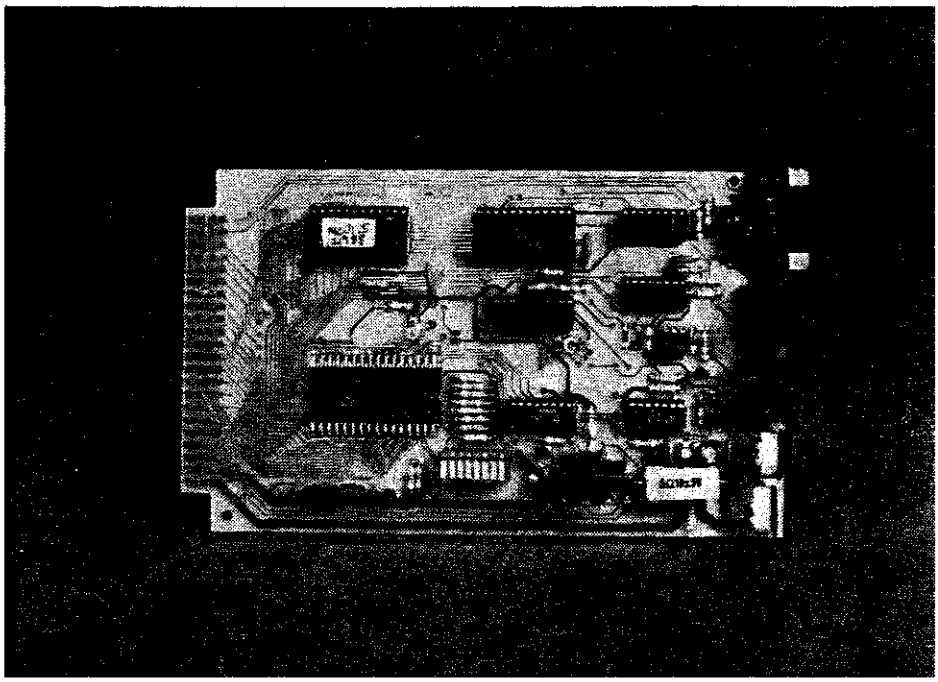
PAIA Electronics, Inc.
3200 Teakwood Lane
Edmond, OK 73013
(405) 340-6300



MCVI

MIDI - CONTROL VOLTAGE INTERFACE / CPU

Operation Manual

MCVI

MIDI - CONTROL VOLTAGE INTERFACE/CPU

The MCVI/CPU board/kit as supplied is a dedicated MIDI to Control Voltage/Trigger and/or Control Voltage/Trigger to MIDI converter. A Control Voltage and Trigger (Gate or S-trigger) applied to the MCVI/CPU will generate a corresponding MIDI Output and/or a MIDI Input to the MCVI/CPU will generate a corresponding Control Voltage and Trigger (Gate or S-trigger). MIDI Transmit and Receive channels are selectable 1 - 16.

The CPU part of the MCVI/CPU is transparent to the user of the board as a converter, however, hackers can access the CPU and Monitor ROM via an RS-232 port. Expansion lines are provided at the board's edge for peripheral development and experimentation.

3200 Teakwood Ln., Edmond, OK 73013 (405) 340-6300

FEATURES	SPECS
Converter-	
¥Linear Control Voltage Input/Output	0 - 10 VDC
¥S-Trigger/Gate Input/Output	5 - 15 VDC Gate Input 5 VDC Gate Output 0 VDC S-Trigger Output
¥MIDI Input	Channel 1 - 16
¥MIDI Output	Channel 1 - 16
CPU-	
¥6511AQ microprocessor chip	4 MHz
¥RAM	192 byte (on 6511AQ)
¥ROM (SmplMON)	
¥User I/O	Parallel Eight Bit RS-232 (3-line TTL level)
¥Expansion Connector	22/44 @ .156
¥A/D	Eight Bit
¥D/A	Eight Bit
¥ACIA	MIDI I/O
¥Counter/Timer	
¥External Power Required	+10 to 15 VDC @ 400 mA -10 to 15 VDC @ 12 mA

ATTENTON

ITEMS YOU MAY NEED FOR SET-UP AND INSTALLATION

There are some other items that you may need which are not provided with the MCVI/CPU. You might want to consider gathering them together prior to installation. Of course, you will need solder (60/40 RESIN core for electronic printed circuits), a soldering iron (25-40 Watt pencil type), and some diagonal cutters as a minimal arsenal of tools.

The MCVI/CPU requires an external power source such as a regulated 10 to 15 Volt bi-polar power supply (Paia 7700, Paia 4771, or equiv.). If you are retrofitting the MCVI/CPU to a synthesizer, or some other powered device, it may be possible to tap into it's supply.

Because of the wide variety of possible applications, wire is not provided for making connections between the MCVI/CPU and external equipment. The power supply and Trigger Input/Output wires should be 18-22 ga. insulated wires and the Control Voltage Input/Output wires should be shielded cable (RG-174 or similar).

A DIP Header is provided for jumpering connections which determine operation characteristics. If you plan to use the MCVI/CPU in a wide variety of applications, it may be desirable for you to obtain and install an eight position DIP switch in place of the DIP Header.

A 9 pin "D" connector can be installed at the S1 socket position to provided access to the "RS-232" lines.

A 22/44 pin .156 spacing card edge connector is recommended for wiring purposes.

SET-UP and USE

The MCVI/CPU can be wired internally to a synthesizer that utilizes a linear 1V/oct. control voltage and trigger (gate or S-trigger), or, circuitry can be added external to the board for stand alone use. As a stand alone unit, the MCVI/CPU can serve as a programmable MIDI and/or control voltage/trigger processor.

Synthesizer Retro-fit

Providing that there is space available and the required power supply voltages and current, it would be possible to install the MCVI/CPU in your synthesizer. Power supply requirements for the board are Positive 15VDC @ 400 mA, Ground (0VDC), and Negative 15VDC @ 12 mA. Voltages as low as positive and negative 10VDC can be used but the Control Voltage Input is restricted to a 0 - 5V range. Voltages lower than 10V would restrict the range even more.

