

# PROMENADE C1 PROFESSIONAL EPROM PROGRAMMER

For the COMMODORE 64 and 128

With PROMOS 2.0 Operating System Software



This manual and the PROMOS 2.0 software program are COPYRIGHT (C) 1988 by J. W. Ranheim. All rights are reserved. Neither this manual nor the PROMOS 2.0 program may be copied, reproduced, transferred, transmitted, disclosed or reduced to machine-readable form including magnetic or other storage media, without the prior written consent of the copyright owner, except that a reasonable number of archival copies of the PROMOS 2.0 software program may be made by the purchaser for his own exclusive personal use. PROMENADE, PROMENADE C1, PROMOS, and PROMOS 2.0 are trademarks of Jason-Ranheim Company.

### **LIMITED WARRANTY**

JASON-RANHEIM COMPANY warrants to the ORIGINAL PURCHASER of these products that they are free from defects in materials and workmanship for a period of ninety (90) days from the date of purchase. These Jason-Ranheim Company products are sold 'AS IS' without express or implied warranty of any kind, and Jason-Ranheim Company is not liable for any losses or damages of any kind resulting from the use of these products. Jason- Ranheim Company agrees for a period of ninety (90) - days to repair or replace, at its option, free of charge, any Jason-Ranheim Company product, postage paid, with proof of the date of purchase at its Factory Service Center. This warranty is not applicable to normal wear and tear. This warranty shall not be applicable and shall be void if the defect in the Jason-Ranheim Company product has arisen through abuse, unreasonable use, mistreatment or neglect. This warranty is in lieu of all other express warranties and no other representation or claims of any nature shall be binding on or obligate Jason-Ranheim Company. Any implied warranties applicable to these products including warranties of merchantability and fitness for a particular purpose are limited to the ninety (90) day period described above. In no event will Jason-Ranheim Company, its officers, employees, agents, successors or assigns be liable for any special, incidental, or consequential damage resulting from the possession, use or malfunction of these Jason-Ranheim Company products.

JASON-RANHEIM COMPANY  
3105 Gayle Lane  
Auburn, Calif. 95603  
(916) 878-0785

## TABLE OF CONTENTS

INTRODUCTION .....	1
GETTING STARTED .....	2
HANDLING EPROM's .....	2
HEXADECIMAL NOTATION .....	2
EPROM's .....	4
EPROM PINOUT, FIGURE 1 .....	5
ENTERING PROMOS 2.0 COMMANDS .....	6
MULTIPLE COMMANDS .....	6
MIXING BASIC AND PROMOS COMMANDS .....	6
ERRORS .....	7
PROMOS 2.0 COMMANDS .....	7
G COMMAND .....	7
R OR £ COMMAND .....	7
CHECKSUMS AND HASHTOTALS .....	8
WHERE TO PUT THE DATA .....	8
B COMMAND .....	8
HOW BIG A RANGE? .....	9
M COMMAND .....	9
CHANGING RAM DATA .....	10
I COMMAND .....	10
F COMMAND .....	10
T COMMAND .....	10
C COMMAND .....	11
H COMMAND .....	11
D COMMAND .....	12
A COMMAND .....	12
E COMMAND .....	12
P OR TT COMMAND .....	12
PROGRAMMING ERRORS .....	14
V COMMAND .....	14
COPYING EPROM's .....	14
S COMMAND .....	15
L COMMAND .....	15
Q COMMAND .....	16
Z COMMAND .....	16
SPECIAL EPROM's .....	17

USING PROMOS COMMANDS IN A BASIC PROGRAM -----	18
SIMPLE SAMPLE PROGRAM -----	19
USING BASIC VARIABLES -----	19
SETTING UP A RAM BUFFER -----	20
ONE FURTHER POINT -----	20
WHERE'S PROMOS? -----	21
TROUBLESHOOTING -----	21
APPENDIX A- THE CONTROL WORD -----	22
APPENDIX B- THE PROGRAM METHOD WORD -----	23
APPENDIX C- TABLE of CONTROL and PROGRAM METHOD WORDS	
APPENDIX D- THE G COMMAND SCREEN DISPLAY	
APPENDIX E- THE PARTS LIST	
APPENDIX F- THE BLOCK DIAGRAM	
APPENDIX G- THE PCB LAYOUT AND SCHEMATIC DIAGRAMS	

## INTRODUCTION

The acronym EPROM stands for Erasable Programmable Read Only Memory. It refers to another of those electronic marvels that came out of INTEL Corp. in the mid Seventies. A memory chip that the user can program to contain any sort of digital information whatsoever. And even better, a chip that can be erased by simple exposure to ultra-violet light and reused many times. With these abilities, it is no wonder that EPROM's are produced in such tremendous quantities. The dollar volume is exceeded only by that of dynamic RAM'S.

With the tremendous proliferation of types and sizes, programming an EPROM became a very complicated undertaking. Many EPROM programming devices were designed to handle just a single EPROM type; and those that could do more required setting a multitude of switches or plugging in a 'personality module'. Some even required a voltmeter to set up. Programming an EPROM could easily take an entire afternoon. That unhappy situation suddenly changed with the introduction of the PROMENADE C1.

By combining some sophisticated hardware with the power of an inexpensive personal computer, the PROMENADE makes the job almost trivial. And with new programming techniques, the PROMENADE cuts the time required to duplicate an EPROM from many minutes to just seconds. With all of its features and capabilities, the PROMENADE has set the standard for EPROM programmers all around the world. It has been widely imitated, even copied exactly; but its most important attribute has never been duplicated- VALUE.

But even the best products can be improved, and with the introduction of PROMOS 2.0 the PROMENADE is made even better. Simple, yet powerful commands for the manipulation of data in the computer and for moving that data to and from EPROM'S, disk drives and the like give performance and ease of use that have never before been achieved. Even better, these commands can be incorporated into BASIC programs that let the user customize his system to any EPROM programming application.

The user is strongly urged to read this manual carefully and to thoroughly get familiar with the material. An hour or two spent this way will be repaid many times over.

