00h,  02h,  0ah,  02h,  04h           ;pin # 22
          db        00h,  04h,  0ah,  02h,  10h           ;pin # 23
          db        03h,  80h,  00h,  00h,  00h           ;pin # 24
          db        00h,  08h,  0ah,  02h,  40h           ;pin # 25
          db        00h,  10h,  0bh,  02h,  01h           ;pin # 26
          db        00h,  20h,  0bh,  02h,  04h           ;pin # 27
          db        00h,  40h,  0bh,  02h,  10h           ;pin # 28
          db        00h,  80h,  0bh,  02h,  40h           ;pin # 29
          db        02h,  80h,  0ch,  02h,  40h           ;pin # 30
          db        02h,  04h,  09h,  01h,  01h           ;pin # 31
          db        04h,  80h,  08h,  01h,  10h           ;pin # 32
          db        02h,  08h,  0dh,  02h,  01h           ;pin # 33
          db        02h,  02h,  09h,  01h,  40h           ;pin # 34
          db        02h,  01h,  08h,  01h,  08h           ;pin # 35
          db        02h,  20h,  0dh,  02h,  04h           ;pin # 36
          db        02h,  40h,  08h,  01h,  20h           ;pin # 37
          db        04h,  40h,  08h,  01h,  40h           ;pin # 38
          db        04h,  10h,  00h,  00h,  00h           ;pin # 39
          db        04h,  08h,  00h,  00h,  00h           ;pin # 40
          db        04h,  04h,  00h,  00h,  00h           ;pin # 41
          db        04h,  02h,  00h,  00h,  00h           ;pin # 42
          db        04h,  01h,  00h,  00h,  00h           ;pin # 43
          db        03h,  40h,  0dh,  02h,  10h           ;pin # 44
          db        05h,  01h,  00h,  00h,  00h           ;pin # 45
          db        05h,  02h,  00h,  00h,  00h           ;pin # 46
          db        05h,  04h,  00h,  00h,  00h           ;pin # 47
          db        05h,  08h,  08h,  01h,  80h           ;pin # 48

;
;         define the port select values
;

v1volt    equ       10h                 ;select v1 voltage
v2volt    equ       11h                 ;select v2 voltage
```

```
v3volt  equ     12h             ;select v3 voltage

v1curr  equ     13h             ;select v1 current limit
v2curr  equ     14h             ;select v2 current limit

outthr  equ     15h             ;output threshold
inthr   equ     0dh             ;input threshold

ledport equ     16h             ;leds and current enables

;
;       define return values
;

true    equ     1
false   equ     0
error   equ     -1

;
;       define the indexes into pintab
;

primry  equ     0               ;primary port
prmask  equ     1               ;primary port bit mask
secndy  equ     2               ;secondary port (0 if none)
scbits  equ     3               ;number of bits in secondary port (1 or 2)
scloc   equ     4               ;location of bits in secondary port (0 - 7)

;
;       define the address of the configuration table
;

contab  equ     050ch

;
;       define the parallel interface ports
;

dbpar   equ     0ah             ;port b data on parpic
cbpar   equ     0bh             ;port b control on parpic

dbsys   equ     1eh             ;port b data on syspic
cbsys   equ     1fh             ;port b control on syspic

;dapar  equ     08h             ;port a data on parpic
;capar  equ     09h             ;port a control on parpic
;dbpar  equ     0ah             ;port b data on parpic
;cbpar  equ     0bh             ;port b control on parpic

;dasys  equ     1ch             ;port a data on syspic
;casys  equ     1dh             ;port a control on syspic
;dbsys  equ     1eh             ;port b data on syspic
;cbsys  equ     1fh             ;port b control on syspic

;
```

```
;       set up a byte to hold the state of the leds
;

leds:   db      03fh

;
;       resall - reset all pins
;
;       entry:  none
;
;       exit:   none
;

resall: push    bc              ;save bc
        ld      b,0
        ld      hl,iostate      ;get the address of iostate
rloop:  ld      a,b             ;get the port number
        cp      6               ;check the range
        jp      z,rend          ;stop if equal to 6
        ld      e,0             ;zero the port
        ld      (hl),e
        push    bc              ;save in case apout destroys their values
        push    hl
        call    apout           ;output zero to the specified port
        pop     hl              ;restore
        pop     bc
        inc     hl
        inc     b               ;increment to next port
        jr      rloop
rend:   pop     bc              ;restore bc
        ret

;
;       disall - disable all pins
;
;       entry:  none
;
;       exit:   none
;

disall: push    bc              ;save bc
        ld      b,8             ;set up first port number
        ld      hl,iostate + 8  ;get the address of the 8th port in iostate
dloop1: ld      a,b             ;get the port number
        cp      0eh             ;check the range
        jr      z,dend1         ;stop if equal to 0eh
        ld      e,0             ;zero the port
        ld      (hl),e
        push    bc              ;save in case apout destroys their values
        push    hl
        call    apout           ;output zero to the specified port
        pop     hl              ;restore
        pop     bc
        inc     hl
        inc     b               ;increment to next port
        jr      dloop1
dend1:  ld      b,0             ;set up first port number
        ld      hl,iostate      ;get the address of iostate
dloop2: ld      a,b             ;get the port number
        cp      6               ;check the range
        jr      z,dend2         ;stop if equal to 6
        ld      e,0ffh          ;set every bit in the port
        ld      (hl),e
        push    bc              ;save in case apout destroys their values
        push    hl
        call    apout           ;output zero to the specified port
        pop     hl              ;restore
        pop     bc
        inc     hl
        inc     b               ;increment to next port
        jr      dloop2
dend2:  pop     bc              ;restore bc
        ret

;
;       respin - reset a pin to its low value
;
;       entry:  parameter # 1 - pin number to be reset
;
;       exit:   none
;

respin:
        call    parm1

        push    bc              ;save bc

        call    primary         ;get the primary port information

        call    disprimary      ;disable primary bit

        pop     bc              ;restore bc
        ret

;
;       dispin - set a pin to its tri-state value
;
;       entry:  parameter # 1 - pin number to be disabled
;
;       exit:   none
;

dispin:
        call    parm1

        push    bc              ;save bc

        call    primary         ;get the primary port information

        push    bc              ;save for later
        inc     hl
        ld      a,(hl)          ;secondary port
        or      a
        jr      z,disprm
        dec     hl

        ld      d,0             ;disable decode
        call    setsecond       ;set up secondary bits

disprm: pop     bc              ;restore the primary port information

        call    enaprimary      ;enable primary bit

        pop     bc              ;restore bc
        ret

;
;       setpin - set a pin to its high value
;
;       entry:  parameter # 1 - pin number to be set
;
;       exit:   none
;

setpin:
        call    parm1

        push    bc              ;save bc

        call    primary         ;get the primary port information

        push    bc              ;save for later

        ld      d,1             ;set decode
        call    setsecond       ;set up secondary bits

        pop     bc              ;restore the primary port information

        call    enaprimary      ;enable primary bit

        pop     bc              ;restore bc
        ret

;
;       alt1pin - set a pin to its alternate high value
;
;       entry:  parameter # 1 - pin number to be set
;
;       exit:   none
;

alt1pin:
        call    parm1
        push    bc              ;save bc

        call    primary         ;get the primary port information

        push    bc              ;save for later

        ld      d,2             ;alt1 decode
        call    setsecond       ;set up secondary bits

        pop     bc              ;restore the primary port information

        call    enaprimary      ;enable primary bit

        pop     bc              ;restore bc
        ret

;
;       alt2pin - set a pin to its alternate high value
;
;       entry:  parameter # 1 - pin number to be set
;
;       exit:   none
;

alt2pin:
        call    parm1

        push    bc              ;save bc

        call    primary         ;get the primary port information

        push    bc              ;save for later

        ld      d,3             ;alt2 decode
        call    setsecond       ;set up secondary bits

        pop     bc              ;restore the primary port information

        call    enaprimary      ;enable primary bit

        pop     bc              ;restore bc
        ret

;
;       qenable - enable the primary port for a pin
;
;       entry:  parameter # 1 - pin number to be enabled
;
;       exit:   none
;

qenable:
;       call    parm1
;
;       push    bc              ;save bc
```

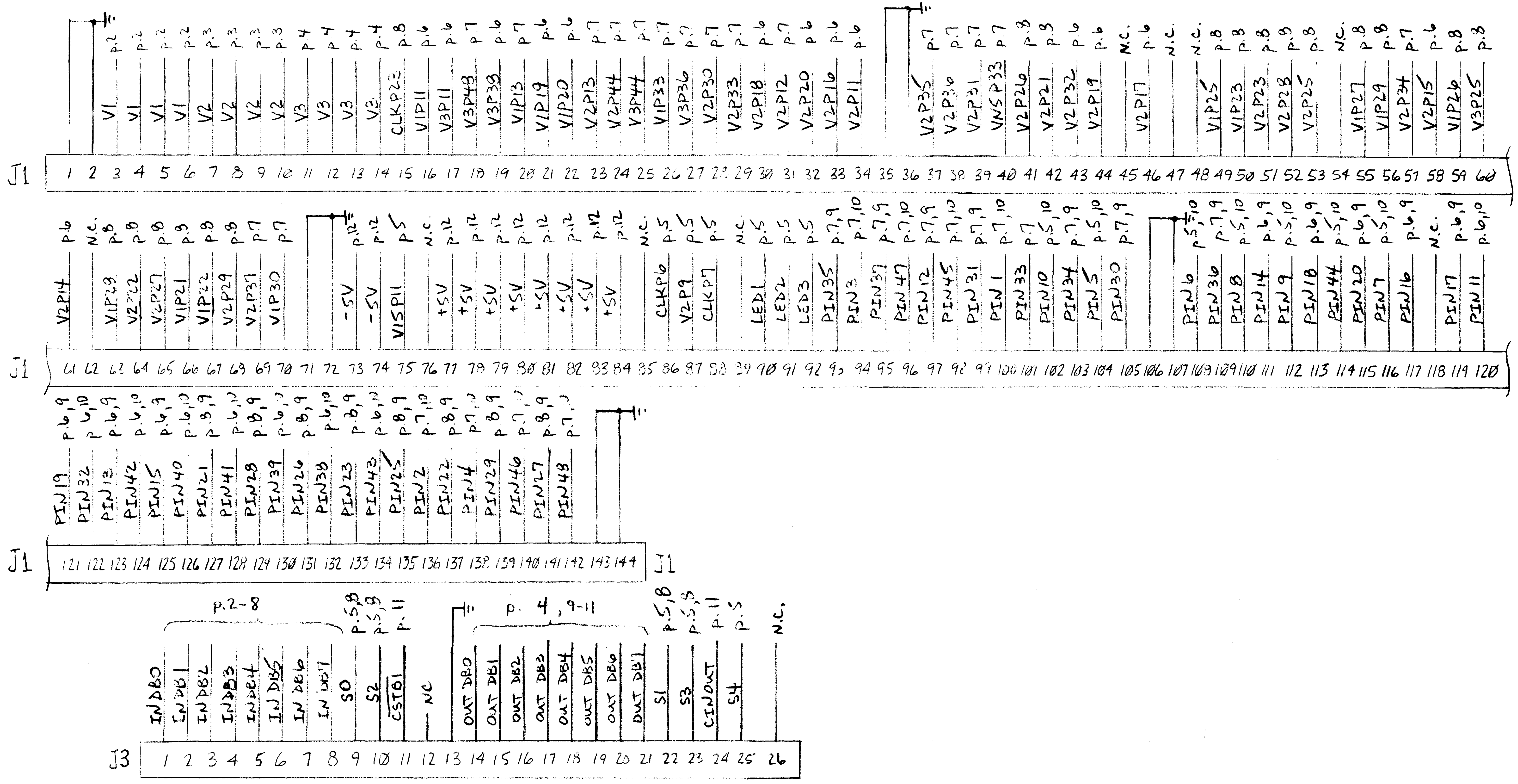
```
;
;       call    primary
;
;       call    enaprimary      ;enable primary bit
;
;       pcp     bc              ;restcre bc
;       ret

;
;       qdispin - set seccndary pcrt fcr a pin tc its tri-state value
;
;       entry:  parameter # 1 - pin number tc be disabled
;
;       exit:   ncne
;

qdispin:
;       call    parm1
;
;       push    bc              ;save bc
;
;       call    primary
;
;       inc     hl
;       ld      a,(hl)          ;seccndary pcrt
;       or      a
;       jr      z,qdisprm
;       dec     hl
;
;       ld      d,0             ;disable deccde
;       call    setseccnd       ;set up seccndary bits
;
;qdisprm:
;       pcp     bc              ;restcre bc
;       ret