Connections to the MCVI/CPU board are made at the 22/44 pin edged connector pins and the MIDI input/output sockets. The board's power inputs should be wired with 18 - 22 ga. insulated wire and separate wires should connect the Analog Ground (ECA) and Digital Ground (ECB) to the power supply's ground. The Control Voltage Input (EC2) and Control Voltage Output (EC1) should be wired with shielded cable to prevent interference from other signals and should terminate to ground on only one end of the shield if the power supply wiring is establishing a ground circuit.

The board should be mounted next to holes in the synthesizer case that will provide access to the MIDI input/output sockets. If this is not possible, it may be necessary to obtain panel mount 5 Pin DIN sockets to wire in the pc mount socket's holes.

Port B Wiring/Switches

The Port B 0 - 7 input lines to the processor serve a dual function. When power is first applied to the MCVI/CPU, or if it is reset, the levels on these lines select the MIDI Transmit and Receive Channels. Any change in the levels from then on selects an offset in semitones of the conversion. This is useful if there is not a tune adjust for aligning the two synthesizer's pitches.

Refer to the PORT B CHANNEL SELECT table and diagram for

the connections required for selecting the Transmit and Receive Channels. An "H" indicates NO connection between the line and ground and an "L" indicates a connection between the line and ground. The DIP Header provided is used to make the connections between the lines and ground when it is plugged into Port B's socket.

If these selections will be changed often, you could obtain and install an eight position DIP switch in Port B's socket, or, build an assortment DIP Headers with various "settings". If the board is mounted inside a synthesizer, it would be worth considering an external means for selection. An external Reset switch could prove beneficial too.

A/D D/A Adjustments

Trimmers R1 and R2 provide scale adjustments for the A/D and D/A converters. R1, the A/D adjust trimmer, affects the Control Voltage to MIDI conversion and R2, the D/A adjust trimmer affects the MIDI to Control Voltage conversion. If only one mode of conversion is being used, it's not necessary to adjust the trimmer for the unused mode.

Connect the synthesizers together via the MCVI/CPU and set them up so you can hear them both playing. The adjustments of R1 and R2 will affect the spacing of the notes played on the converted synthesizer. Rotating the trimmer one way spreads out the notes and rotating the trimmer the other way bunches together the notes.

The goal is to obtain a corresponding note change between the two synthesizers (not necessarily the same note). Go for octaves first, then fine adjust to get the notes in between the octaves.

MCVI/CPU Stand Alone Use

Refer to the schematic for the USER EXPANSION board for the circuitry required for the MCVI/CPU to function as a stand alone unit. The expansion board includes external RAM, RS-232 level shifting, CV/Trigger Input/Output connectors, and a power supply. The expansion board can be wire wrapped/ hard wired on a perfboard w/ 22/44 pin edge connections and parts available from a neighborhood parts store (excluding the 6116 static RAM which is readily available via mail order or other stores).

Referring to the parts placement diagram, there are points which provide the user access to the reset input, PA5 I/O line, and PD7 I/O line (Ro/Id). It is not necessary to

make connections to these points otherwse. The MCVI/CPU is reset when the reset input line s grounded. A momentary contact SPST could be soldered to the reset point and adjacent ground point providing an on-board reset switch.

PA5 access is provided for user applications (counter-timer I/O, I/O flag, etc.).

PD7 access is provided for user applications (status indication, I/O flag, etc.) with connections to an optional indicator LED and its asociated current limiting resistor.

PROGRAMMING TIPS

When the MCVI/CPU is configured for stand alone use as previously described, the board can be programmed from a computer working like a terminal, or a terminal, via the RS-232 port using 8 bit ASCII data @ 4800 baud.

The 65xx family op-codes are the type used for machine language programming of the MCVI/CPU boad.

The development system used for the MCVI/CPU board consisted of an Apple II w/single drive and serial card running an Editor/Assembler/Terminal program, the USER EXPANSION external circuitry, and a MCVI/CPU board.

I/O PORT USE AND CONTROL

Locations:

PORT A - $0000
PORT B - $0001
PORT C - $0002
PORT D - $0003

I/O ports PA, PB, PC, and PD are parallel eight bit type. Their default mode is as inputs. Writing $00 to PA, PB, and PC puts these ports in output mode; $FF returns them to input mode. Setting the Mode Control Register ($0014) Bit 5 to a 1 selects the output mode for PD.

Port B, Port C 0 - 3, PA5, and PD7 are lines available to the user. PA5 has a wiring point on the board and PD7 has a wiring point/space for an LED (Io) and it's associated current limiting resistor (Ro).

INTERRUPTS

Locations:

IRQ Vector - $FFFE-FF ($00F3-F4 when using monitor ROM)
NMI Vector - $FFFA-FB ($00F1-F2 when using monitor ROM)
RST Vector - $FFFC-FD
IRQ Flag Clear - $0010
IRQ Flag Register - $0011
IRQ Enable Register _ $0012

A 1 in a bit position of the IRQ Flag Register indicates an interrupt condition has been met.

A 0 in a bit position of the IRQ Flag Clear resets the corresponding bit of the IRQ Flag Register.

A 1 in a bit position of the IRQ Enable Register enables an interrrupt upon the condition corresponding to the bit. A 0 disables the interrupt.

IRQ Flag Bit	Condition
0	PA0 Positive edge detect.
1	PA1 Positive edge detect.
2	PA2 Negative edge detect.
3	PA3 Negative edge detect.
4	Counter A underflow.
5	Counter B underflow.
6	Serial Port status register state (Rcv.)
7	Serial Port status register state (Tr.)

ACIA USE and CONTROL

Locations:

ACIA Control Register - $8600
ACIA Data Register - $8601

The ACIA must be reset at the start of a program by writing $03 to the ACIA Control Register.

To put the ACIA in MIDI Transmit mode, write $56 to the ACIA Control Register.

A 1 at bit 1 of the ACIA Control Register indicates the transmit buffer is empty.

Writing the transmit data to the ACIA Data Register initiates the transmission.

To put the ACIA in MIDI Receive mode, write $D6 to the ACIA Control Register.