JASON-RANHEIM COMPANY

Auburn, California

November, 1988

## GETTING STARTED

With the power OFF, plug your PROMENADE into the user Port at the left rear of your COMMODORE 64 or 128. Install the PROMOS 2.0 diskette in your 1541, 1571 or 1581 disk drive. Turn ON your computer. If you are in 128 mode, the software will load automatically. If in 64 mode, type:

```
LOAD"PB",8,1 <CR>
```

The <CR> above means 'type a carriage return'.

You will shortly see a brief sign-on message and copyright notice. At this point, your system is ready.

## HANDLING EPROMS

EPROM's need to be handled with reasonable care if one is to avoid irretrievable damage to them. They are very susceptible to static electricity. This can be a problem in a cold, dry climate in the wintertime. Always discharge yourself by touching a grounded object before handling them. The aluminum case of the PROMENADE itself is convenient for this purpose. To install an EPROM in the PROMENADE socket, RAISE the operating lever, drop the EPROM into place and then close the socket by moving the lever down to the case as far as it will go. NOTE- EPROM's having 24 pins (e.g. 2716, 2732) use the RIGHTMOST 24 socket positions. Twenty eight pin EPROM's occupy the entire socket. MAKE SURE the orientation notch points to the LEFT as the figure on the top of the cover indicates.

NOW TO INSURE PROPER ELECTRICAL CONNECTION to all pins of the EPROM, move the EPROM from side to side slightly by forcing it with the thumbs. This may take quite a bit of force. Doing this becomes habitual after a while. The objective here is to have the contacts in the socket cut through any tarnish films which might otherwise prevent proper operation.

## HEXADECIMAL NOTATION

If you are already familiar with HEX notation, skip this section. If not, you will need to spend a little effort at this point to acquaint yourself with a very useful idea.

In the world of EPROM's and computers, information is conveyed in little units known as 'BITS'. A bit can have one of two values. The meaning we associate with these values can be anything capable of expression as a 'binary' or two-valued entity. For example, a statement can be either TRUE or FALSE.

A light is either ON or OFF. In your computer, a bit is represented by the voltage on a wire: it is either HIGH (near 5 volts) or LOW (near 0 volts). The high value we associate with the symbol '1'. The low value we associate with the symbol '0'. In order to represent members of larger collections of objects, letters of the alphabet, for example, or large numbers, binary digits must be grouped together.

Four binary digits taken together can represent 16 different objects, two to the 4<sup>th</sup> power. Eight binary digits taken together form a unit called a 'BYTE' in computers. A byte can stand for any one of two to the 8<sup>th</sup> power (or 256) different objects. Your computer deals in 'BYTE' sized information units. Data flows from one part of your machine to another along eight wires called the **data bus** in bytes.

In order to specify exactly what information is being accessed at a given instant, your computer sends out an 'ADDRESS'. This address is a 16 bit binary quantity that flows along 16 wires called the **address bus**. Since there are 16 address lines, two to the 16<sup>th</sup> power, or 65536 unique addresses can be specified by your machine.

Now, instead of using binary numbers directly like your computer does, it is customary to consider binary digits in groups of four. Each possible value of this group is denoted by a symbol: the numbers from 0 to 9 plus the letters A through F. These are called 'HEXADECIMAL' numbers. Accordingly, we have the following:

<u>BINARY</u>	<u>HEXADECIMAL</u>	<u>DECIMAL</u>
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15
10000	10	16

And we can represent numbers of any size:

11110000	F0	240
11111111	FF	255
100000000	100	256
111111111111	1FFF	8191
11101000101011010	1D15A	119130

And so on. It is common to see a '\$' sign affixed to a hexadecimal number so that the reader will not take it to be an ordinary decimal quantity. Or sometimes a following 'h' does the same thing. So \$1234 or 2345h are to be considered hexadecimal numbers. In this manual, all numbers are in 'HEX' unless specifically stated otherwise.

### EPROM's

Now, consider for a moment the diagram of Figure 1. It shows the 'pin out' of several commonly used EPROM'S. In use, the pins marked Q0 through Q7 are connected to the data bus. The EPROM, when enabled to do so, outputs data on these pins. Since there are eight output pins, the EPROM can output an entire byte at a time. An EPROM is a 'byte-wide' device.

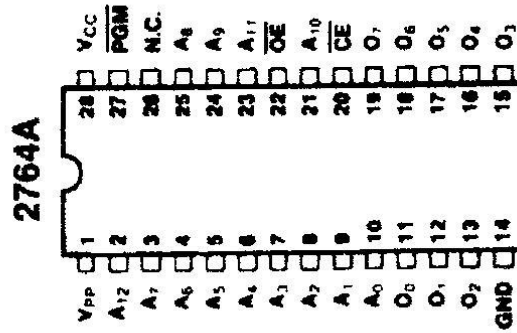
The pins marked A0 through A15 are address pins. The computer imposes a binary address on these pins to specify which-location in the EPROM the data is to come from. The lowest address is, of course, \$0000, all address lines low. The highest EPROM address depends on how many address pins there are. The capacity of an EPROM is usually specified as a certain number of 'KILOBYTES'. A kilobyte is two to the 10<sup>th</sup> power or \$400 bytes, (1024 decimal). A '1k' EPROM like the 2758 must therefore have 10 address pins. The following table gives the maximum EPROM address for several common types.

TYPE	CAPACITY	HIGHEST ADDRESS
2758	1k	3FF
2716	2k	7FF
2732	4k	FFF
2764	8k	1FFF
27128	16k	3FFF
27256	32k	7FFF
27512	64k	FFFF

When the PROMENADE programs an EPROM, the desired EPROM address is applied to the address pins, the data to be programmed into the EPROM at that address is applied to the data pins, and various signals and voltages are applied to other pins as required.