;
;       qsetpin - set the seccndary pcrt cf a pin tc its high value
;
;       entry:  parameter # 1 - pin number tc be set
;
;       exit:   ncne
;

qsetpin:
;       call    parm1
;
;       push    bc              ;save bc
;
;       call    primary         ;get the primary pcrt infcrmaticn
;
;       ld      d,1             ;set deccde
;       call    setseccnd       ;set up seccndary bits
;
;       pcp     bc              ;restcre bc
;       ret

;
;       qalt1pin - set the secondary port of a pin to its alternate high value
;
;       entry:  parameter # 1 - pin number to be set
;
;       exit:   none
;

qalt1pin:
;       call    parm1
;
;       push    bc              ;save bc
;
;       call    primary         ;get the primary port information
;
;       ld      d,2             ;set decode
;       call    setsecond       ;set up secondary bits
;
;       pop     bc              ;restore bc
;       ret

;
;       qalt2pin - set the secondary port of a pin to its alternate high value
;
;       entry:  parameter # 1 - pin number to be set
;
;       exit:   none
;

qalt2pin:
;       call    parm1
;
;       push    bc              ;save bc
;
;       call    primary         ;get the primary port information
;
;       ld      d,3             ;set decode
;       call    setsecond       ;set up secondary bits
;
;       pop     bc              ;restore bc
        ret

;
;       primary - get the primary port information
;
;       entry:  register e - pin number
;
;       exit:   register b - primary port number
;               register c - primary port mask
;

primary:
        ld      a,e             ;multiply pin by 5
        sla     e               ;get the port information for the pin
        sla     e
        add     a,e
        ld      e,a
        ld      hl,pintab       ;port table
        ld      d,0
        add     hl,de           ;address in pintab of information
        ld      b,(hl)          ;primary port
        inc     hl
        ld      c,(hl)          ;pin mask
        ret

;
;       disprimary - disable the primary bit
;
;       entry:  register b - primary port number
;               register c - primary port mask
;
;       exit:   none
;

disprimary:
        ld      e,b             ;get the current state of all the bits
        ld      d,0
        ld      hl,icstate
        add     hl,de

        ld      a,c
        cpl
        and     (hl)            ;merge the select bits into the byte
        ld      (hl),a          ;save back into icstate

        ld      e,a             ;set up to call pout
        ld      a,b
        call    apout
        ret

;
;       enaprimary - enable the primary bit
;
;       entry:  register b - primary port number
;               register c - primary port mask
;
;       exit:   none
;

enaprimary:
        ld      e,b             ;get the current state of all the bits
        ld      d,0
        ld      hl,icstate
        add     hl,de

        ld      a,c             ;get the mask for oring
        or      (hl)            ;clear the secondary bits
        ld      (hl),a          ;save back into iostate

        ld      e,a             ;set up to call apout
        ld      a,b
        call    apout
        ret

;
;       setsecond - set up secondary bits
;
;       entry:  register hl - address of byte preceeding secondary port
;               register d  - type of setup:
;                               00 - disable
;                               01 - set
;                               10 - alt1
;                               11 - alt2
;
;       exit:   none
;

setsecond:
        inc     hl
        ld      b,(hl)          ;secondary port number
        inc     hl
        ld      a,(hl)          ;number of bits in secondary port
        inc     hl
        ld      c,(hl)          ;secondary pin mask
        or      a               ;check for no secondary port
        ret     z
        cp      2
        jr      nz,onebit       ;one bit decode on secondary port

        ld      a,c
        sla     a
        or      c
        cpl
        ld      e,a

        ld      a,d             ;get decode type
        or      a
        jr      nz,dec01

        ld      a,e             ;and mask in register a
        ld      c,0             ;or mask in register c
        jr      decode

dec01:  cp      1
        jr      nz,dec10

        ld      a,e             ;and mask in register a
                                ;or mask already in register c
        jr      decode

dec10:  cp      2
        jr      nz,dec11
        ld      a,e             ;and mask in register a
        sla     c               ;or mask in register c
        jr      decode

dec11:  ld      a,e
        cpl
        ld      c,a             ;or mask in register c
        cpl                     ;and mask in register a
        jr      decode

onebit: ld      a,c
        cpl
        ld      e,a

        ld      a,d
        cp      1
        jr      z,dec1

        ld      a,e             ;and mask in register a
        ld      c,0             ;or mask in register c
        jr      decode

dec1:   ld      a,e             ;and mask in register a
                                ;or mask already in c

decode: ld      e,b             ;get the current state of all the bits
        ld      d,0
        ld      hl,iostate
        add     hl,de

        and     (hl)            ;clear the secondary bits
        or      c               ;set the decode pattern

        ld      (hl),a

        ld      e,a             ;set up to call apout
        ld      a,b
        call    apout
        ret

;
;       rdpin - read the value of a pin
;
;       entry:  parameter # 1 - pin number to be read
;
;       exit:   return true  if the pin = 1
;                      false if the pin = 0
;

rdpin:  call    parm1

        push    bc              ;save bc

        call    primary         ;get the primary port information
        push    bc              ;save for later

        ld      a,b             ;set up to call pin
        call    apin            ;get the value for the pin
        pop     bc

        and     c               ;check the value of the pin

        pop     bc

        jr      nz,rtrue
        ld      hl,false        ;bit was not set
        ret

rtrue:  ld      hl,true         ;bit was set
        ret

;
;       setv1 - set the voltage of v1
;
;       entry:  parameter # 1 - the number of volts to set v1 to
;
;       exit:   none
;
;       notes:  the voltage can be set to anything between 0 and 30.60
;               volts in increments of 0.12 volts
;

setv1:  ld      a,v1volt        ;v1 output port is in register a
        jr      setreg          ;output the voltage level

;
;       setv2 - set the voltage of v2
;
;       entry:  parameter # 1 - the number of volts to set v2 to
;
;       exit:   none
;
;       notes:  the voltage can be set to anything between 0 and 30.60
;               volts in increments of 0.12 volts
;

setv2:  ld      a,v2volt        ;v2 output port is in register a
        jr      setreg          ;output the voltage level

;
;       setv3 - set the voltage of v3
;
;       entry:  parameter # 1 - the number of volts to set v3 to
;
;       exit:   none
;
;       notes:  the voltage can be set to anything between 0 and 30.60
;               volts in increments of 0.12 volts
```

```
;
setv3:  ld      a,v3vclt        ;v3 cutput pcrt is in register a
        jr      setreg          ;cutput the vcltage level

;
;       setcv1 - set the current limit cf v1
;
;       entry:  parameter # 1 - the number cf milliamps tc set v1 tc
;
;       exit:   ncne
;
;       nctes:  the vcltage can be set tc anything between 0 and 1020
;               milliamps in increments cf X milliamps
;

setcv1: ld      a,v1curr        ;v1 cutput pcrt is in register a
        jr      setreg          ;cutput the current level

;
;       setcv2 - set the current cf v2
;
;       entry:  parameter # 1 - the number cf milliamps tc set v2 tc
;
;       exit:   ncne
;
;       nctes:  the vcltage can be set tc anything between 0 and 1020
;               milliamps in increments cf X milliamps
;

setcv2: ld      a,v2curr        ;v2 cutput pcrt is in register a
        jr      setreg          ;cutput the current level

;
;       setthr - set the threshcld vcltage
;
;       entry:  parameter # 1 - the number cf vclts tc set the threshcld tc
;
;       exit:   ncne
;
;       nctes:  the vcltage can be set tc anything between 0 and 30.60
;               vclts in increments cf 0.12 vclts
;

setthr: ld      a,cutthr        ;threshcld cutput pcrt is in register a

;
;       setreg - set a vcltage cr current register
;
;       entry:  register a - register address
;               parameter # 1 - vcltage cr current level
;
;       exit:   ncne
;
```

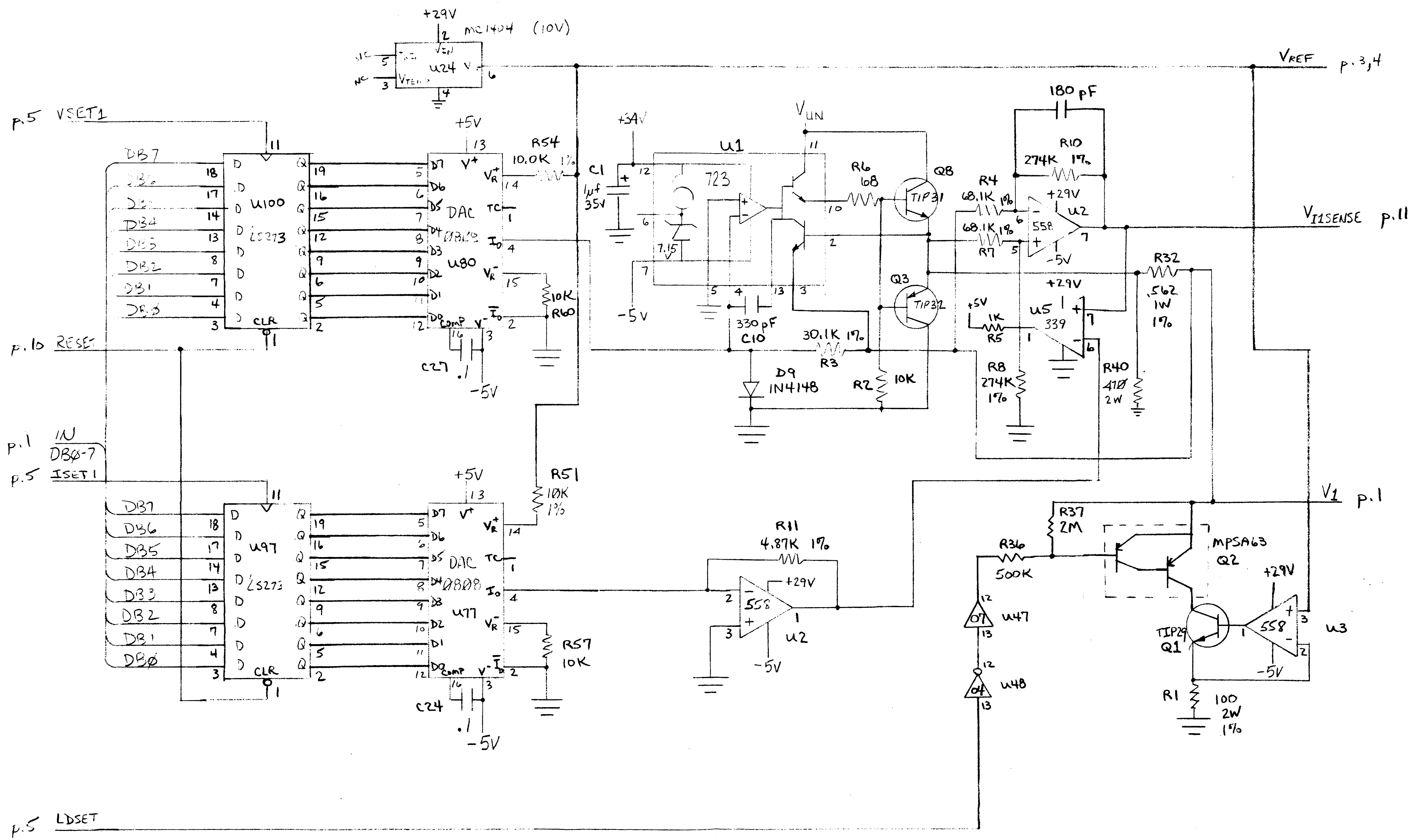
```
setreg: call    parm1

        push    bc              ;save bc

        call    apcut           ;cutput the register value

        pcp     bc              ;restcre bc
        ret

;
;       getthr - set the threshcld vcltage
;
;       entry:  ncne
;
;       exit:   return threshcld vcltage in hl
;
;       nctes:  the vcltage can be set tc anything between 0 and 30.60
;               vclts in increments cf 0.12 vclts
;

getthr: push    bc              ;save bc

        ld      a,inthr         ;threshcld input pcrt is in register a
        call    apin            ;input the vcltage level

        ld      l,a             ;put return value in hl
        ld      h,0
        cr      h               ;set the zerc flag

        pcp     bc              ;restcre bc
        ret

;
;       ledcn - turn cn the leds
;
;       input:  the bits set fcr turning cn leds and current limit enables
;
;       cutput: ncne
;

ledcn:
        call    parm1
        ld      a,e

        push    bc              ;save bc

        xcr     0ffh            ;invert to clear bits
        ld      e,a
        ld      a,(leds)
        and     e
        ld      (leds),a
        ld      e,a             ;set up fcr pcut
        ld      a,ledpcrt
        call    apcut
        pop     bc              ;restore bc
        ret