A 1 at bit 2 of the IRQ Flag Register indicates the receive buffer is full; the receive data can be read and the IRQ Clear bit 2 must be reset. If bit 2 of the IRQ Enable Register is set an interrupt would be executed.

A/D CONVERTER USE and CONTROL

Location:

A/D Converter - $8800

To initiate a conversion, write any data to the A/D.

A 1 at bit 3 of the IRQ Flag Register indicates the end of conversion; the A/D can be read and the IRQ Clear bit 3 must be reset. If bit 3 of the IRQ Enable Register is set, an interrupt would be executed.

D/A CONVERTER USE and CONTROL

Location:

D/A Converter - $0003 (Port D)

Put Port D in output mode by setting bit 5 of $0014 to a 1 and write the data to the D/A converter ($0003).

COUNTER/TIMER USE and CONTROL
(Counter/Timer A n/a on MCVI/CPU)

Locations:

Lower Counter B (Read; clears flag)/Lower Latch B (Write) - $001C
Upper Counter B (Read)/Upper Latch B (Write) - $001D
Lower Counter B (Read)/Lower Latch B (Write; clears flag) - $001E

Mode Select ($0014)

Bit 1	Bit 0	Mode
0	0	Interval Timer
0	1	Asymmetric Pulse Generation
1	0	Event Counter
1	1	Retriggerable Interval Timer

Interval Timer

The upper and lower latches are loaded with a value ($0001-FFFF) that will be decremented at the phase 2 clock rate (0.5 uS -32.767 mS).

IRQ Flag Register bit 5 is set when the count decrements from 0.

Asymmetric Pulse Generation

A variable width pulse signal can be output via PA5 with the pulse width determined by the values written to locations $001C and $001D. The duration between pulses is determined by the values written to locations $001C and $001E.

Event Counter

The upper and lower latches are loaded with values that will be decremented with each rising edge on PA5.

Retriggerable Interval Timer

The upper and lower latches are loaded with values that will be decremented at the phase 2 clock rate. A rising edge on PA5 resets the counters to the latch values.

SERIAL I/O - (n/a on MCVI/CPU)

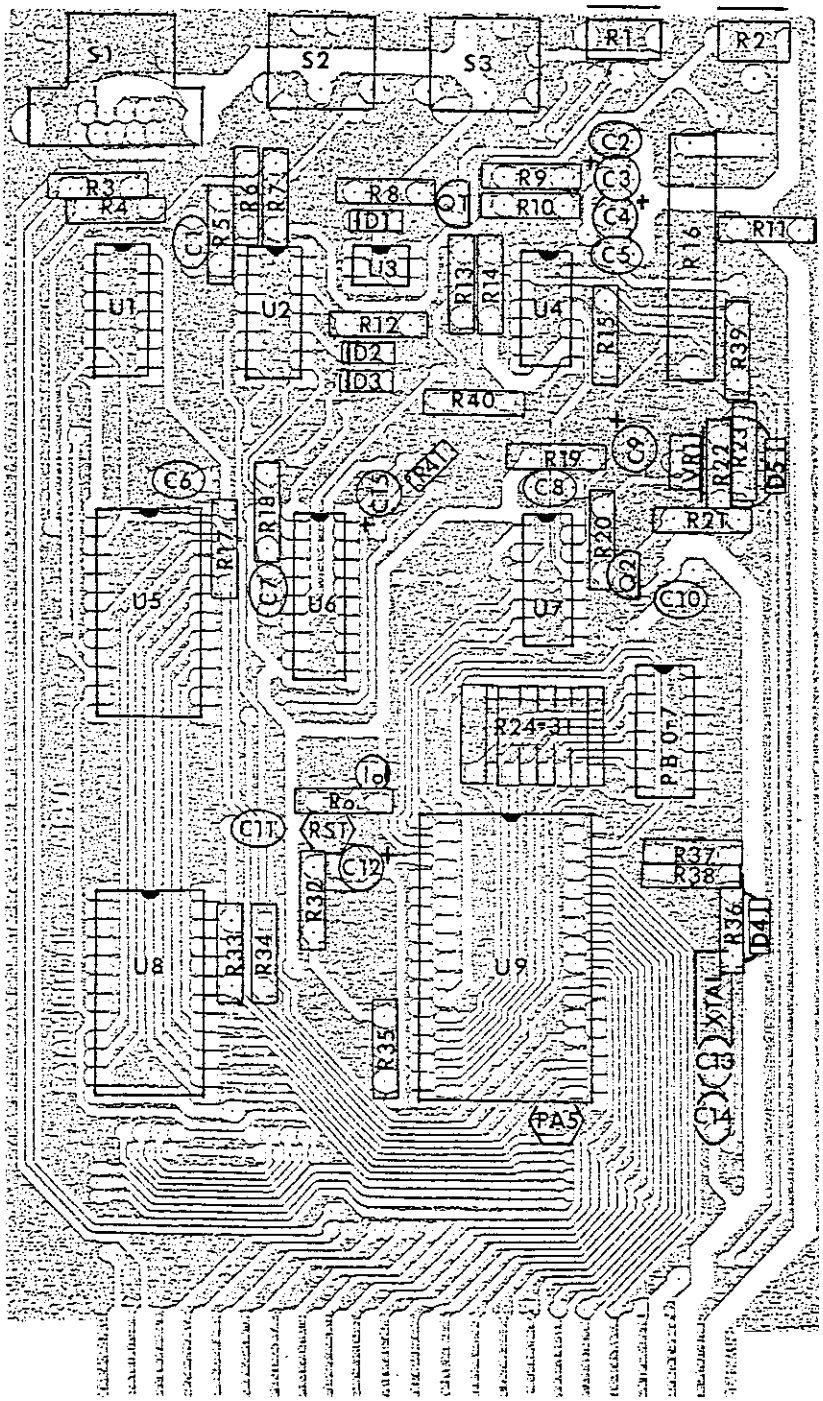
RAM

Locations:

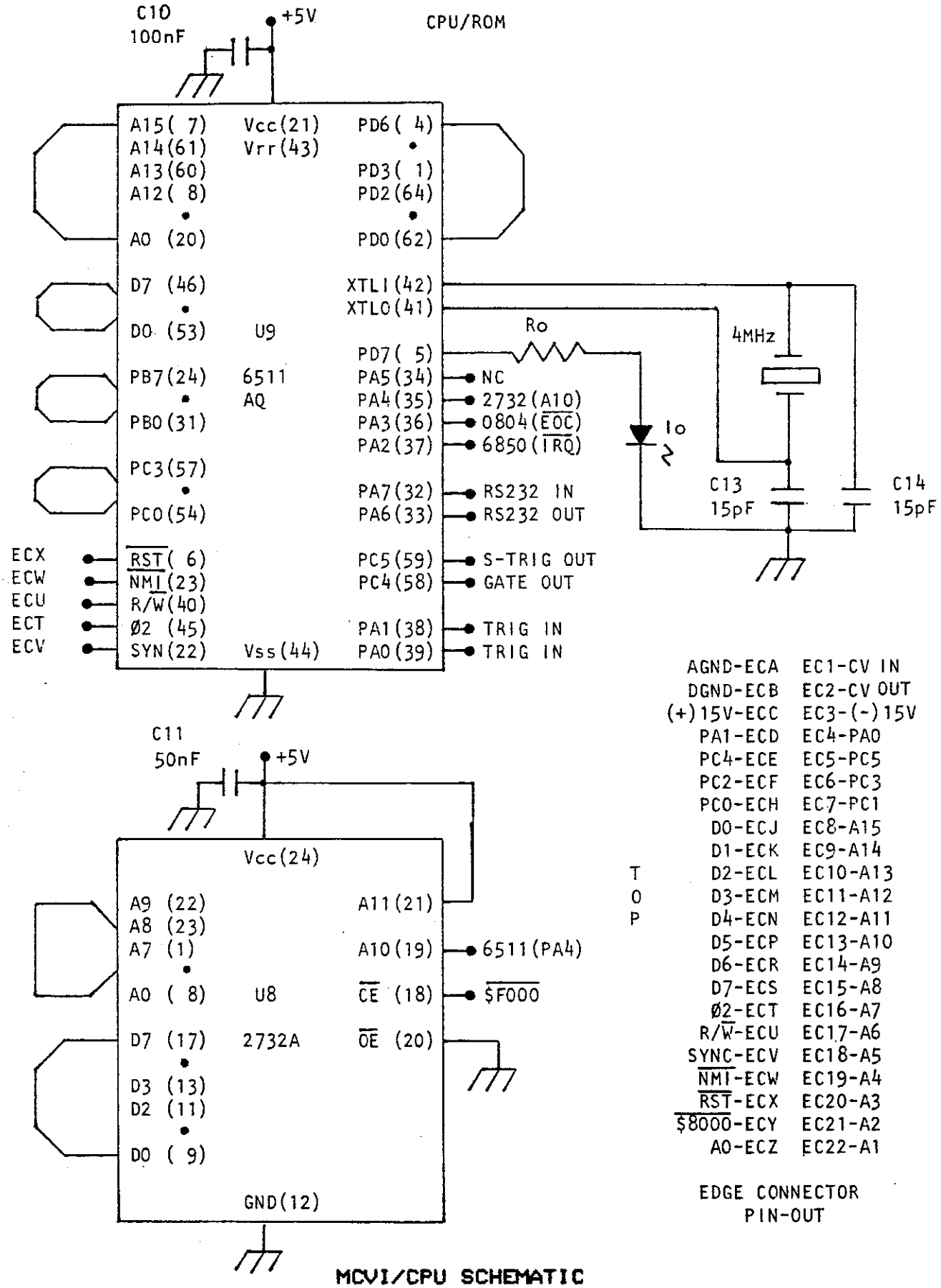
$0041 - 00FF

The vectors and stack use the top part of this RAM which should be taken into account when using these locations.

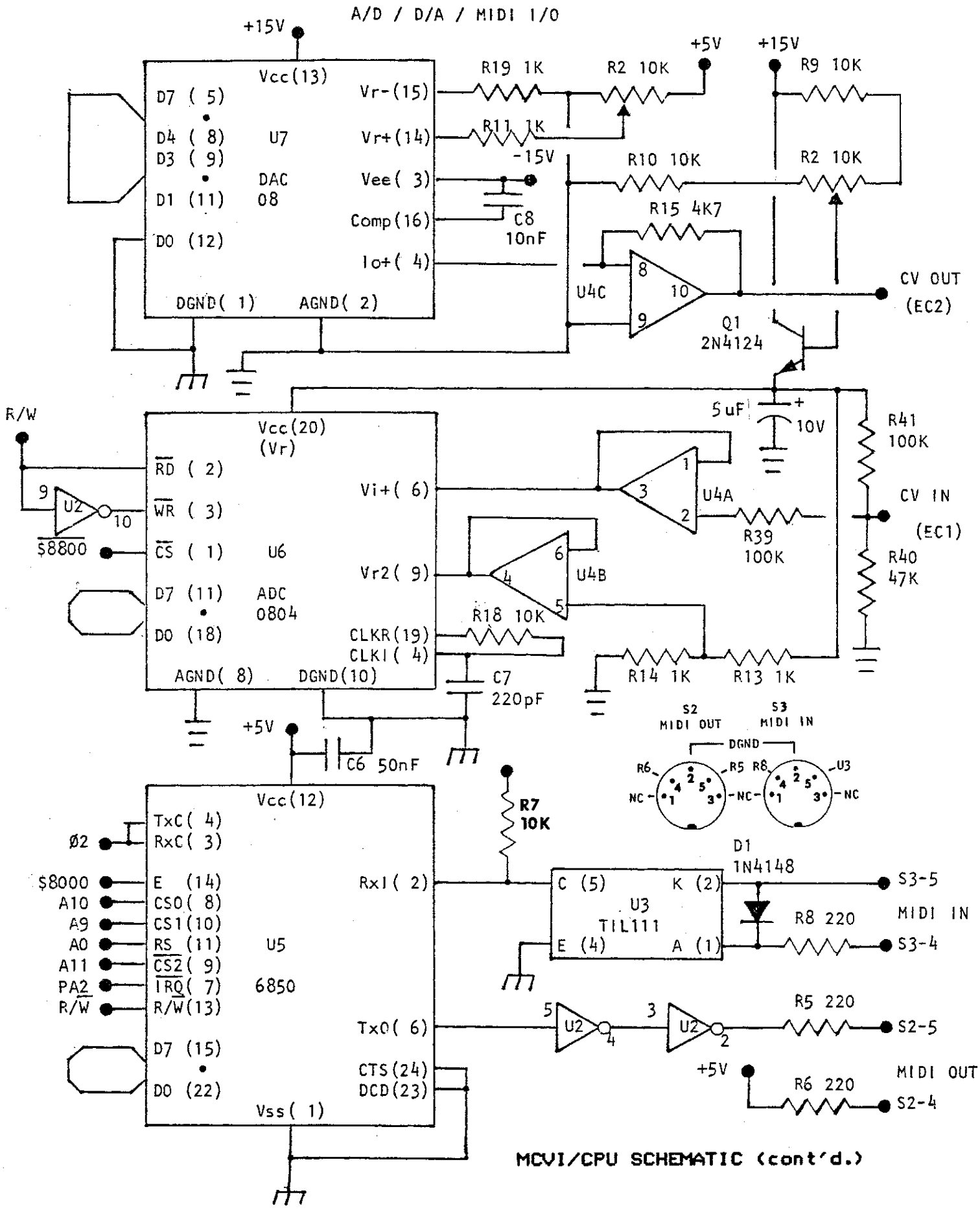
MORE IN-DEPTH PROGRAMMING INFORMATION AVAILABLE
SEE SOURCE LISTED IN BIBLIOGRAPHY
ON PAGE 23 OF THIS MANUAL