2716	2732A	27128A 27C128	27256 27C256	27512 27C512
V <sub>CC</sub> A <sub>8</sub> A <sub>9</sub> V <sub>PP</sub> OE A <sub>10</sub> CE O <sub>7</sub> O <sub>6</sub> O <sub>5</sub> O <sub>4</sub> O <sub>3</sub>	V <sub>CC</sub> A <sub>8</sub> A <sub>9</sub> A <sub>11</sub> OE/V <sub>PP</sub> A <sub>10</sub> CE O <sub>7</sub> O <sub>6</sub> O <sub>5</sub> O <sub>4</sub> O <sub>3</sub>	V <sub>CC</sub> PGM A <sub>13</sub> A <sub>9</sub> A <sub>9</sub> A <sub>11</sub> OE A <sub>10</sub> CE O <sub>7</sub> O <sub>6</sub> O <sub>5</sub> O <sub>4</sub> O <sub>3</sub>	V <sub>CC</sub> A <sub>14</sub> A <sub>13</sub> A <sub>9</sub> A <sub>9</sub> A <sub>11</sub> OE A <sub>10</sub> CE O <sub>7</sub> O <sub>6</sub> O <sub>5</sub> O <sub>4</sub> O <sub>3</sub>	V <sub>CC</sub> A <sub>14</sub> A <sub>13</sub> A <sub>9</sub> A <sub>9</sub> A <sub>11</sub> OE/V <sub>PP</sub> A <sub>10</sub> CE O <sub>7</sub> O <sub>6</sub> O <sub>5</sub> O <sub>4</sub> O <sub>3</sub>



27512 27C512	27256 27C256	27128A 27C128	2732A	2716
A <sub>15</sub> A <sub>12</sub> A <sub>7</sub> A <sub>6</sub> A <sub>5</sub> A <sub>4</sub> A <sub>3</sub> A <sub>2</sub> A <sub>1</sub> A <sub>0</sub> O <sub>0</sub> O <sub>1</sub> O <sub>2</sub> GND	V <sub>PP</sub> A <sub>12</sub> A <sub>7</sub> A <sub>6</sub> A <sub>5</sub> A <sub>4</sub> A <sub>3</sub> A <sub>2</sub> A <sub>1</sub> A <sub>0</sub> O <sub>0</sub> O <sub>1</sub> O <sub>2</sub> GND	V <sub>PP</sub> A <sub>12</sub> A <sub>7</sub> A <sub>6</sub> A <sub>5</sub> A <sub>4</sub> A <sub>3</sub> A <sub>2</sub> A <sub>1</sub> A <sub>0</sub> O <sub>0</sub> O <sub>1</sub> O <sub>2</sub> GND	A <sub>7</sub> A <sub>6</sub> A <sub>5</sub> A <sub>4</sub> A <sub>3</sub> A <sub>2</sub> A <sub>1</sub> A <sub>0</sub> O <sub>0</sub> O <sub>1</sub> O <sub>2</sub> GND	A <sub>7</sub> A <sub>6</sub> A <sub>5</sub> A <sub>4</sub> A <sub>3</sub> A <sub>2</sub> A <sub>1</sub> A <sub>0</sub> O <sub>0</sub> O <sub>1</sub> O <sub>2</sub> GND

The user has to be concerned with only one of these other voltages. This is a relatively high value called the PROGRAMMING VOLTAGE. There are three standard programming voltages in general use: 12.5v, 21v, and 25 volts. As we shall see a little later, the user must specify this voltage for the particular EPROM he is dealing with.

## **ENTERING PROMOS 2.0 COMMANDS**

All of the various tasks you will want to accomplish using the PROMENADE will be done by means of one or more PROMOS 2.0 commands. These commands consist of a key letter followed by up to nine hexadecimal numbers called parameters. To execute a PROMOS 2.0 command, simply type it on the screen. Edit it as required and then type a carriage return.

Multiple parameters in your command may be separated by one or more spaces or by (at most) one comma. You do not need to supply a '\$' sign nor are you required to supply leading zeros. The following examples of the PROMOS 'M' command will serve to illustrate. This command is used to display the contents of computer memory on the screen, and it will be described in detail later.

```
M0000 03FF
M0000,03FF
M0 3FF
M0,3FF
M 0      3FF
M 00 , 03FF
```

All of these forms are equivalent. PROMOS does not perversely require that you type in an exact form as long as your meaning is unambiguous.

## **MULTIPLE COMMANDS**

You may enter two or more PROMOS commands separated by colons on the same command line. The commands will be executed sequentially in the order you typed them. For example:

```
M2000,2040:12000,2040
```

It results in a hex display of computer RAM data from 2000 through 2040, followed by an ASCII display of the same data.

## **MIXING BASIC AND PROMOS COMMANDS**

You may mix PROMOS and BASIC commands together on the same command line. They will execute in the order in which you typed them. For example:

Print "ZERO PAGE RAM":M0,FF:Print"END OF ZERO PAGE"

REMEMBER: ALL PROMOS 2.0 COMMAND PARAMETERS ARE ENTERED AS HEXADECIMAL NUMBERS.

## **ERRORS**

If you make a mistake typing in a PROMOS command, chances are you'll be advised by the SYNTAX error message. If you fail to supply all the parameters required by a command, an UNDEFINED FUNCTION error will result. If you input incorrect hexadecimal numbers, an ILLEGAL QUANTITY error will be reported. Some PROMOS commands report other errors as may be necessary.

## **PROMOS 2.0 COMMANDS**

### **THE G COMMAND- Print command Glossary.**

This isn't really a command at all. Type a G and a carriage return and a list of the PROMOS commands and an example of each is printed on the screen. If you commit only one command to memory, make it this one.

THE R (or £) COMMAND- Read EPROM data into computer memory.

By reading an EPROM, we mean transferring some or all of the information it contains into the computer's memory. The key letter can either be R or £, the English monetary symbol. Example:

R2000, 3FFF, 0, 6

The meaning of this command is: fill computer RAM from 2000 to 3FFF with data from this EPROM, beginning at EPROM address 0000. The '6' is a parameter called the CONTROL WORD. It is a number that tells the PROMENADE everything it needs to know about the particular EPROM you are dealing with. In the example, 6 is appropriate for the 2764A. A list of control words appears inside the back cover of this manual. A detailed discussion of control words and what they mean is presented in APPENDIX A. Another example:

£ 5000 BFFF 1000 E6

Here, we have used the £ symbol rather than R. The command does exactly the same thing either way.

In this example, we are telling the PROMENADE to read a 27256 into computer RAM from 5000 through BFFF, beginning at EPROM address 1000. E6 is the control word for a 27256 programming at 12.5 volts.