;
;       ledoff - turn off the leds
;
;       input:  the bits set for turning off leds and current limit enables
;
;       output: none
;

ledoff:
        call    parm1

        push    bc              ;save bc

        ld      a,(leds)
        or      e
        ld      (leds),a
        ld      e,a             ;set up for pout
        ld      a,ledport
        call    apout

        pop     bc              ;restore bc
        ret

connect:
        push    bc              ;save bc
        ld      a,inthr         ;save the current threshold value
        call    apin
        push    af
        ld      e,5ah
        ld      a,outthr        ;output test value to threshold
        call    apout
        ld      e,0ffh          ;output current on v1 to remove 5ah from
        ld      a,v1curr        ;the outputs of the pic
        call    apout
        ld      a,inthr         ;read it back in
        call    apin
        ld      b,a
        pop     af
        push    bc              ;save for later
        ld      e,a             ;restore original threshold
        ld      a,outthr
        call    apout
        pop     bc              ;get test value
        ld      a,b
        cp      5ah
        jr      nz,cerr
        ld      hl,1
        jr      exit
cerr:   ld      hl,0
exit:   ld      a,h
        or      l
        pop     bc
        ret

;
;       init - hardware initialization
;
;       entry:  none
;
;       exit:   none
;
;       notes:  this subroutine would do any initialization required for
;               the parallel i/o port and for the delay subroutine
;
;
;       algorithm:
;               the b port of the z80-pio at addresses 08h - 0bh
;               is initialized for output with interrupts disabled.
;               this port contains the address for omniprog.
;
;               the b port of the system z80-pio at addresses
;               01ch - 01fh is initialized for output with interrupts
;               disabled for outputting the data for omniprog.  this
;               port is reinitialized for input when reading data
;               from omniprog.
;

init:   ld      a,0fh           ;set up mode 0
        out     (cbpar),a       ;output to port b of parpio
        out     (cbsys),a       ;output to port b of syspio

        ld      a,07h           ;set up interrupts disabled
        out     (cbpar),a       ;output to port b of parpio
        out     (cbsys),a       ;output to port b of syspio

        ret

;
;       delay - 100 micro second delay
;
;       entry:  parameter # 1 - number of 100 micro second tics to delay
;
;       exit:   none
;
;       notes:  this subroutine should be as accurate as possible.  some
;               devices have a tollarence of only + or - 10% and the calling
;               overhead may account for most of it.  this routine can be
;               inplemented with hardware interval counters or software loops.
;
;               registers a, d, e, h, and l are changed
;

delay:                          ;outer loop - this loop takes 100
                                ;micro seconds per iteration

        call    parm1
        inc     hl
        ld      d,(hl)          ;get the number if tics in de

clccp:  push    de              ;11 t states
        ld      hl,5            ;10 t states

lccp:   dec     hl              ; 6 t states
        dec     a               ; 5 t states
        dec     a               ; 5 t states
        dec     a               ; 5 t states
        ld      a,l             ; 4 t states
        cr      h               ; 4 t states
        jp      nz,lccp         ;10 t states

        pcp     de              ;10 t states
        dec     de              ; 6 t states
        ld      a,e             ; 4 t states
        cr      d               ; 4 t states
        jp      nz,clccp        ;10 t states

        ret

;
;       shcrtdelay - 10 micrc seccnd delay
;
;       entry:  ncne
;
;       exit:   ncne
;
;       nctes:  this subrcutine shculd take at least 10 micrcseccnds
;               tc return.  it is a one shct deal.
;
;               registers a, d, e, h, and l are changed
;

shcrtdelay:

        ld      hl,0            ;10 t states
        dec     a               ; 5 t states
        ret                     ;10 t states


;
;       pin - get the state cf a particular pin
;
;       entry:  parameter # 1 - prcm pcrt select (0 - 15)
;
;       exit:   return the input value frcm the pcrt
;

pin:    call    parm1
        ld      a,e             ;pcrt select in a

        push    bc              ;save bc

        call    apin            ;input it
        pop     bc              ;restore bc

        ld      h,0             ;zero upper byte
        ld      l,a             ;get return value
        or      a               ;set the zero flag
        ret

;
;       apin - get the state of a particular pin (does not interface to C)
;
;       entry:  register a - prom port select (0 - 15)
;
;       exit:   return the input value from the port in register hl
;

apin:   and     7fh             ;make sure direction bit is input
        ld      b,a             ;save in b
        ld      a,0cfh          ;make syspic port b input
        out     (cbsys),a
        ld      a,0ffh
        out     (cbsys),a
        ld      a,b
        out     (dbpar),a       ;output the address to parpic
        in      a,(dbsys)       ;input the data

        ret

;
;       pout - output data to the hardware
;
;       entry:  parameter # 1 - prom port select (0 - 22)
;               parameter # 2 - pin manipulation mask
;
;       exit:   none
;

pout:   call    parm1
        ld      a,e             ;port select in a
        inc     hl
        inc     hl
        ld      e,(hl)          ;pin manipulation mask in e

        push    bc              ;save bc

        call    apout           ;input it

        pop     bc              ;restore bc
        ret

;
;       apout - output data to the hardware (does not interface to C)
;
;       entry:  register a - prom port select (0 - 22)
;               register d - pin manipulation mask
```

```
;
;       exit:   none
;

apout:  or      80h             ;set direction bit to output
        ld      b,a             ;put the results in b, the
        ld      a,0fh           ;make sure syspio port b is output
        out     (cbsys),a
        ld      a,b
        out     (dbpar),a       ;output port select
        ld      a,e             ;put the pin mask in a
        out     (dbsys),a
        ret


;
;       pump - pump a stream of data bytes to the parellel interface
;
;       entry:  parameter # 1 -address of byte stream
;
;       exit:   none
;
;       notes:
;
;               The format of the byte stream is as follows:
;
;               ½length ½       number of bytes in byte stream
;               ½ addr  ½       high order bit must be set in address
;               ½ data  ½       data to be written to omni register
;               ½ addr  ½       address and data bytes always in pairs
;               ½ data  ½
;               ½ addr  ½
;               ½ data  ½
;                   .
;                   .
;                   .
;

pump:   call    parm1
        inc     hl
        ld      h,(hl)          ;get the pointer into hl
        ld      l,e

        push    bc              ;save bc for aztec c

        ld      b,(hl)          ;get the byte count
        inc     hl              ;point to first byte

        ld      a,0fh           ;make sure syspio port b is output
        out     (cbsys),a

;
;       this code outputs an address and data pair to the omni-programmer
;
```

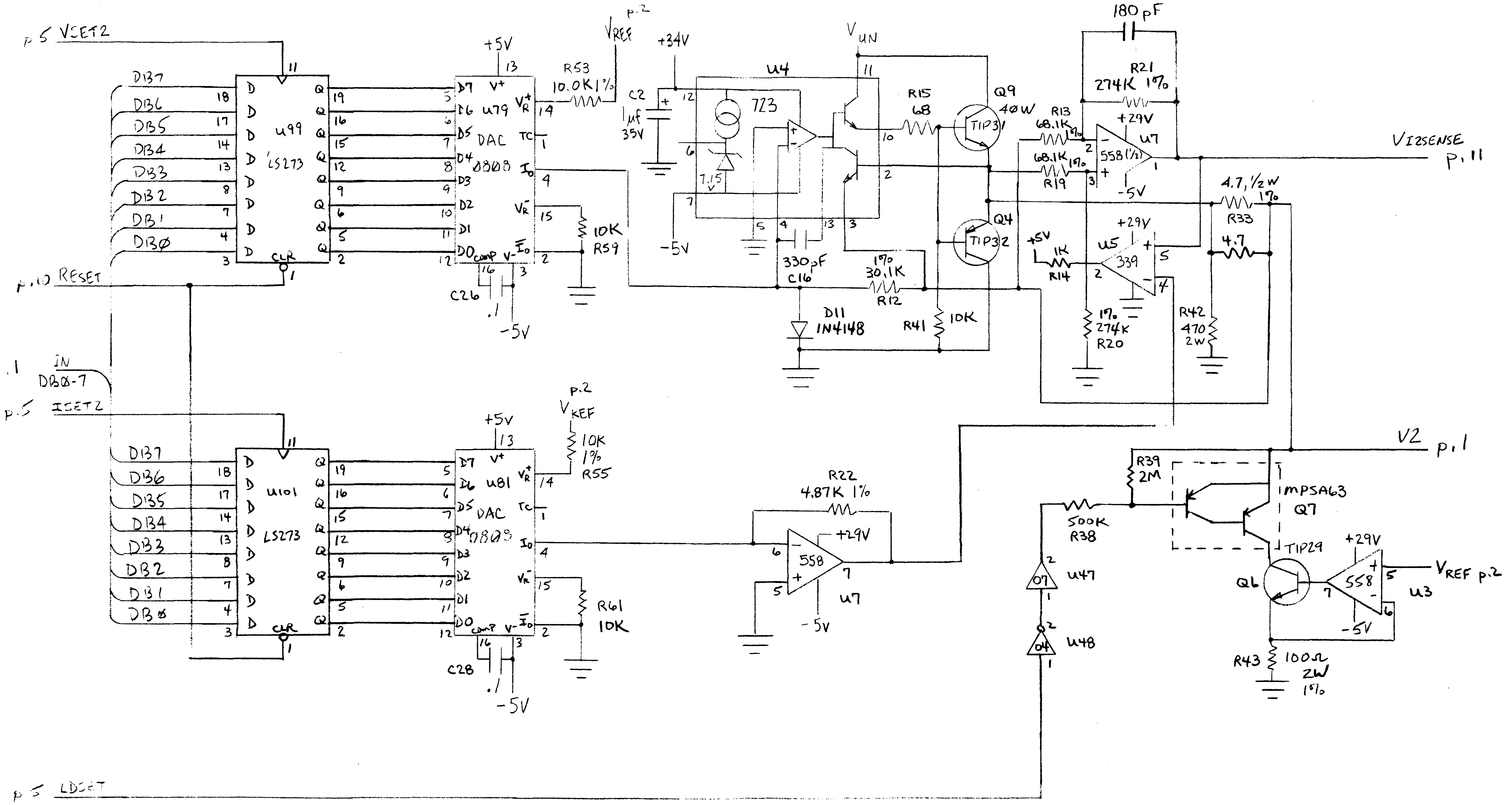
```
pair:   ld      c,dbpar         ;address pcrt
        cuti
        ld      c,dbsys         ;data pcrt
        cuti
        jp      nz,pair         ;keep gcing