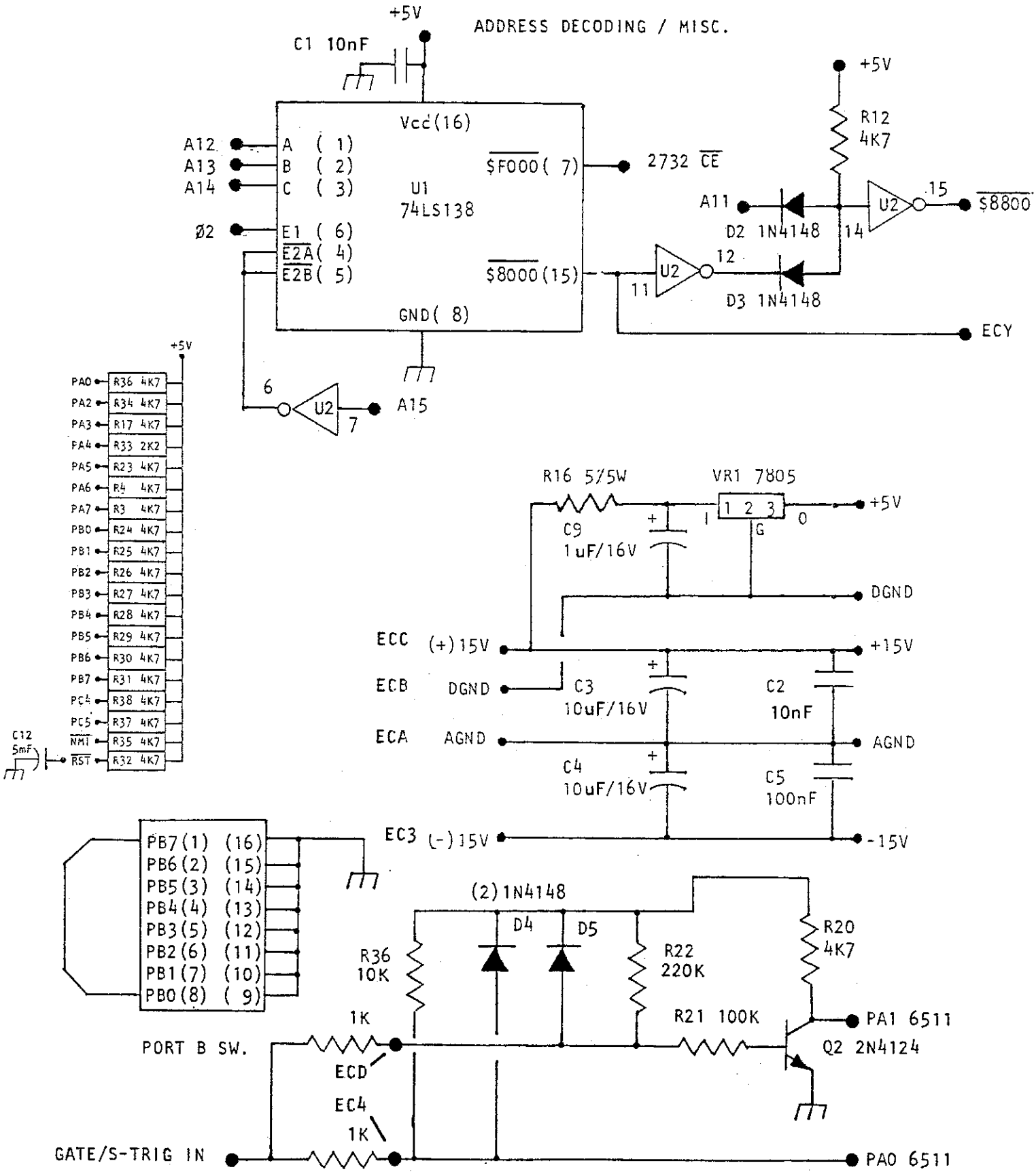
PARTS PLACEMENT DIAGRAM



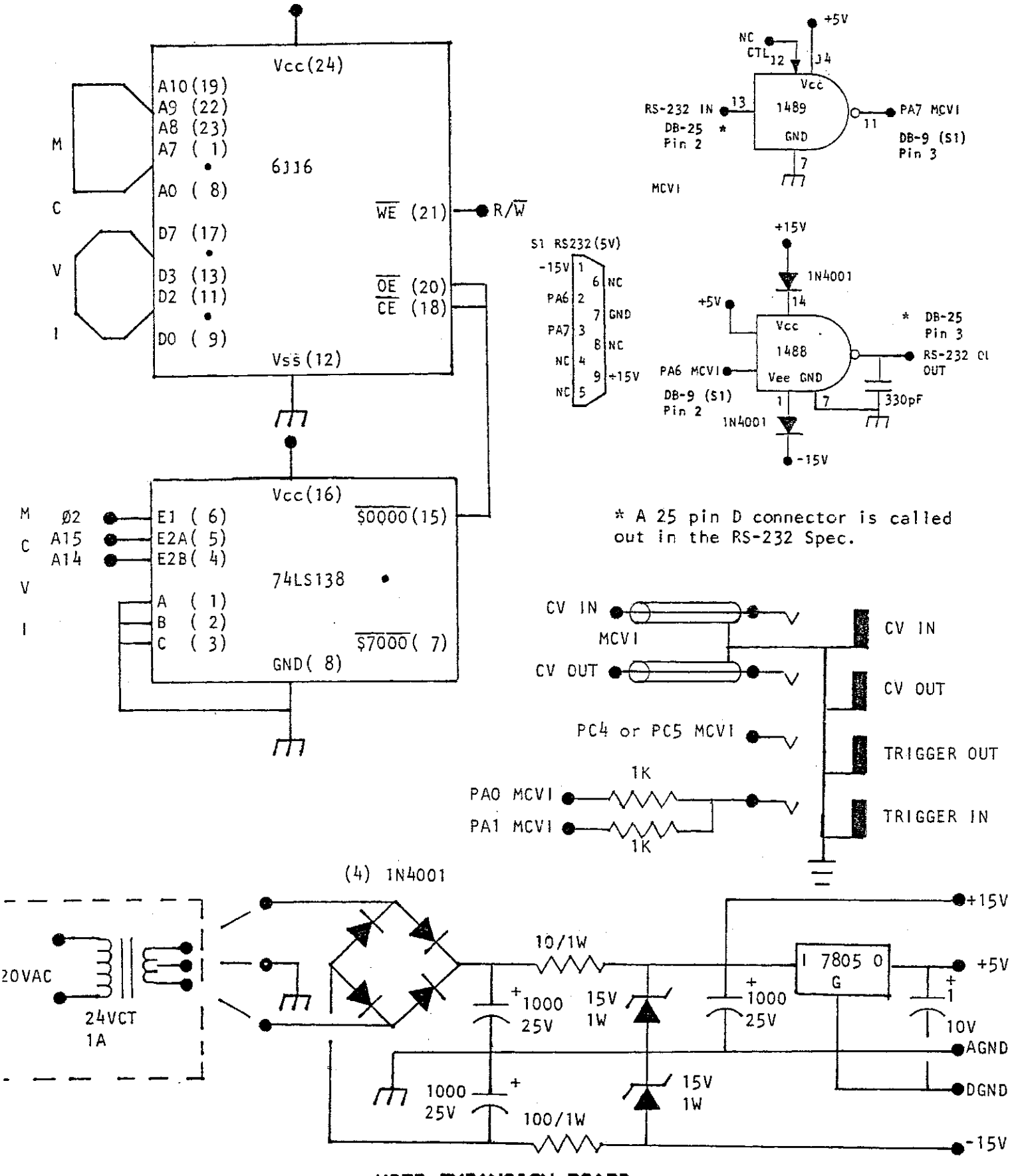
MCVI/CPU SCHEMATIC



MCVI/CPU SCHEMATIC (cont'd.)



MCVI/CPU SCHEMATIC (cont'd.)



USER EXPANSION BOARD

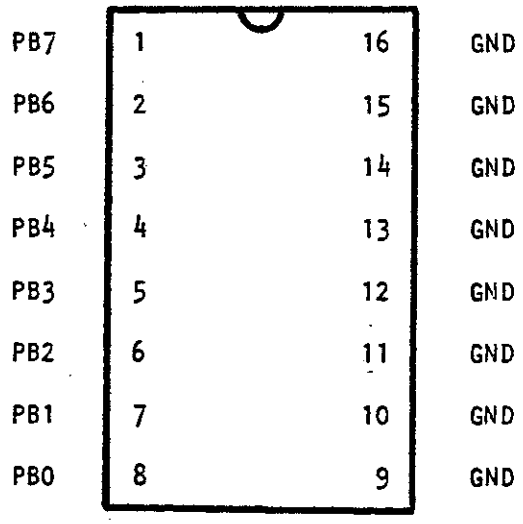
TRANSMIT CHANNEL

	PB0	PB1	PB2	PB3
1	H	H	H	H
2	L	H	H	H
3	H	L	H	H
4	L	L	H	H
5	H	H	L	H
6	L	H	L	H
7	H	L	L	H
8	L	L	L	H
9	H	H	H	L
10	L	H	H	L
11	H	L	H	L
12	L	L	H	L
13	H	H	L	L
14	L	H	L	L
15	H	L	L	L
16	L	L	L	L

RECEIVE CHANNEL

	PB4	PB5	PB6	PB7
1	H	H	H	H
2	L	H	H	H
3	H	L	H	H
4	L	L	H	H
5	H	H	L	H
6	L	H	L	H
7	H	L	L	H
8	L	L	L	H
9	H	H	H	L
10	L	H	H	L
11	H	L	H	L
12	L	L	H	L
13	H	H	L	L
14	L	H	L	L
15	H	L	L	L
16	L	L	L	L