## CHECKSUMS and HASHTOTALS

When you execute the READ command as in the examples above, you will notice that a 'CHECKSUM' and a 'HASHTOTAL' is printed on the screen as soon as the operation is completed. The CHECKSUM is the low-order byte of the arithmetic sum of all the data bytes read from the EPROM. The HASHTOTAL is the result of exclusive-or of all bytes read with each other. These numbers can serve as a quick means of identification of an EPROM, or as a means of checking the validity of the data itself.

## WHERE TO PUT THE DATA

Notice that the READ command gives you complete freedom: you can read as many bytes from the EPROM as you want, starting anywhere on the chip, and in principle you can put them anywhere in computer memory. As a practical matter, certain areas of computer RAM are off-limits because your computer must use them for its own purposes. If your read command overwrites these areas, the computer will crash.

Consider first the COMMODORE 64 or the 128 in 64 MODE: if you are not running a BASIC program, then you are free to use computer RAM from 0800 through DFFF. A BASIC program ordinarily resides in RAM starting at 0801. In a later section we discuss the matter of using PROMOS commands in a BASIC application program.

In the case of the COMMODORE 128 in 128 MODE, we need to consider a special PROMOS command which pertains only to the 128, the bank select command.

### The B COMMAND- Select PROMOS RAM bank, (128 mode command).

This command selects the computer RAM bank, **0** or **1**, which will be used for subsequent PROMOS operations. Example:

**B1**

Execution of this command causes PROMOS to use bank **1** RAM, PROVIDED the RAM involved lies in the range 2000 through DFFF. Outside this range, bank **0** RAM will be used. Using this command does not in any way affect the BASIC 7.0 'BANK' command. PROMOS bank **1** will remain in use until you switch back to PROMOS bank **0**. To do this, execute either **B0** or just **B**. Example:

B1:R2000,5FFF,0,6:B0

Here, we combine the bank select commands and the read command. First we switch to bank 1 RAM; then we read a 27128A into the range 2000 through 5FFF in bank 1; then we switch back to bank 0 RAM.

In 128 mode, BASIC starts at 1C01 in bank 0. If no BASIC program is being used, useable RAM in bank 0 starts at 1300.

### **HOW BIG A RANGE?**

The question arises: How large must the memory range in the read command be to insure that each and every Byte on the EPROM is included? Earlier, we presented a table showing the largest EPROM address for EPROMs of various capacities. If we start at address 0 on the EPROM and read each address to and including the highest address, then we know we have read them all. A little reflection will show that all one has to do is add this highest EPROM address to the chosen computer RAM start address to obtain the necessary computer RAM end address. Some examples:

We want to read all of a 2732 (4k) EPROM into RAM starting at C000. Since the EPROM address runs from 0 to FFF, the required read command is:

RC000,CFFF,0,E0

A 27256 (32k) is to be read into RAM starting at 4000:

R4000,BFFF,0,E6 does the job.  $4000 + 7FFF = BFFF$ .

A 27128 (16k) is to be read into RAM starting at 3840:

R3840,783F,0,5 is what is needed.

### **THE M COMMAND- Display computer memory.**

This is the command to use to view and alter computer RAM data. To illustrate its use we execute the following:

MA000

And there appears:

>A000 3E 41 00 A9 20 FE 84 D0

Execution of this 'M' command causes the computer to display the specified address followed by eight hexadecimal byte values, the data from locations A000 through A007. To display a memory range, from 2000 through DFFF, say, type:

M2000,DFFF

The screen fills and then scrolls rapidly as the data is printed. To stop the display at any point, press the RUN/STOP key. To begin the display again at the point where you stopped, type M alone and a carriage return.

Notice the 'M' command displays RAM data, not ROM or IO devices which may overlay a particular address.

### **CHANGING RAM DATA**

Suppose you want to change the data in memory somewhere. Just use the M command to display that area, move the cursor into the display to the desired location, type in the new hex value(s) and press RETURN on each altered line. RAM data is changed accordingly. You can also change the address field. In this way you can quickly copy data from one area of RAM to another.

### **THE I COMMAND- Display ASCII data.**

This command works the same way as the M command except that the display shows printable ASCII characters rather than hex data. Thirty two characters appear on each line. For example:

I6000,7FFF

Displays ASCII characters in the range 6000 through 7FFF. You cannot alter an ASCII string by over typing and pressing RETURN.

### **THE F COMMAND- Fill computer RAM.**

This command fills a specified range of memory with a specified data byte. For example:

F2000,DFFF,FF

This command fills memory from 2000 through DFFF with FF characters.

### **THE T COMMAND- Transfer RAM data.**

This command copies data in a specified range from one area of computer RAM to another. By way of example:

T2000,2FFF,C000

This command transfers data from RAM in the range 2000 through 2FFF to C000 through CFFF. Another example:

T4000,40FF,4001

This command moves data from 4000 through 40FF up one address location.

### **THE C COMMAND- Compare computer RAM data.**

This command causes the computer to compare the data in one range of memory locations with that of another. The address of all locations where the data differs is printed out. For example:

C100,1FF,4000

This command compares RAM data in the range 0100 through 01FF to data in the range 4000 through 40FF. The addresses of all discrepant locations appear on the screen.

### **THE H COMMAND- Hunt for byte data.**

This command causes the computer to hunt for occurrences of a group of specified data bytes. The address of each occurrence is printed on the screen. The sought for group may be up to seven bytes long. For example:

H2000,BFFF,20,D2,FF

This command examines RAM from 2000 through BFFF for the byte sequence 20,D2,FF: 6502 code for JSR FFD2. Another example:

H1000,7FFF,52,41,4E,47,45 49,4D

This command hunts for a group of seven ASCII characters.

### **THE D COMMAND- Scan EPROM data.**

The D command reads data from an EPROM and prints it on the screen as hex data just as the M command does. Computer RAM is not affected. Example:

D0,1FFF,5

This command produces a screen display of all the data on a 2764. The first two parameters are the starting and ending EPROM ADDRESSES of the range to be displayed. The third parameter is the CONTROL WORD for the particular EPROM. Another example:

D6000,7FFF,E6

This command scans the top 8k of a 27256.