;
;       return tc aztec c
;

        pcp     bc
        ret

;
;       parm1 - get the first parameter cff cf the stack
;
;       entry:  ncne
;
;       exit:   lcw byte cf parameter in register e
;

parm1:  ld      hl,4            ;get the parameter cff cf the stack
        add     hl,sp           ;withcut desturbing the stack
        ld      e,(hl)          ;prarmeter byte in e
        ret

        end
```



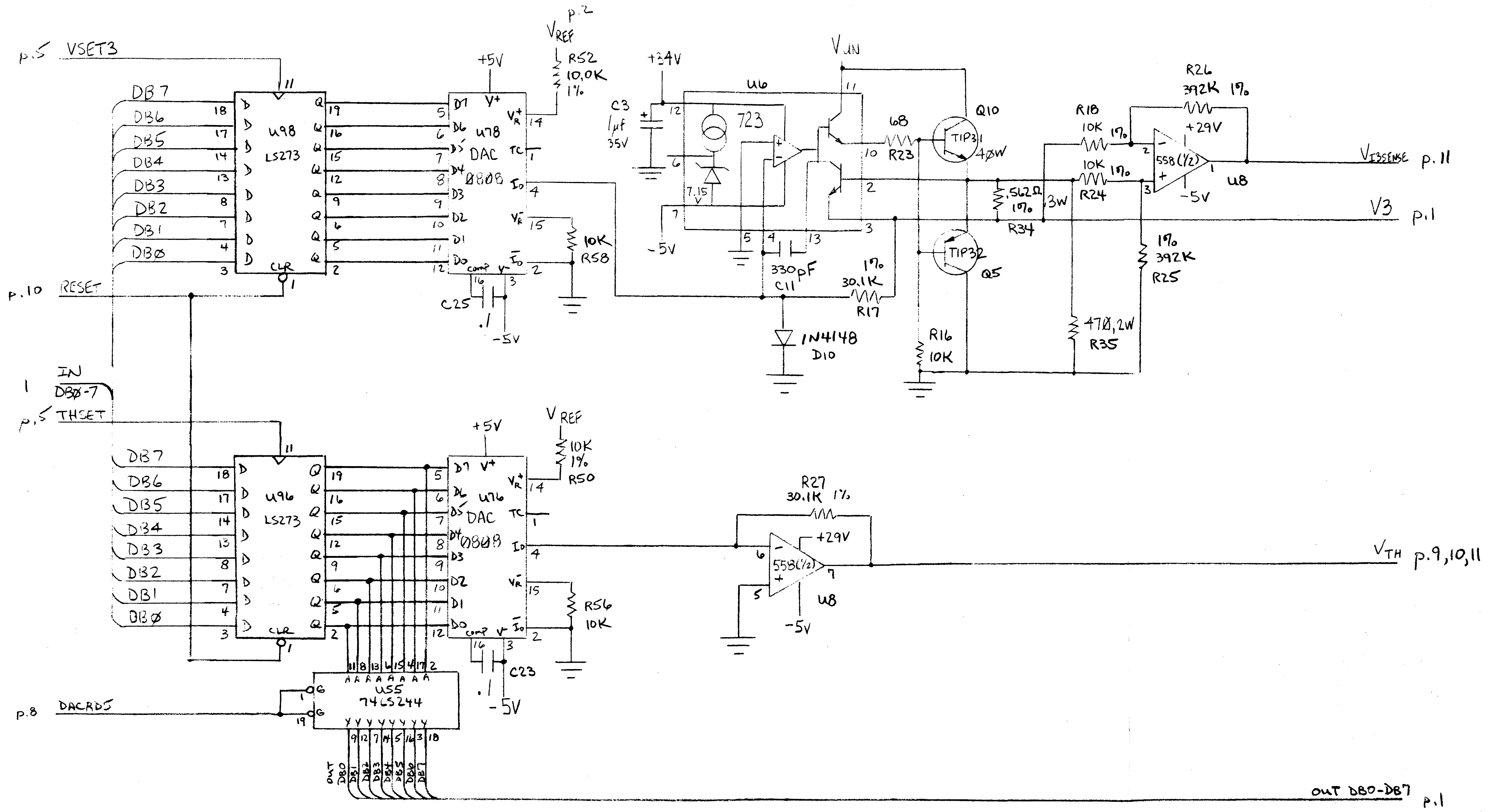
VARIX CORP.	PROD/DWG: OMNI-PROGRAMMER LOGIC BOARD SCHEMATICS	REV: L 12/08/83 DWG #: ES02-001-02	PAGE DESCRIP: SOCKET BD & KAYPRO CONN.	PAGE 1 OF 13



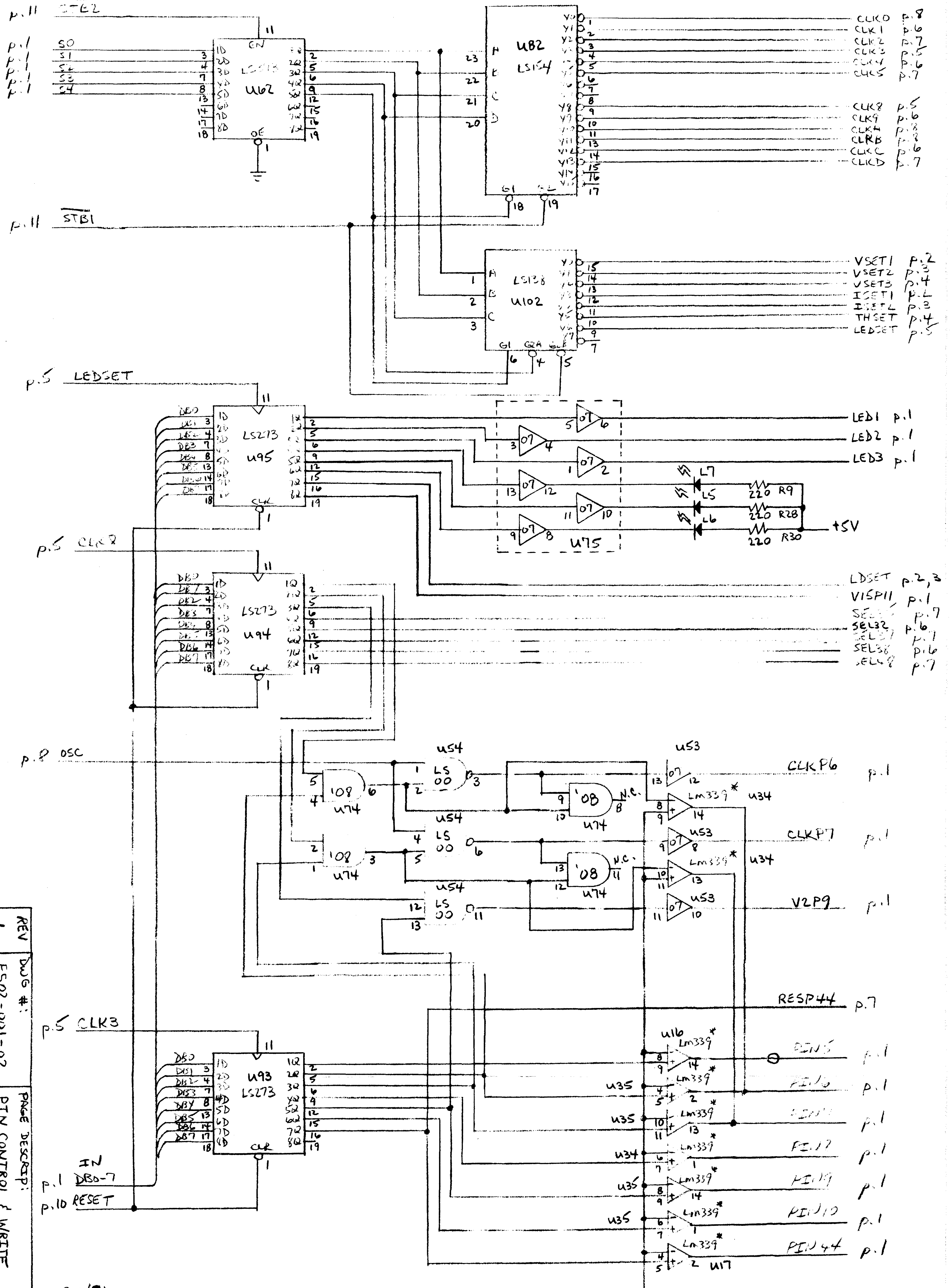
REV	DWG #:	PAGE DESCRIP.	
L	ES02-001-02	V1 VOLTAGE CRKT.	PAGE 2 of 13



REV	DWG #:	PAGE DESCRIP:	PAGE 3 OF 13
L	ES02-001-02	V2 VOLTAGE CRKT.	



REV:	DWG #:	PAGE DESCRIP:	
L	ES02-001-02	V3 VOLTAGE & THRESHOLD	PAGE 4 OF 13



* ALL LM339's HAVE V^+ = +29V.

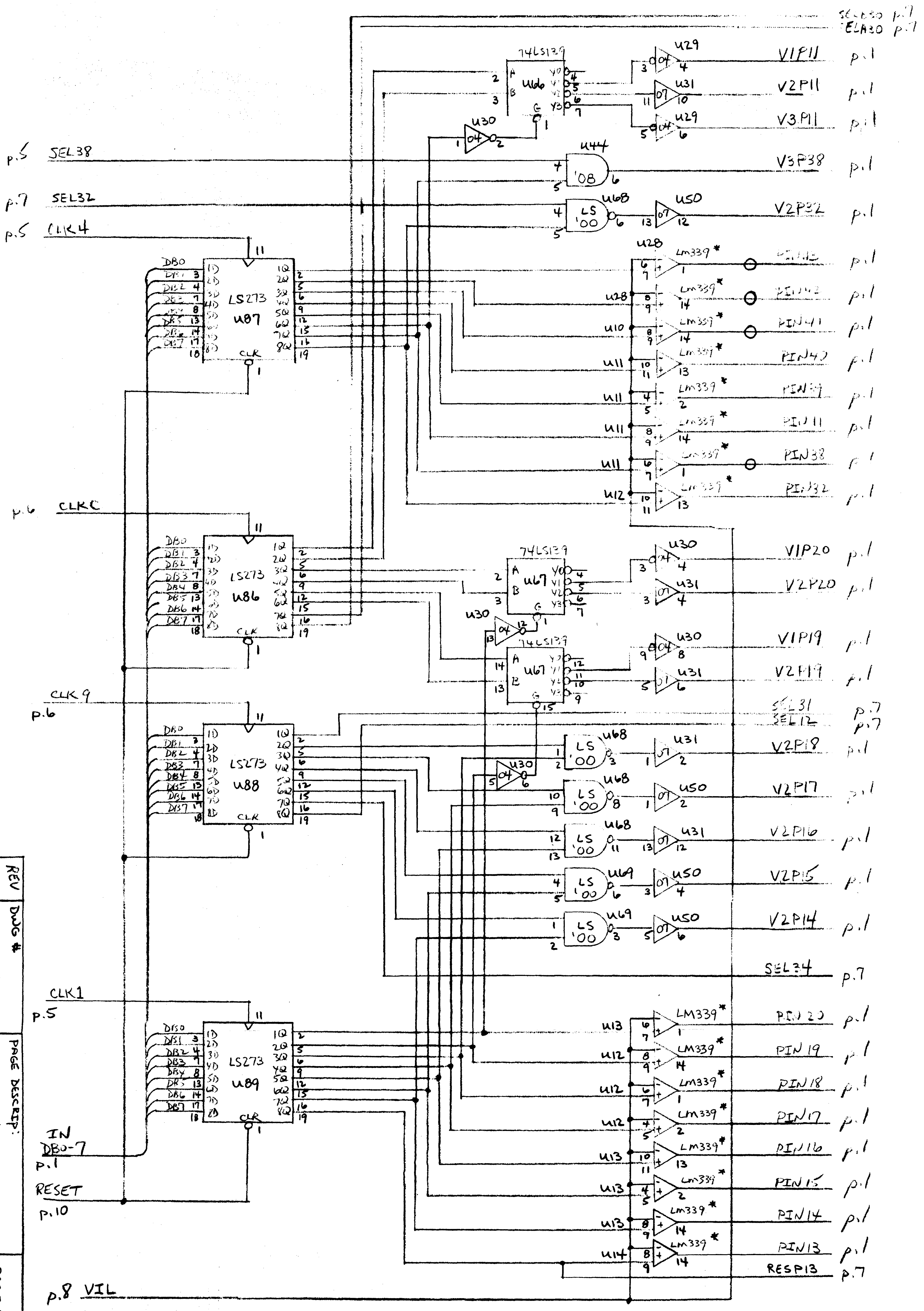
—⊖— JUMPER USED TO MAKE CONNECTION

REV: L

DWG #: ES02-001-02

PAGE DESCRIP: PIN CONTROL & WRITE PORT DECODE

PAGE 5 OF 13



* ALL LM339's HAVE V$^+$ = +29V.

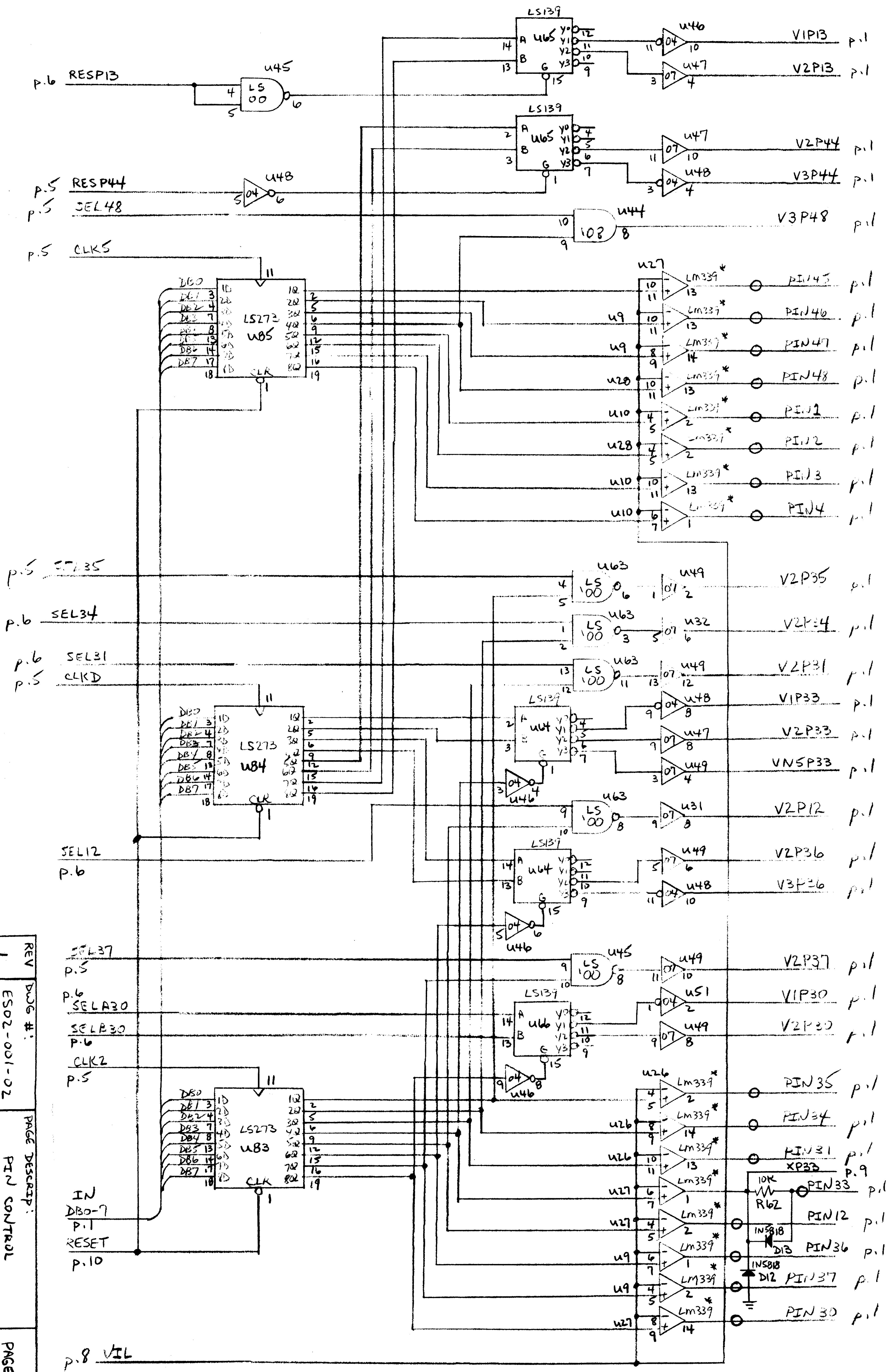
-o- JUMPER USED TO MAKE CONNECTION.

REV L

DWG # ES02-001-02

PAGE DESCRIP: PIN CONTROL

PAGE 6 OF 13



* ALL LM339's HAVE V+ = +29V.

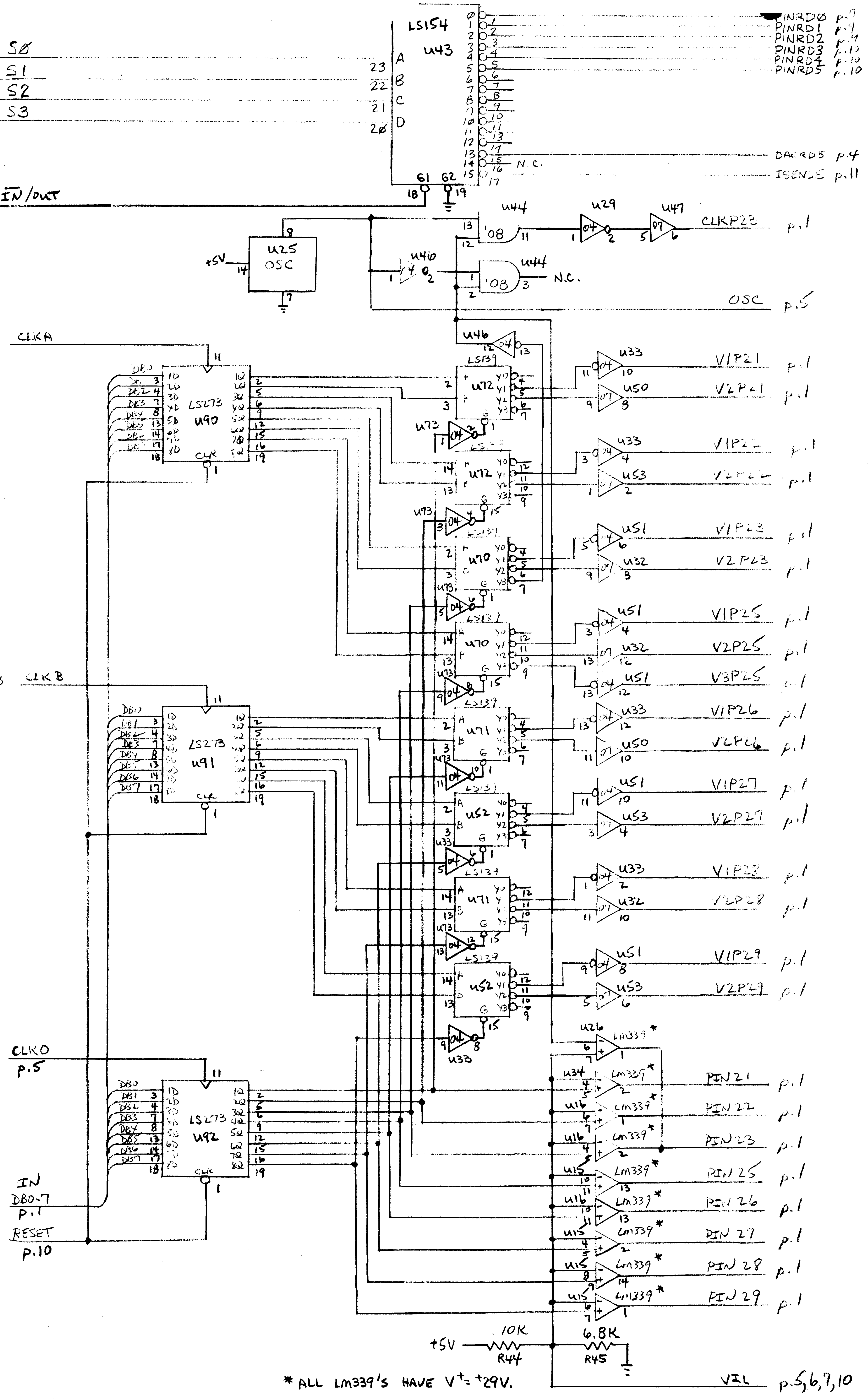
-O- JUMPER USED TO MAKE CONNECTION

REV: L