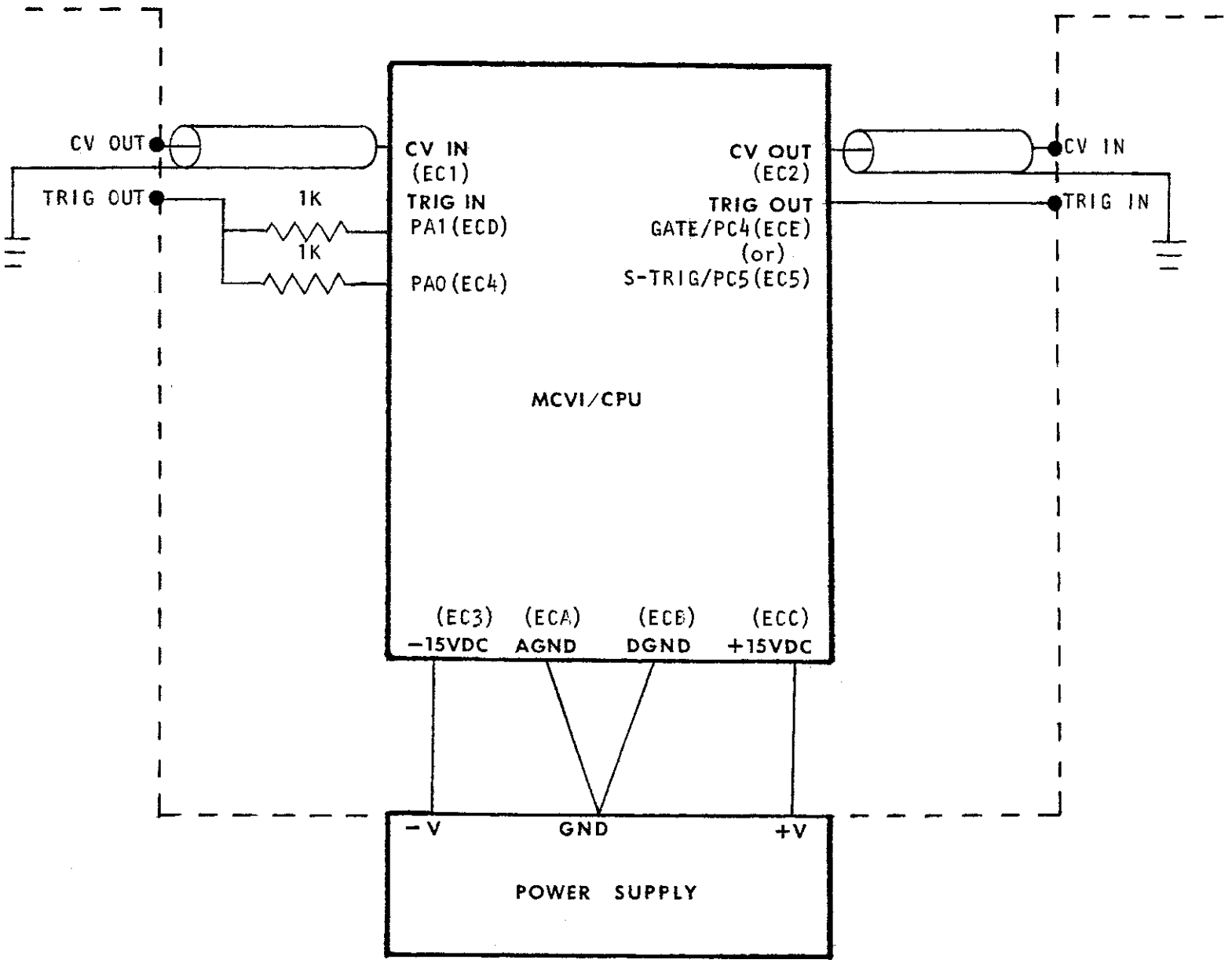
PORT B SOCKET



H = NO connection between PB line and GND.

L = Connection between PB line and GND.

Transmit Channel Port B socket Receive channel



S Y N T H E S I Z E R

SYNTHESIZER - MCVI/CPU WIRING

USING THE SMPLMON (tm) MONITOR

SMPLMON is a simple monitor program, but despite its simplicity it provides the user with all.of the functions necessary to turn the MCVI into a completely adequate developement platform for not only MIDI but numerous other musical electronic and control functions as well.

CONNECTING A TERMINAL

Smplmon communicates with the user through the MCVI's 4800 baud RS232-like interface. Unlike normal RS-232, the MCVI's comm channel is not bi-polar, but has a 0v. mark and 5v. space. This is not an uncommon compromise in contemporary equipment and the protocol will perfectly well with much equipment available today.

On the other hand, equipment which recognizes only "real" RS-232 will require external voltage translators to convert the MCVI's 0-5 to RS-232's +12 to -12 convention. National Semiconductors 1488 and 1489 IC's are quick and convenient means of providing this interface and are readily available from Radio Shack. Wiring of these parts is shown in Fig. 1.

ACCESSING SMPLMON

SMPLMON is accessed by sending an ASCII Carriage Return ($8D or $0D) into the MCVI boards RS-232 input. These characters will cause the application program running at the time to be abandoned and SMPLMON to be called.

SMPLMON responds the the CR by announcing:

SMPLMON (tm) (c) 1987 JSS
(I will try to remember to use boldface for computer stuff)

and then waits for input from the terminal.

SMPLMON COMMANDS

SMPLMON recognizes 4 command character strings

M - for Memory, as in examine and alter memory.

This command will normally followed by a 4 character hexadecimal address which will be the point from which SMPLMON will list the contents of the next 8 memory locations. As, for example:

MFFF8 [CR] (don't forget the Carriage Return)

will be responded to with:

XX XX XX XX XX XX XX XX

which is the data in memory locations $FFF8 - $FFFF

The specific memory locations mentioned above are in ROM and so cannot be changed, but to alter the data in locations which can changed, simply enter the new data on the line below the one printed by SMPLMON. Like this:

```
M0040 [CR]                        (you ask, quizicly)
xx xx xx xx xx xx xx xx           (this may be anything)
01 02 03 04 05 06 07 08 [CR]      (you reply)
M [CR]                            (you again, 0040 not needed now)
01 02 03 04 05 06 07 08           (new memory contents)
[CR]                              (smplmon quirk, must have)
M0048 [CR]                        (let's see next 8 bytes)
xx xx xx xx xx xx xx xx           (and so on .... )
```

Sorry about the CR being needed after a line of memory is printed out, it simply has not been bother enough to me to warrant fixing, and I use this thing every day.