#### **THE A COMMAND- ASCII scan of EPROM data.**

The A command works just like the D command except that the display is of printable ASCII characters. For example:

A0,7FF,28

In this example, we do an ASCII scan on all bytes of a 2716.

#### **THE E COMMAND- Check EPROM for erased condition.**

The data at all erased locations on an EPROM is FF; that is, all data bits are ones. To check for this condition, use the E command. An example:

E0,3FFF,6

This command checks for FF at all locations on a 27128A. The EPROM addresses of all un-erased locations are printed on the screen. Their total number is also printed. The first and last addresses on the 27128A are 0 and 3FFF. The control word is 6. Another example:

E800,FFF,E0

This command checks for the erased condition of the top half of a 2732.

#### **THE P (or TT symbol) COMMAND- Program EPROM from computer memory.**

The key letter for this command can be either **P** or (**TT** symbol). The command works the-same either way.

There are five parameters needed for this command: the STARTING and ENDING computer MEMORY ADDRESSES defining where the information is to come from, the STARTING ADDRESS on the EPROM specifying where we want the data to go, the CONTROL WORD; and the last parameter is the PROGRAM METHOD WORD (PMW for short).



The PMW is discussed at length in Appendix B. It tells PROMOS how you want the programming to be done. Some examples:

P2000,3FFF,0,6,7

This command fills a 2764A completely with data from computer RAM in the range 2000 through 3FFF. The EPROM is programmed from beginning to end. The CONTROL WORD is 6; and we have used intelligent programming method #1 as defined by the PMW of 7.

Notice that the P command gives you complete freedom. You can take information from anywhere in computer RAM and put it anywhere on the EPROM. Suppose you wanted to program just one byte on an EPROM:

P213A,213A,3456,5,A

In this case, the starting and ending RAM addresses are the same, 213A. The EPROM address is 3456. The control word, 5 is for the 27128 and we have chosen A for the PMW. This command takes a single byte of data at 213A and programs it into EPROM location 3456 in the 27128.

The considerations of RANGE size are exactly the same as with the R command. To be sure you are programming ALL of an EPROM; add the maximum EPROM address to the computer RAM starting address to obtain the computer RAM ending address. Some more examples:

Suppose you wanted to take the data from two 2764's and put it on a single 27128A. Here's one way to do it. First let's choose to use computer RAM from 2000 through 5FFF to contain the data. Read the first 2764:

R2000,3FFF,0,5

Now read the second:

R4000,5FFF,0,5

Now program the 27128A:

P2000,5FFF,0,6,6

That's all there is to it.

Suppose you wanted to read a COMMODORE Kernel ROM, make some changes, then put the altered data on a 2764A. Here's how to do it: The Kernel ROM in the C64 is compatible with the 68764 (8k) EPROM. The control word is 30. Read it in:

TT2000,3FFF,0,30

Use the M command to change the data in RAM as you wish it to be. Now program the revised data into the 2764A:

P2000,3FFF,0,6,7

## **PROGRAMMING ERRORS**

While programming an EPROM, PROMOS checks to see that the job is being done correctly. If an error is detected, programming is stopped and you are advised what happened by messages on the screen. For suggestions on determining the source of the trouble, see the section on troubleshooting.

### **THE V COMMAND- Verify EPROM data.**

The V command compares the data on an EPROM with data in computer memory. It has two forms: the long form, with parameters exactly the same as in the R command; and the short form which is just V alone with no parameters at all. The short form uses the parameters from the most recent R or P command. The EPROM addresses of all discrepant locations, the checksum and the hash total are printed on the screen. Some examples:

V2000,3FFF,0,6

Here, we compare data from 2000 through 3FFF with the data on a 2764A.

P4000,BFFF,0,E6,6:V

Here, we have programmed a 27256 with data from 4000 through BFFF and have followed with the short form of the V command.

## **COPYING EPROMS**

The PROMENADE is a very efficient piece of production equipment. One can quite easily program 250 2764's in an hour. The usual procedure is to read an EPROM MASTER into RAM using the R command, then program the duplicates using the P command. It isn't necessary to type a new P command for each copy. Just move the cursor up to the P command line and press RETURN.

Here's a simple example of a command that programs 2764A's, verifies the data, keeps track of the number of copies that have been made and prints it on the screen:

R2000,3FFF,0,6:N=0

This reads the master and initializes the count. Now:

P2000,3FFF,0,6,V:N=N+1:?:?"TOTAL COPIES:"N

### **THE S COMMAND- Save computer RAM.**

This situation continually arises: You've read a ROM or an EPROM. Now you want to save the data on disk so that you can make a copy later. The S command is what you need. It causes a defined section of RAM to be saved to disk as a PGM file. An example:

S"EPROM DATA",8,4000,BFFF

This command takes data from 4000 through BFFF and stores it on disk as the program file called EPROM DATA. Experienced users will recognize this as the same S command their monitor programs have provided them for years. But NOTE one important difference. Monitor S commands require that you specify a range one byte longer than what you really want to save. This is the case with the C128's built-in monitor. With the PROMOS S command, you specify the range exactly.

There is one important restriction on the use of the S command in 64 mode: You cannot save from the address range D000 through DFFF. This restriction does not apply in 128 mode.

One further note for 128 users: If you want to save from bank 1, just use a prior B command to set bank 1 RAM first. Another example:

£2000,2FFF,0,18:S"2532 DATA",9,2000,2FFF

Here, we read a 2532 and save the data to disk on drive 9 as a PGM file called '2532 DATA'.

### **THE L COMMAND- Load program from disk.**

Now, you have the data on disk. How do you get that data back into the computer so an EPROM can be burned from it? The answer is the L command. An example:

L"2532 DATA",9

This command loads the PGM file '2532 DATA' back into the same locations in memory from where it was saved, namely 2000 through 2FFF, from device 9. On completion of the load, PROMOS reports the load start address and the load end

address so you know where in RAM the data is. Oftentimes, it's a good idea to jot these addresses down. As with the S command, you cannot load into the range D000 through DFFF from 64 mode. In 128 mode, if you want to load the file into bank one RAM, first set bank 1 with the B command.