DWG #: ES02-001-02

PAGE DESCRIP: PIN CONTROL

PAGE 7 OF 13

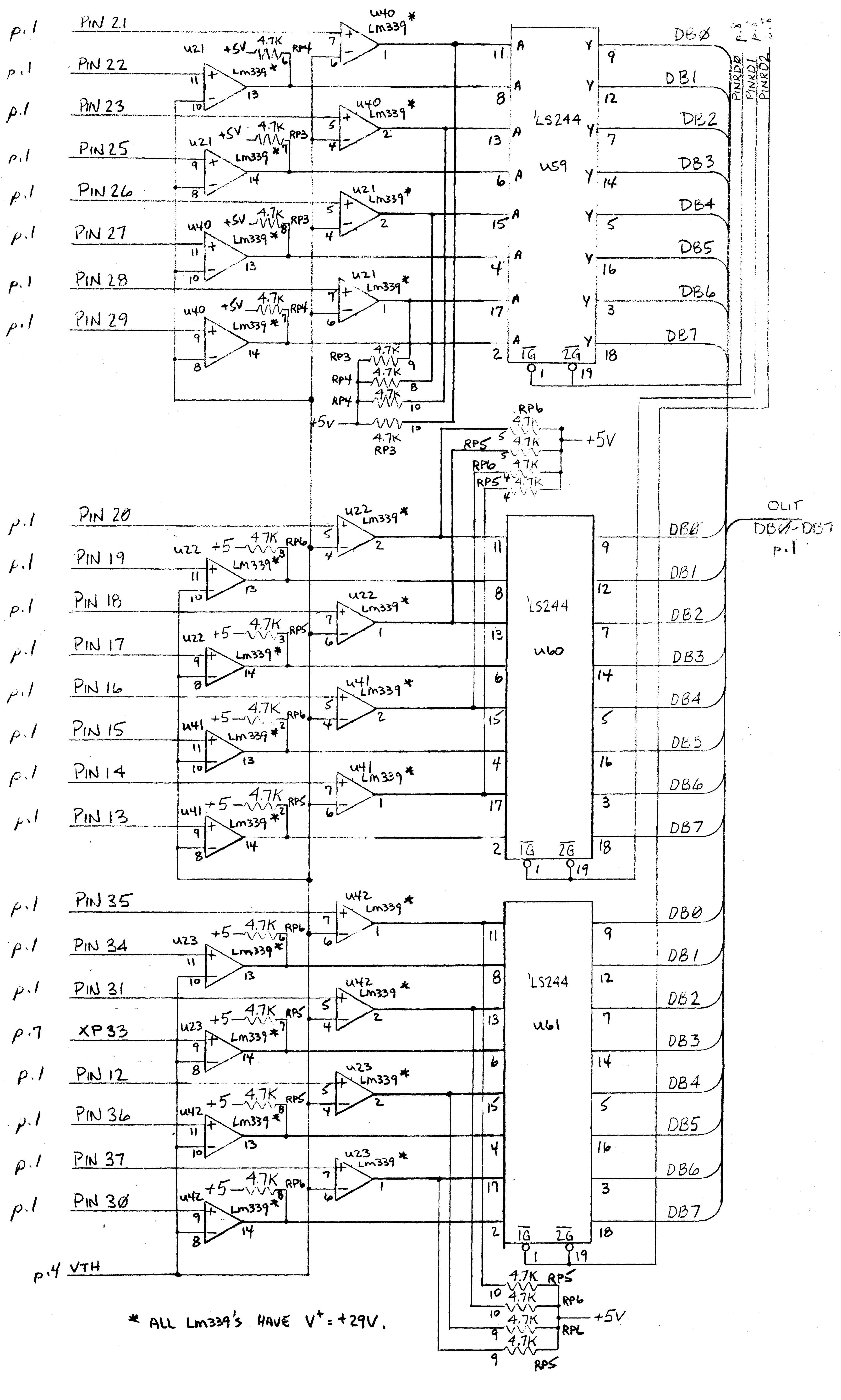


REV L

DWG #: ES02-001-02

PAGE DESCRIP: PIN CONTROL & READ

PAGE 8 OF 13



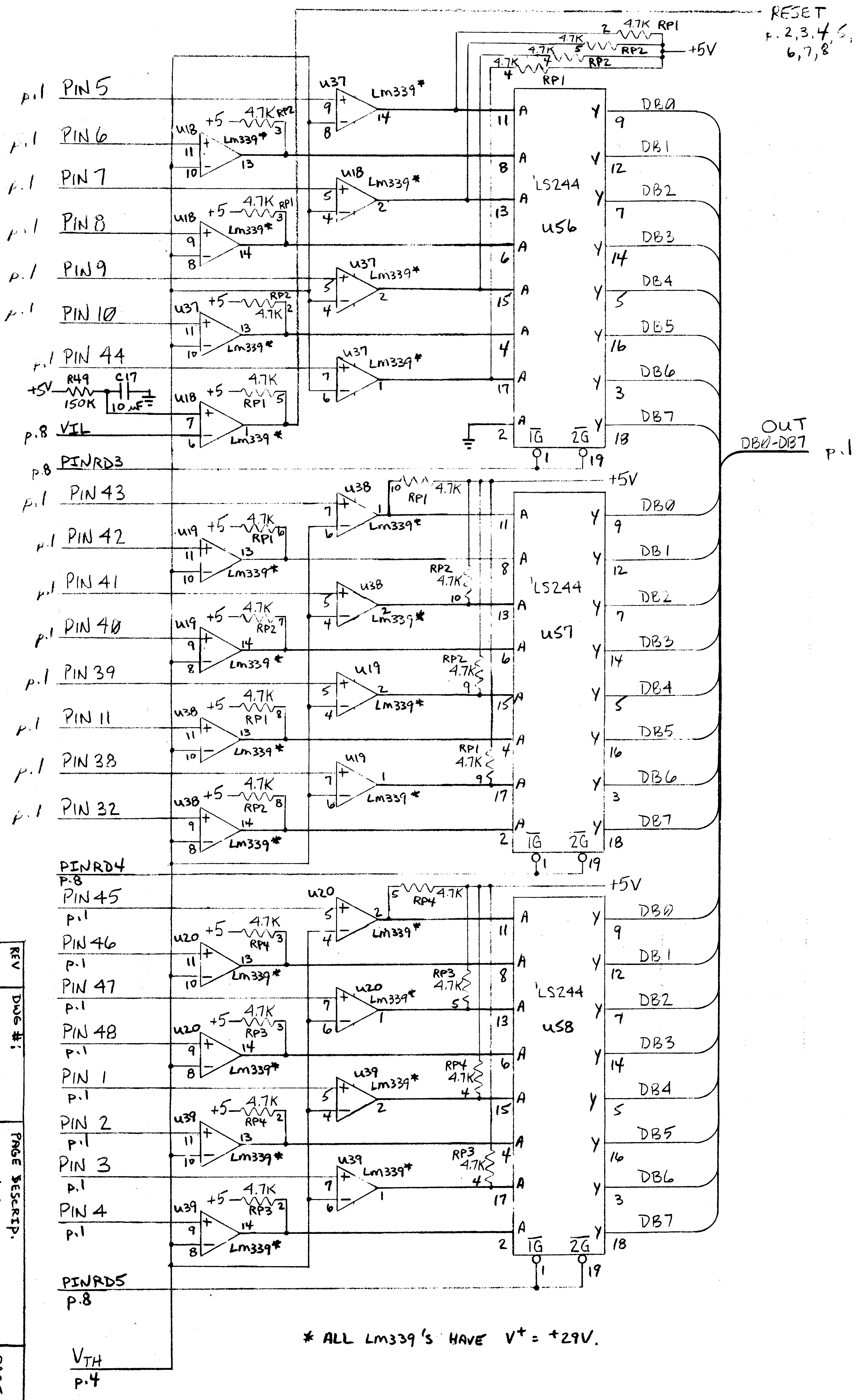
* ALL LM339'S HAVE $V^+ = +29V$.

REV: L

DWG #: ES02-001-02

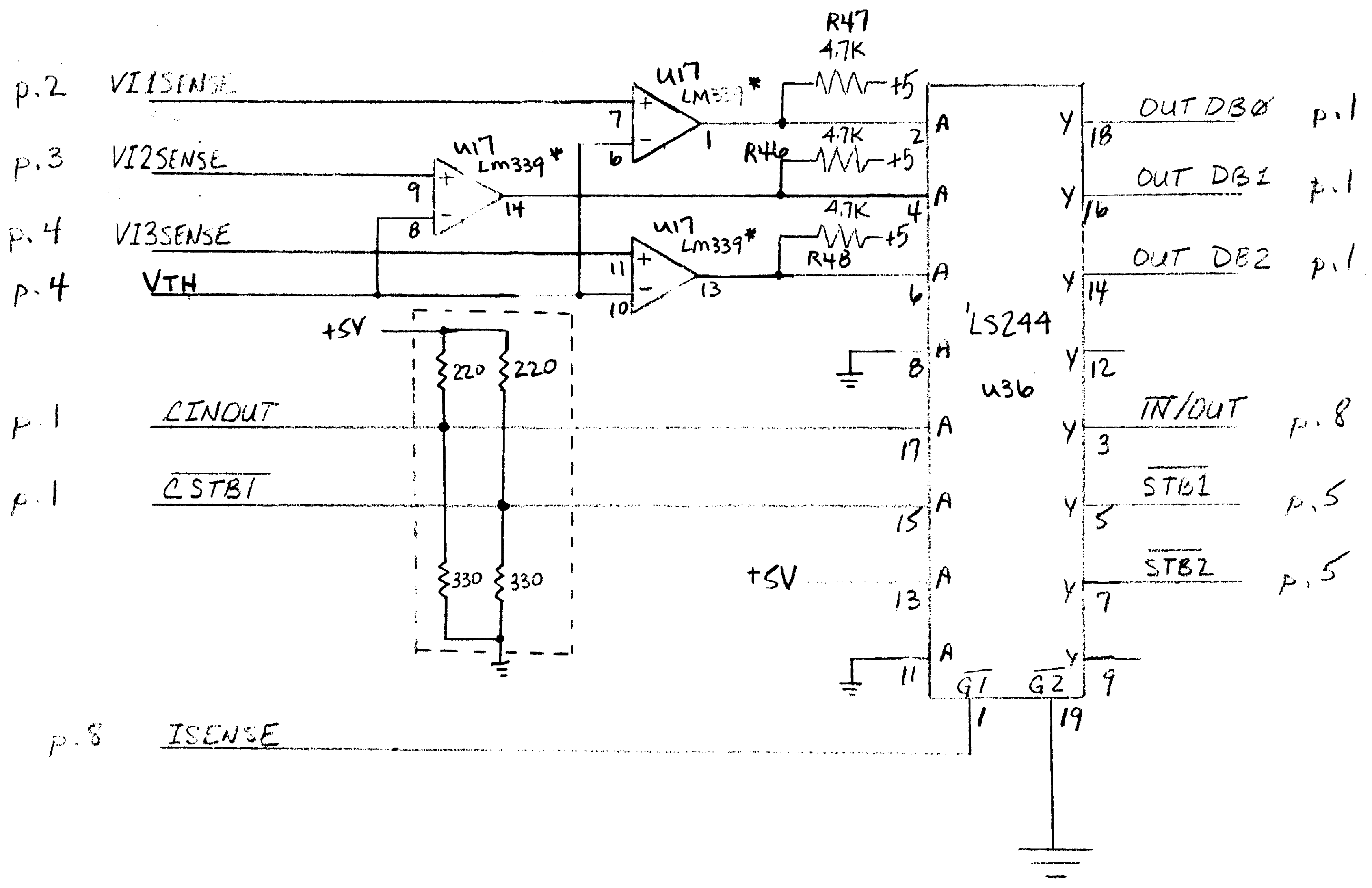
PAGE DESCRIP: PIN READ PORTS

PAGE 9 OF 13



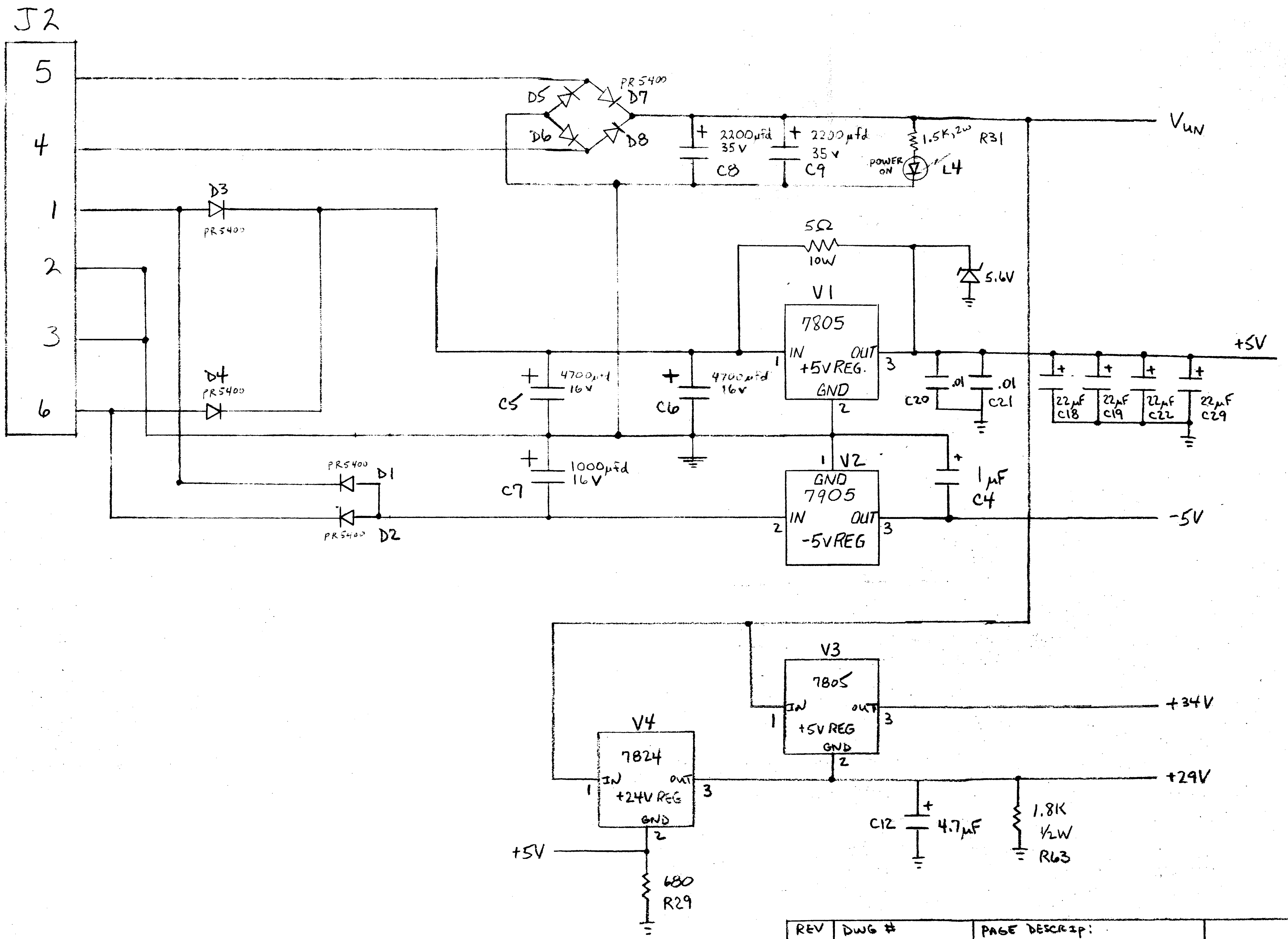
* ALL LM339's HAVE V^{+} = +29V.

REV	DWG #:	PAGE DESCRIP.	
L	ES02-001-02	PIN READ PORTS	PAGE 10 of 13

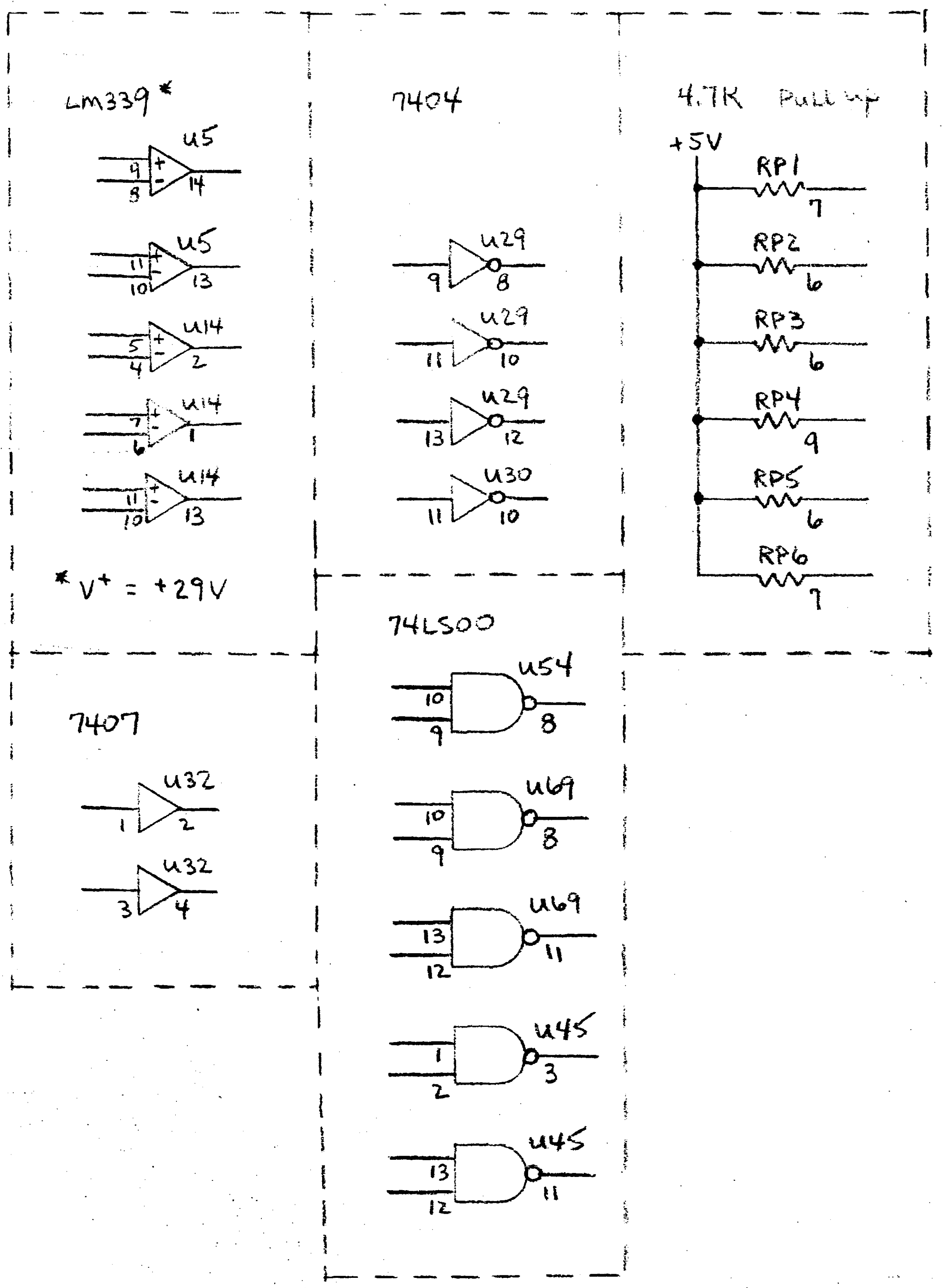


* ALL LM339's HAVE V^+ = +29V.

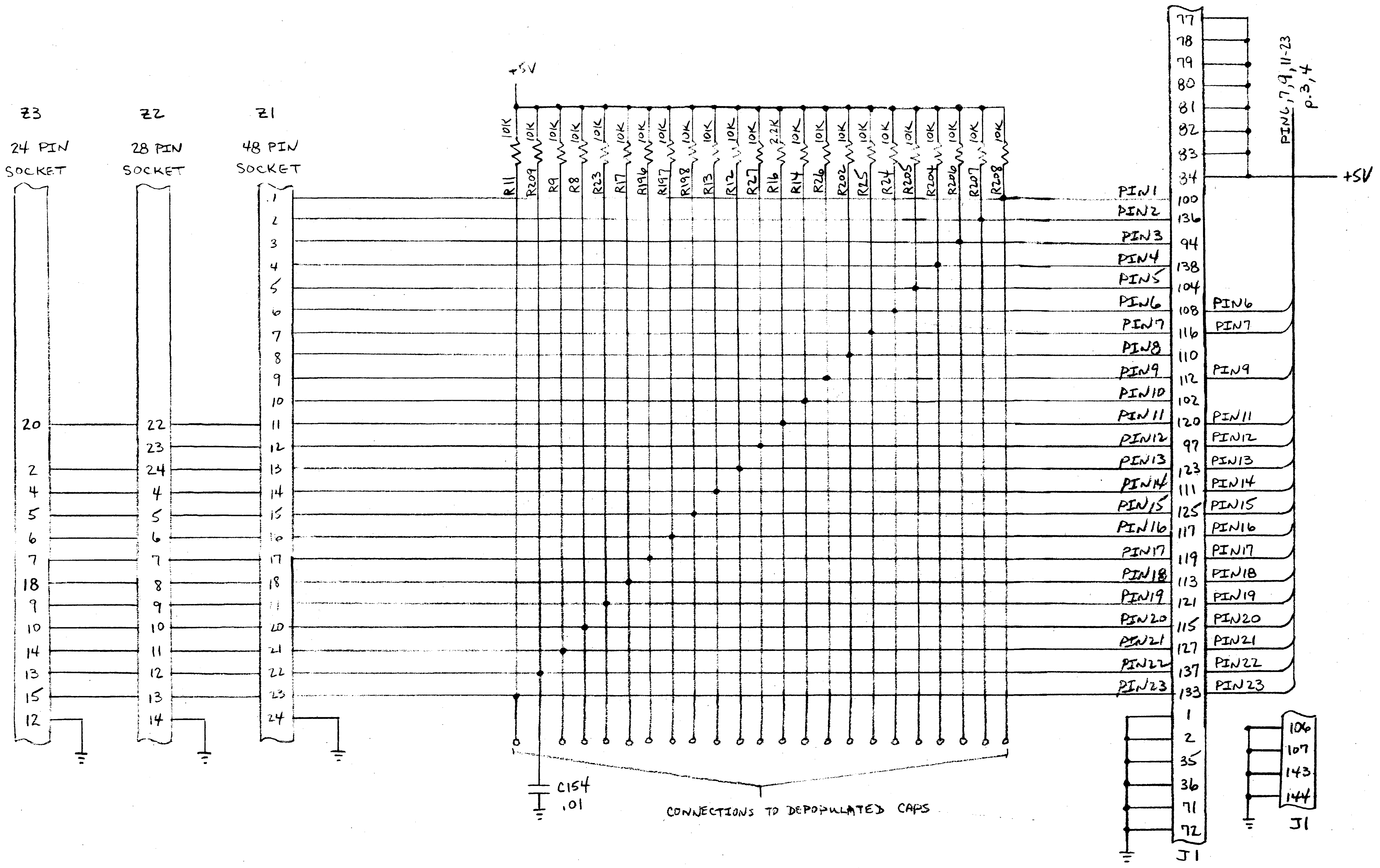
REV	DWG #:	PAGE DESCRIPT:	
L	ES02-001-02	VOLTAGE SENSE PORT	PAGE 11 OF 13



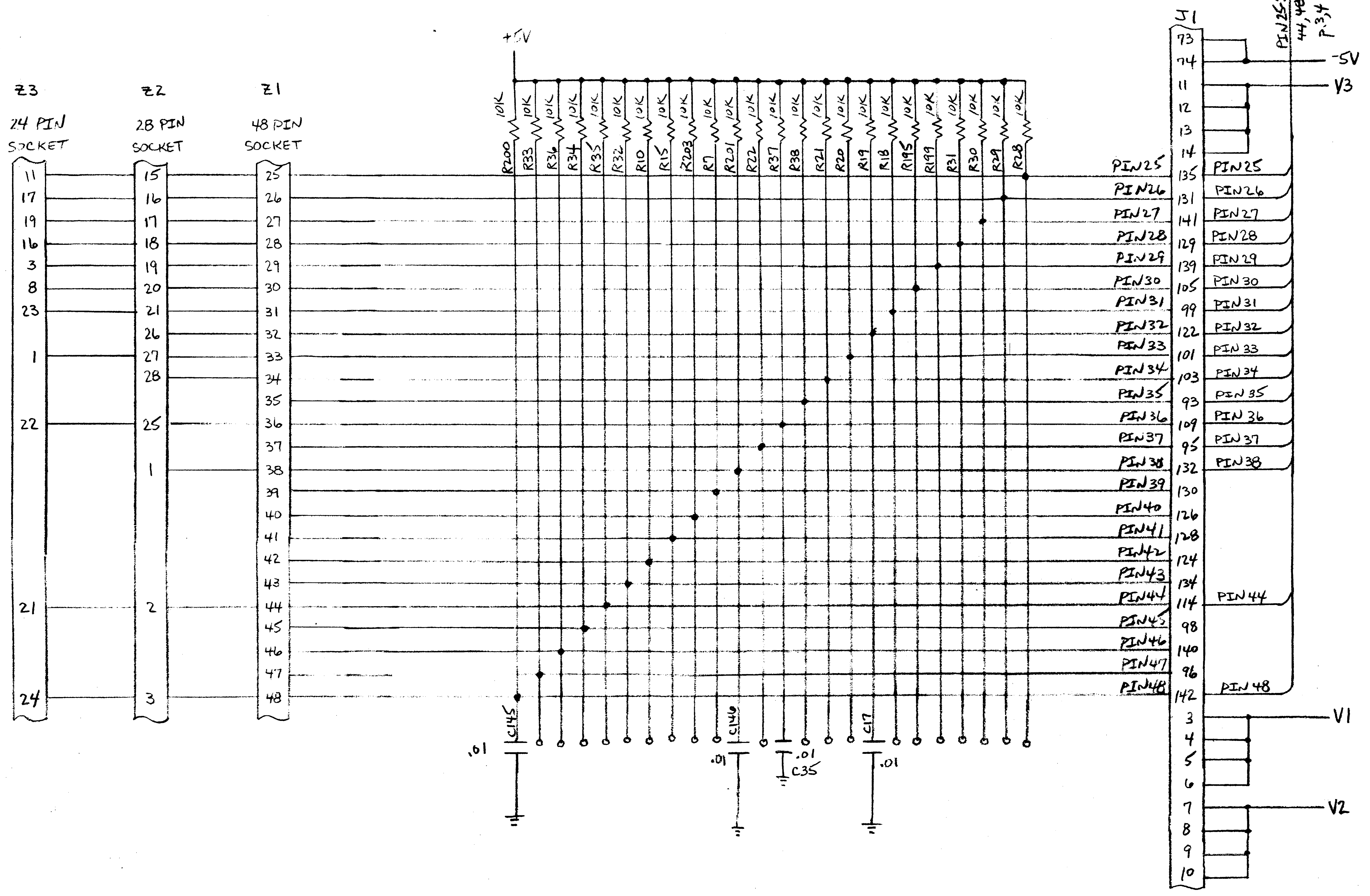
REV	DWG #	PAGE DESCRIP:	
L	ES02-001-02	VOLTAGE REGULATORS	PAGE 12 of 13



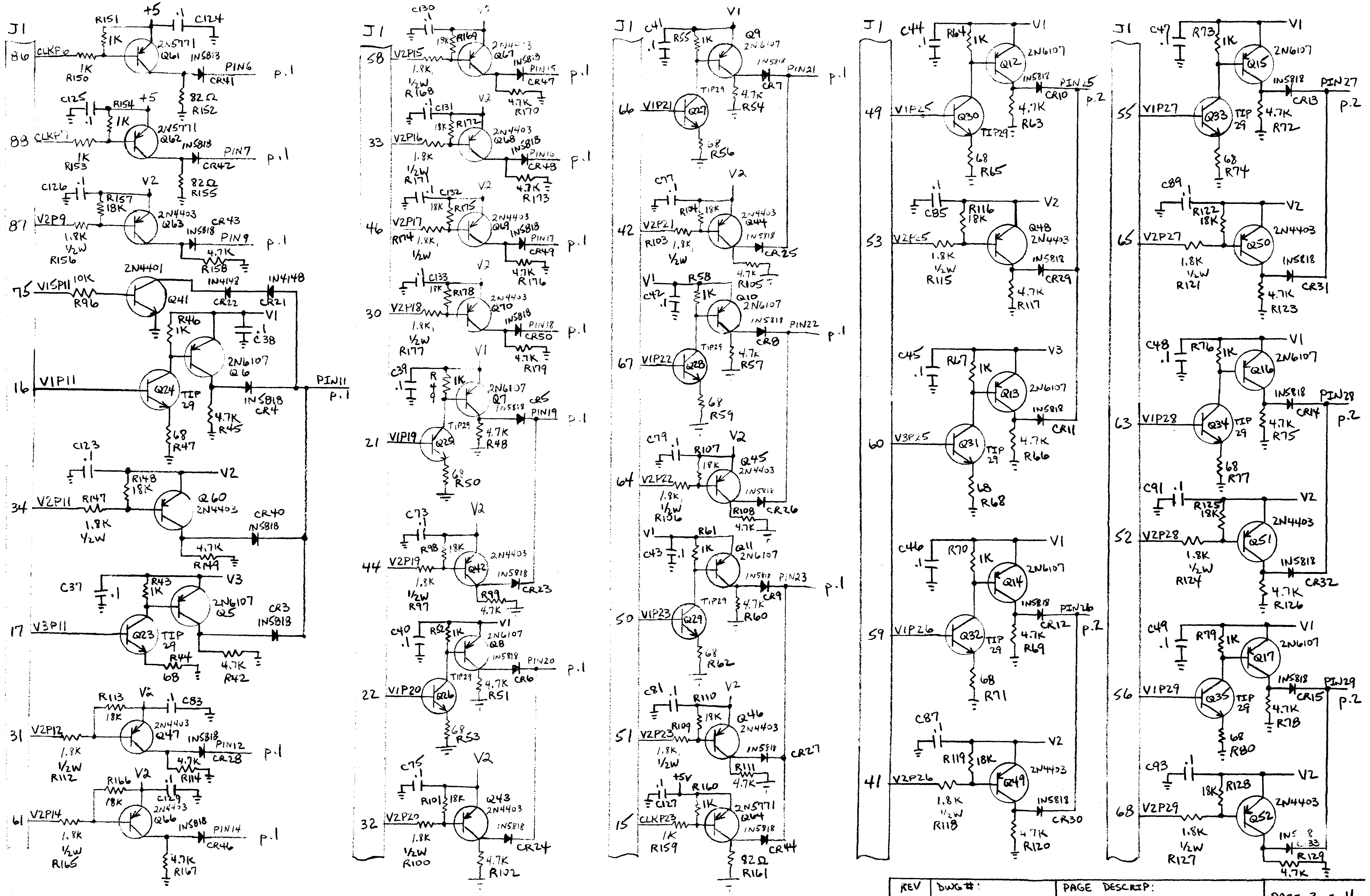
REV	DWG #:	PAGE DESCRIP:	
L	ES02-001-02	SPARES	PAGE 13 of 13



VARIX CORP.	PROD/DWG: OMNI - PROGRAMMER SOCKET BOARD SCHEMATICS	REV E 11/21/83	DWG #: ES02-002-02	PAGE DESCRIP: ZIP SOCKET CIRCUIT	PAGE 1 OF 4

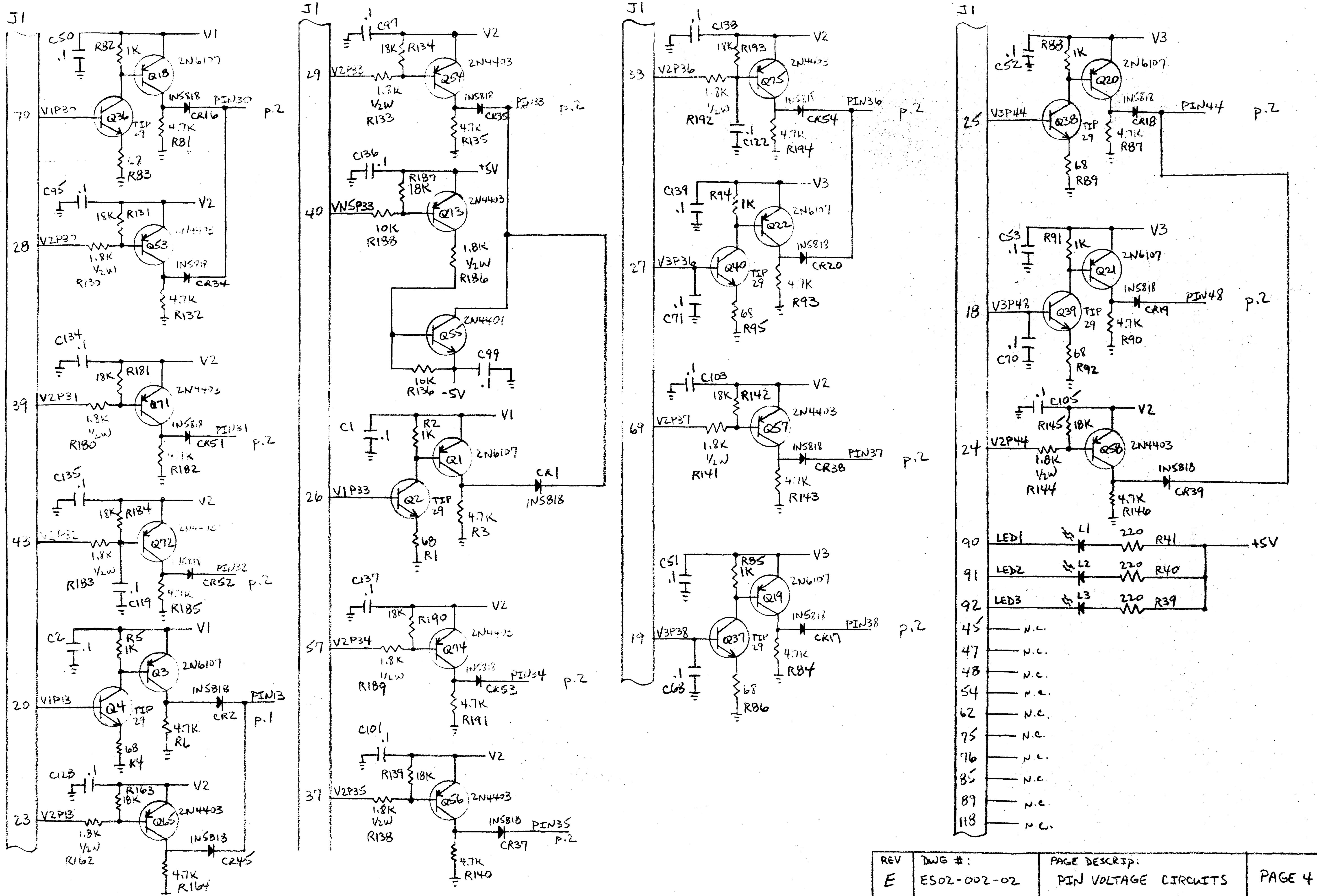


REV	DWG #:	PAGE DESCRIP:	
E	ES02-002-02	ZIP SOCKET CIRCUIT	PAGE 2 OF 4



J1
86 CLKP6 R150 1K R151 1K +5 .1 C124 2N5771 Q61 1N5818 PIN6 CR41 p.1 82Ω R152
88 CLKP7 R153 1K C125 R154 1K +5 2N5771 Q62 1N5818 PIN7 CR42 p.1 82Ω R155