T - Transfer (download) host memory to MCVI

This command will always be followed by 2 addresses, the starting address and ending address. The starting address is the first address at which data from a host will be put and the ending address is the last address to be transfered.

```
T0100 0200 [CR]                   (you assert commandingly)
```

will cause SMPLMON to listen for a continuous stream of data on the RS-232 which it will respond to by placing sequential input bytes in sequenctial address starting at location $0100. When enough bytes have been received to fill memory to the ending address ($0200 in this case), SMPLMON will send a carriage return ($8D) to the host system.

This command allows programs being written on a developemt system to be transfered to the target MCVI for test, evaluation and (most likely) debug. There is no provision for error detecting on this transfer other than simply that the correct number of bytes have been transfered as indicated by the CR coming back from SMPLMON when the host expects it to. I have used this system for several years, and in that time have had a single bad transfer. In this case, lack of error detection would not seem to be too critical.

In my case, the developement system is an APPLE II running EDASM. The terminal program which I use as the APPLE II end of talking to the APPLE is listed in appendix B.

R - Restart system

this command causes SMPLMON to be restarted and procuces

SMPLMON (tm) (c)1987 JSS

on the terminal. You can use this command to refresh your memory on the exact copyright date of SMPLMON and sometimes its handy to test to see if it's really SMPLMON that you're talking to.

G - Go (forth and run this program)

This command will also often be followed by 4 bytes worth of address in hexadecimal and the address is the entry address of the program to be run.

G0100 [CR] (you command assertively)

will of course produce no response from SMPLMON because you are no longer "in" (as we like to say a lot) SMPLMON. You are now "in" (see) your application and unless it addresses some output to the terminal all will be quiet.

Sometimes too quiet. And it is for this reason that it is handy to be able to take look at where your program is (and where it isn't). The 65xx provision for this is the BRK instruction ($00). There is no need to go into the 6511's BRK since it is covered so well in the material in the bibliography. But here is a good place to mention that BRK are always responded to by SMPLMON which upon encounter one will announce:

Bxxxx aa xx yy ss

where: xxxx = the address where the BRK was encountered
aa = the contents of the accumulator at the BRK
xx = the contents of the X index
yy = the contents of the Y index
ss = the contents of the processor status register

SMPLMON then waits for commands.

SMPLTERM - PROGRAM LISTING

```
1  DIM H$(15)
2 H$(5) = "5":H$(6) = "6":H$(7) = "7":H$(8) = "8":H$(9) = "9"
3 H$(10) = "A":H$(11) = "B":H$(12) = "C":H$(13) = "D":H$(14) = "E":H$(15) =
     "F"
4 H$(0) = "0":H$(1) = "1":H$(2) = "2":H$(3) = "3":H$(4) = "4"
5 D$ = ""
20  PRINT D$;"IN#2": PRINT D$;"PR#2"
32  INPUT B$
35  IF  LEFT$ (B$,1) = "T" THEN 60
37  POKE 2042, PEEK (2042) + 128
40  PRINT B$
45  POKE 2042, PEEK (2042) - 128
50  GOTO 32
60  IF  LEFT$ (B$,2) = "T#" THEN 360
65 C$ =  RIGHT$ (B$, LEN (B$) - 1)
70  PRINT D$;"BLOAD";C$
80 HB =  INT (( PEEK (43635)) / 16)
90 LB =  PEEK (43635) - HB * 16
100  PRINT "T";: FOR N = 1 TO 500: NEXT N
105  PRINT H$(HB);H$(LB);
110 HB =  INT (( PEEK (43634)) / 16)
120 LB =  PEEK (43634) - HB * 16
130  PRINT H$(HB);H$(LB)
140 HB =  INT (( PEEK (43617)) / 16)
150 LB =  PEEK (43617) - HB * 16
160  PRINT H$(HB);H$(LB);
170 HB =  INT (( PEEK (43616)) / 16)
180 LB =  PEEK (43616) - HB * 16
190  PRINT H$(HB);H$(LB);
200 BL =  PEEK (43634):BH =  PEEK (43635)
210 EL = BL +  PEEK (43616):CRY =  INT ((EL) / 256):EH = BH + CRY +  PEEK
     (43617)
212  REM PRINT BL;"  ";BH;"  ";EL;"  ";EH
215  PRINT " "
220  POKE 60,BL: POKE 61,BH: POKE 62,EL: POKE 63,EH
230  CALL  - 13503
240  GOTO 32
360 C$ =  RIGHT$ (B$, LEN (B$) - 2)
370  PRINT D$;"BLOAD";C$;",A$5000"
380  PRINT "T";: FOR N = 1 TO 500: NEXT N: PRINT "0100"
390  GOTO 140
```

BIBLIOGRAPHY

Zaks, Rodney. (1979. 6502 Applications. Berkley: Sybex.

Lancaster, Don. (1983). Micro Cookbook Vol. II; Machine Language Programming. Indianapolis: Howard W. Sams & Co., Inc.

Chamberlin, Hal. (1980). Musical Applications of Microprocessors. Rochelle Park, NJ: Hayden Book Company, Inc.

R6500 Microcomputer System Programming Manual (1979). Anaheim: Rockwell International Corp.

Leventhal, Lance A. & Saville, Winthrope. (1982) 6502 Assembly Language Subroutines. Berkley: Osborne/McGraw-Hill.

6511AQ DOCUMENTATION REPRINTS AVAILABLE

User's who desire more detailed programming information for the 6511AQ microprocessor can acquire a reprint of the complete 40 page text of Rockwell International Corporation's documentation on this chip. It contains detailed specifications and programming data for this one-chip microprocessor.

To order 6511AQ Documentation, include $10.00 check or money order and your name, address and zip code.

MAIL YOUR ORDER TO:

PAIA ELECTRONICS, INC.
3200 TEAKWOOD LN.
EDMOND, OK 73013

VISA & MASTERCARD ACCEPTED FOR PHONE ORDERS:

(405) 348-6300 (9 a.m. to 5 p.m. CST)