There is one other important feature of the L command: you can specify your own load-start address. This is mandatory if you want to load a PGM file whose load-start address is unknown or if you want to load it someplace else. To specify your own load-start address, just add it to your L command:

```
L"2532 DATA",9,6000
```

The load-start address of this file we know to be 2000. But we have forced the data to be loaded at 6000 by means of this command.

This feature makes it very easy to merge separate PGM files into one.

### **THE Q COMMAND- Quit PROMOS.**

Should you wish to disengage PROMOS, use the Q command. Just execute:

```
Q
```

and PROMOS is gone. You can re-enable PROMOS by executing the following SYS:

```
SYS320
```

COMMODORE 128 users can switch to 64 mode using the G064 command, then re-enable PROMOS using SYS320 from 64 mode. After a RESET, you can use SYS320 to re-enable PROMOS in 128 mode without having to reload the program.

### **THE Z COMMAND- Zero the PROMENADE programming socket.**

This command sets the voltage at all pins of the C1 socket to (nearly) 0 volts so EPROM's can be safely installed and removed. This is done automatically for the most part. **C128 users will notice that after a 128 mode LOAD or SAVE, the red light is left on, indicating an energized condition.** Here is where the Z command comes in. Execute this command to reset the C1 socket.

## SPECIAL EPROM's

There are EPROM's the PROMENADE can read and program but which require special methods. Among these are the 27512, the 27513 and the 27011. We now consider each of these in turn.

The 27512 is the standard 64k byte EPROM. It has 16 address pins as can be seen in FIGURE 1. The 16th address bit is applied to pin 1. The PROMENADE cannot use pin 1 as an ordinary address pin. Rather, it is normally pulled high by a pull-up resistor. Some other means must be used to bring this pin low for reading and programming the lower 32k of the 27512.

A crude, but simple and effective way to do this is to take a short length of 22 gauge stranded hook-up wire and ground one end by securing it under one of the PROMENADE cover mounting screws. To access the low 32k addresses on the 27512, insert the free end of your grounding wire into the programming socket along with pin 1 of the 27512, (Pin 1 is the one closest to the operating lever). To read or program the upper 32k, simply remove the grounding wire.

Jason-Ranheim Company can install a switch to handle this more conveniently for a nominal charge. Contact our sales department.

The 27513 and the 27011 are 'page-addressed' EPROMs. They have an internal bank switching register which selects one 16k 'Page' of the EPROM at a time. The 27513 looks to the outside world like four 27128A EPROM'S; the 27011 looks like eight.

To read or program the 27513, use the control word 4XE2 where you substitute for X the number from 0 to 3 of the page you wish to read or program. For example:

R2000,5FFF,0,40E2

will read the lowest 16k page of the 27513 into RAM. Another example:

P4000,7FFF,0,43E2,6

will program page 3 of the 27513.

The 27011 works the same way. Use the control word 4X06, where X is the number from 0 to 7 of the page you wish to read or program. For example:

P8000,BFFF,0,4506,6

This command programs page 5 of a 27011 with data from computer RAM from 8000 through BFFF.

Again:

£A000,DFFF,0,4706

reads page 7 into the specified range.

### **USING PROMOS COMMANDS IN A BASIC PROGRAM**

PROMOS commands can be used in a BASIC program just like any regular BASIC command. Of course, PROMOS must be loaded first and active or your BASIC program will not run.

Some of the commands work a little differently from a running program as compared with direct mode:

First, control messages such as 'SEARCHING FOR', 'PROGRAMMING', etc. are not printed unless you enable them by setting bit 7 at address 9D high. See the PROGRAMMERS REFERENCE GUIDE for details.

Those commands which from direct mode report results on the screen, such as R, P, L, S, V, C, H, and E, do not do so when executed from a running program. Rather, certain results are made available in low RAM locations as detailed below. Your BASIC program can 'PEEK' these locations to make use of this information.

In general, address A5 (165 decimal) reports errors. If an error is detected, A5 will contain FF, otherwise it will be 0. A6 and A7 will report an address in low-high order, with the significance described below. A8 and A9 report the appropriate checksum and hash total respectively of the data in the address range involved in the command.

L command	A5- 00 = Load ok. FF = load failed. A6,A7- Address of highest byte loaded. A8- Checksum for data loaded. A9- Hash total for data loaded.
S command	A5- 00 = Save ok. FF = Save failed. A6,A7- Address of highest byte saved. A8- Checksum for data saved. A9- Hash total for data saved.
R (or £)	A6,A7- Address of highest EPROM byte read. A8- Checksum for data read. A9- Hash total for data read.
P (or TT symbol)	A5- 00 = PGMinig ok. FF = PGMinig failed. A6,A7- EPROM address at failure. A8- Checksum for data programmed. A9- Hash total for data programmed.

V command	A5- 00 = verify ok. FF = verify failed. A6,A7- EPROM address at failure.
C command	A5- 00 = COMP ok. FF = COMP failed. A6,A7- RAM address at lowest failure. A8- Checksum of compared data. A9- Hash total of compared data.
H command	A5- 00 = No occurrence. FF = Bytes found. A6,A7- RAM address of first occurrence.
E command	A5- 00 = EPROM erased. FF = Not erased. A6,A7- EPROM address of lowest un-erased byte.

Note that the C command generates a checksum and a Hash total. You can use a 'dummy' C command to generate these quantities for any range of RAM data. For example:

C2000,DFFF,2000

compares RAM from 2000 through DFFF with itself. In the process, the checksum and hash total over that range are generated and reported.