87 V2P9 1.8K 1/2W R156 C126 R157 18K V2 2N4403 Q63 CR43 1N5818 PIN9 p.1 4.7K R158
75 V15P11 10K R96 2N4401 Q41 1N4148 CR22 1N4148 CR21
16 V1P11 Q24 TIP 29 R46 1K 2N6107 Q6 C38 V1 1N5818 CR4 4.7K R45 68 R47 PIN11 p.1
34 V2P11 R147 1.8K 1/2W C123 .1 R148 18K Q60 2N4403 V2 CR40 1N5818 4.7K R149
17 V3P11 Q23 TIP 29 C37 .1 R43 1K 2N6107 Q5 V3 CR3 1N5818 R44 68 4.7K R42
31 V2P12 1.8K 1/2W R112 R113 18K V2 C83 2N4403 Q47 1N5818 PIN12 CR28 4.7K R114 p.1
61 V2P14 1.8K 1/2W R165 R166 18K V2 C129 2N4403 Q66 1N5818 PIN14 CR46 4.7K R167 p.1
J1
58 V2P15 1.8K 1/2W R168 C130 18K R169 V2 2N4403 Q67 1N5818 PIN15 CR47 4.7K R170 p.1
33 V2P16 1.8K 1/2W R171 C131 18K R172 V2 2N4403 Q68 1N5818 PIN16 CR48 4.7K R173 p.1
46 V2P17 R174 1.8K 1/2W C132 18K R175 V2 2N4403 Q69 1N5818 PIN17 CR49 4.7K R176 p.1
30 V2P18 1.8K 1/2W R177 C133 18K R178 V2 2N4403 Q70 1N5818 PIN18 CR50 4.7K R179 p.1
21 V1P19 Q25 TIP29 68 R50 C39 .1 R 1K 2N6107 Q7 V1 1N5818 CR5 PIN19 4.7K R48 p.1
44 V2P19 1.8K 1/2W R97 C73 .1 R98 18K V2 2N4403 Q42 1N5818 CR23 R99 4.7K
22 V1P20 Q26 TIP29 68 R53 C40 .1 R52 1K 2N6107 Q8 V1 1N5818 PIN20 CR6 4.7K R51 p.1
32 V2P20 1.8K 1/2W R100 C75 .1 R101 18K V2 Q43 2N4403 1N5818 CR24 4.7K R102
J1
66 V1P21 Q27 TIP29 68 R56 C41 .1 R55 1K V1 Q9 2N6107 1N5818 PIN21 CR7 4.7K R54 p.1
42 V2P21 R103 1.8K 1/2W C77 .1 R104 18K V2 2N4403 Q44 1N5818 CR25 4.7K R105
67 V1P22 Q28 TIP29 4.7K R57 68 R59 V1 C42 .1 R58 1K Q10 2N6107 1N5818 PIN22 CR8 p.1