### **SIMPLE SAMPLE PROGRAM**

```
10 PRINT"INSTALL 27256 IN PROMENADE SOCKET."
20 PRINT:PRINT"TYPE A CARRIAGE RETURN WHEN READY."
30 GETA$:IFA$<>CHR$(13)THEN30
40 £4000,BFFF,0,E6
50 S"27256 DATA",8,4000,BFFF:ETC.
```

### **USING BASIC VARIABLES**

PROMOS commands would be of little utility in a BASIC program if they could not make use of variable data. The PROMOS L and S commands accept a string variable for the file name. Any numeric parameters that you want to express as a variable- addresses, control words, device numbers, etc. MUST USE the INTEGER variable form. These are numeric variables with a '%' sign suffix. Regular floating point variables will not work. REMEMBER, numeric BASIC variables are always DECIMAL quantities. Here's a simple example of the use of variable quantities in a program:

```
10 NM$="MY FILE":DN%=8:LS%=8192:REM DEFINE PARAMETERS
20 LNM$,DN%,LS%:REM LOAD "MY FILE" STARTING AT $2000 FROM DEVICE 8.
30 ES%=0:CW%=5:PM%=7:REM DEFINE PARAMETERS FOR PROGRAMMING.
30 LE%=PEEK(166)+256*PEEK(167):REM CALC LOAD END ADDRESS.
40 PLS%,LE%,ES%,CW%,PM%:REM PROGRAM 2764 EPROM.
```

```

50 IFPEEK(165)=255THEN4000:REM REPORT PROGRAMMING ERROR.
60 PRINT"PROGRAMMING OK."
70 V:REM VERIFY ONCE AGAIN.
80 IFPEEK(165)=255THEN4020:REM REPORT VERIFY ERROR.
90 PRINT"VERIFY OK.":ETC, ETC....

```

### SETTING UP A RAM BUFFER

Let's say you're using a 64 mode BASIC application program and you want to set aside a large RAM buffer for moving data to and from EPROM's and disk that you're sure BASIC will not intrude upon. Here's a suggestion. Early in your program, execute the following BASIC statement:

```
10 POKE55,0:POKE56,64:CLR
```

This confines your BASIC program to the 14k from 0801 through 3FFF. Now, you have 32k from 4000 through BFFF for your RAM buffer; and 8k from C000 through DFFF to use as a utility area for ML programs, etc.

### ONE FURTHER IMPORTANT POINT

A decimal address of 32768 must become a negative number when expressed as a numeric variable. This number will decrease to -1 at 65535. This is of concern only for addresses above 7FFF. Examination of the following table will give you an idea of how to handle this.

DESIRED HEX ADDRESS	DECIMAL VALUE EQUIVALENT	NUMERIC VARIABLE EQUILAVENT
1000	4096	4096
2000	8192	8192
4000	16384	16384
7FFF	32767	32767
8000	32768	-32768
8FFF	36863	-32673
9000	36864	-32672
9FFF	40959	-24577
A000	40960	-24576
FFFF	45055	-20481
B000	45056	-20480
BFFF	49151	-16385
C000	49152	-16384
CFFF	53247	-12289
D000	57248	-12288
DFFF	57343	-8193



Here's a BASIC statement that converts a DECIMAL address (DA) from 0 to 65535 into the necessary NUMERIC VARIABLE equivalent (NV%):

```
100 NV%=DA+65536*(DA>32767)
```

### **WHERE'S PROMOS?**

As you use the M command to browse through memory, you come across nothing resembling a sizable machine language program. So where's PROMOS? It's tucked away in RAM under the Kernel at E000. Operating from there, it leaves everything else available to you. There are a few bytes of code well down on the stack which take care of switching things in and out. In 64 mode, PROMOS hooks in through IGONE. In 128 mode, ICRNCH is taken over as well.

### **TROUBLESHOOTING**

The frustrations that can arise programming balky EPROM's can often be avoided by making sure of a few things:

Make sure your EPROM is erased. Use the D, A or E commands for a quick check.

Make sure your EPROM is seated properly in the socket and is making contact at all pins.

Make sure your programming command is correct. Especially check the CONTROL WORD. If you use an incorrect one you can destroy your EPROM.

Make sure your PROMENADE is making good contact with the user port connector. The board-edge surfaces may need cleaning. An easy way to do this is to go over them lightly with a pencil eraser.

Make sure your power supply is in good shape. Commodore 64 power supplies have a tendency to get tired after a while.

You may wish to try a modified programming method that turns off Vpp during read-back as discussed in Appendix B.

## APPENDIX A- THE CONTROL WORD

The control word determines the way the PROMENADE works in dealing with a particular EPROM. The PROMOS 2.0 CW is actually a 16 bit quantity. The low byte is particularly important. The low eight bits control things as follows:

BITS 1,0	00- Set Vpp to 25 volts.
	01- Set Vpp to 21 volts.
	10- Set Vpp to 12.5 volts.
	11- Set Vpp to 5 volts.
BITS 3,2	00- Apply Vpp to pin 22.
	01- Apply Vpp to pin 1.
	10- Apply Vpp to pin 23.
	11- Set pin 1 low when Vpp is on.
BITS 5,4	00- PGM pulse at pin 27.
	01- PGM pulse at pin 22.
	10- PGM pulse at pin 20.
	11- No PGM pulse required.
BIT 6	0- No action at pin 20 on read.
	1- Invert pin 20 level on read.
BIT 7	0- Set standby level on pin 20 low.
	1- Set standby level on pin 20 high.

The upper byte works as follows:

BITS 5,4,3,2,1,0	Select particular page of page mode EPROM. Up to sixty four 16k pages.
BIT 6	0- Page mode indicator off.
	1- Page mode indicator on.
BIT 7	Not used.

## APPENDIX B- THE PROGRAM METHOD WORD

The program method word tells PROMOS how you want the programming done. You have some latitude. If you check the TABLE of RECOMMENDED CONTROL and PROGRAM METHOD WORDS inside the back cover, you'll see a 'standard method' and three 'intelligent' methods. We examine these in greater detail here.

### STANDARD PROGRAMMING

In standard programming, the programming pulse applied to the EPROM is of constant duration, usually about 50 ms (milliseconds). This is always a time-consuming method. A 2764 takes about 7 minutes to program this way. For PHW's from 0 to 3, PROMOS uses Pulse duration's as follows:

PMW	PULSE DURATION
0	6 ms.
1	12 ms.
2	24 ms.
3	48 ms.

### INTELLIGENT METHODS FOR PROGRAMMING

Intelligent methods for programming EPROM's have been developed which can greatly reduce the time required. These methods in effect 'test' the EPROM as the process goes on. It works like this: The address and data are set up and a short duration pulse is applied. The EPROM is then read back and the data is compared. If the data doesn't 'verify', then another pulse is applied and so on. Programming time is accumulated. At some point the data 'sticks' and a positive verification is obtained.