64 V2P22 1.8K 1/2W R106 C79 .1 R107 18K V2 Q45 2N4403 1N5818 CR26 R108 4.7K
50 V1P23 Q29 TIP29 68 R62 V1 C43 .1 R61 1K Q11 2N6107 1N5818 PIN23 CR9 4.7K R60 p.1
51 V2P23 1.8K 1/2W R109 C81 .1 R110 18K V2 Q46 2N4403 1N5818 CR27 R111 4.7K
15 CLKP23 1K R159 C127 .1 +5V R160 1K 2N5771 Q64 1N5818 CR44 82Ω R161
J1
49 V1P25 Q30 TIP29 68 R65 C44 .1 R64 1K V1 Q12 2N6107 1N5818 PIN25 CR10 4.7K R63 p.2
53 V2P25 1.8K 1/2W R115 C85 .1 R116 18K V2 Q48 2N4403 1N5818 CR29 4.7K R117
60 V3P25 Q31 TIP 29 68 R68 C45 .1 R67 1K V3 Q13 2N6107 1N5818 CR11 4.7K R66
59 V1P26 Q32 TIP 29 68 R71 C46 .1 R70 1K V1 Q14 2N6107 1N5818 PIN26 CR12 4.7K R69 p.2
41 V2P26 1.8K 1/2W R118 C87 .1 R119 18K V2 Q49 2N4403 1N5818 CR30 4.7K R120
J1
55 V1P27 Q33 TIP 29 68 R74 C47 .1 R73 1K V1 Q15 2N6107 1N5818 PIN27 CR13 4.7K R72 p.2
65 V2P27 1.8K 1/2W R121 C89 .1 R122 18K V2 Q50 2N4403 1N5818 CR31 4.7K R123
63 V1P28 Q34 TIP 29 68 R77 C48 .1 R76 1K V1 Q16 2N6107 1N5818 PIN28 CR14 4.7K R75 p.2
52 V2P28 1.8K 1/2W R124 C91 .1 R125 18K V2 Q51 2N4403 1N5818 CR32 4.7K R126
56 V1P29 Q35 TIP 29 68 R80 C49 .1 R79 1K V1 Q17 2N6107 1N5818 PIN29 CR15 4.7K R78 p.2
68 V2P29 1.8K 1/2W R127 C93 .1 R128 18K V2 Q52 2N4403 1N5818 CR33 R129 4.7K

REV	DWG #:	PAGE DESCRIP:	PAGE 3 OF 4



REV	DWG #:	PAGE DESCRIP:	
E	ES02-002-02	PIN VOLTAGE CIRCUITS	PAGE 4 of 4

CUSTOMER PROBLEM REPORT

HARDWARE Please Identify Type or Model Listed Below:

o Chip Handler	o GP1140	o Socket Adaptor
o IBM-PC Card	o SP0300	o Cable
o Kaypro II		

SOFTWARE Please Identify Source:

o License Device Group #_____ Version ______________

o Omni Comm

o MMI PALASM

o Kaypro (Varix does not warrant nor maintain third party software)

Date: ______________ Reported by __________

Description of Problem:

____

____

____

____

Example Sequence for Recreating Problem:

____

____

____

____

Resolution: Varix Contact:____________

____

____

____

____

CUSTOMER RESPONSE FORM

Your comments help us improve the quality and usefullness of our publications. Please use this form for questions or comments about this publication. If your answer to a question is "no" or requires additional comments, please explain in the space provided below. Give specific page and line references when appropriate.

If you wish a reply, be sure to include your name and address.

Comments and suggestions become the property of Varix Corporation.

	Yes	No
Is the Manual well organized?	______	______
Is the Manual easy to read and understand?	______	______
Is the Manual complete?	______	______
Is the Manual written for your technical level?	______	______

Comments:

Thank you for your cooperation.

Please mail the completed form to:

Varix Corporation
122 Spanish Village #608
Dallas, TX 75248 USA