At this point, another longer 'margin' pulse is applied which is proportional in duration to the sum of the preceding pulses. The data is verified once again, and the programmer then moves on to the next location.

Intelligent #1- Developed by Jason-Ranheim, this method has been widely copied. It is the fastest way to program an EPROM. Programming starts with a short .1 ms pulse which doubles in duration with each failure to verify. The margin pulse is equal in length to the sum of all preceding pulses.

Intelligent #2- In this method, short pulses of equal duration are applied. The margin pulse is 3 times the total accumulated pulse time. If after 25 ms of short pulses, a location still does not verify, programming stops and the failure is reported.

Intelligent #3- In this method, the margin pulse is 4 times the preceding total accumulation. If after 15 ms of pulses verification has not been achieved, a 60 ms pulse is tried. If the location now verifies, programming moves on. If not, the EPROM has failed.

For any of the Intelligent programming methods, the EPROM Supply (VCC) is set to 6 volts.

PMW's from 4 to F have the meanings given in the following table:

PMW INT METHOD TEST PULSE TOTAL ACCUMULATION\*

4	1	variable	12 ms.
5	1	variable	25 ms.
6	1	variable	50 ms.
7	1	variable	100 ms.
8	2	.25 ms.	75 ms.
9	2	.50 ms.	75 ms.
A	2	1.0 ms.	75 ms.
B	2	2.0 ms.	75 ms.
C	3	.25 ms.	100 ms.
D	3	.50 ms.	100 ms.
E	3	1.0 ms.	100 ms.
F	3	2.0 ms.	100 ms.

\* = the maximum time that can be accumulated at a location before an error is reported.

If you add \$40 to the PHW, that is set bit 6 high, the programming is altered in this respect: The PROMENADE turns off Vpp before attempting to read back the EPROM. This slows down programming; but it is advisable in few cases. For example:

PC000,C7FF,0,28,47

This command programs a 27C16 with data from C000 through C7FF. Another example:

P4000,BFFF,0,E6,46

This programs an OKI 27256 with the specified data.

\*\*\*\*\*

This scanned document is not intended to defraud the original author or infringe on any copyrights. It was only done to support an otherwise unsupported Commodore accessory. Since the manufacturer no longer builds, sells or in any way profits from this hardware or software, it is up to the users themselves to help each other with whatever it takes to keep a useable product alive as long as possible. It is in that spirit of co-operation that this document is made available to end users at no charge.

\*\*\*\*\*

## APPENDIX C

### PROMENADE C1 with PROMOS 2.0 Operating System Software

**TABLE of CONTROL and PROGRAM METHOD WORDS**

EPROM TYPE	PGM VOLTAGE	CONTROL WORD	PROGRAM METHOD WORDS			
			STD	I#1	I#2	I#3
2758	25	28	3	7	A	E
2516	25	28	3	7	A	E
2716	25	28	3	7	A	E
27C16	25	28	43	47	4A	4E
2716B	12.5	2A	NR	46	4A	4E
2532	25	18	3	7	A	E
2732	25	E0	3	7	A	E
2732A	21	E1	3	7	A	E
2732B	12.5	E2	NR	46	4A	4E
2564	25	14	3	7	A	E
2764	21	5	3	7	A	E
27C64	21	5	3	7	A	E
2764A	12.5	6	NR	6	A	E
27128	21	5	3	7	A	E
27128A	12.5	6	NR	6	A	E
27256	12.5, 21	E6, E5 *	NR	6	A	NR
27C256	12.5, 21	E6, E5 *	NR	6	A	NR
27512	12.5	E2 **	NR	6	NR	NR
27C512	12.5	E2 **	NR	6	NR	NR
27513	12.5	4XE2 **	NR	6	NR	NR
27011	12.5	4X06 **	NR	6	NR	NR
68764	25	30	NR	46	4B	4F
68766	25	30	NR	46	4B	4F
68769	25	30	NR	46	4B	4F
2815	21	39	3	6	B	F
2816	21	39	1	7	NR	NR
X2804A	21	39	1	NR	NR	NR
X2816A	21	39	1	NR	NR	NR
X2864A	21	7	1	NR	NR	NR
48016	25	28	2	5	9	D
5133	21	5	3	NR	NR	NR
5143	21	5	3	NR	NR	NR
5213	21	39	2	4	9	D
52B13	5	7	NR	4	NR	NR
52B23	5	7	NR	7	NR	NR

NR- NOT RECOMMENDED.

\*- Be SURE to use the CORRECT CW corresponding to the PGM VOLTAGE required by your EPROM.

\*\*-. Requires special programming. See manual for details.

## APPENDIX D

COMMAND	EXAMPLE
A EPROM ASCII DUMP	A0,1FFF,5
B SELECT RAM BANK	B1
C COMPARE RAM	C2000,DFFF,E6
D EPROM HEX DUMP	D4000,5FFF,E6
E EPROM ERASE CHECK	E0,3FFF,6
F FILL RAM	F2000,DFFF,FF
G GLOSSARY	G
H HUNT FOR BYTES	H2000,9FFF,20,D2,FF
I ASCII RAM DUMP	IA000,A3FF
L LOAD FROM DEVICE	L"A\$",8,2000
M HEX RAM DUMP	M0,1FF
P PROGRAM EPROM	P4000,BFFF,0,E5,7
Q QUIT PROMOS	Q
R READ EPROM	R2000,5FFF,0,6
S SAVE TO DEVICE	S"DATA",8,4000,BFFF
T TRANSFER RAM	T2000,3FFF,4000
V VERIFY EPROM	V4000,5FFF,0,6
Z ZERO PROMENADE	Z
£ SAME AS R	£4000,BFFF,0,E6
π SAME AS P	πC000,DFFF,0,6,A

Above is the G COMMAND screen display

TYPE	CAPACITY	HIGHEST ADDRESS
2758	1k	03FF
2716	2k	07FF
2732	4k	0FFF
2764	8k	1FFF
27128	16k	3FFF
27256	32k	7FFF
<b>SPECIAL EPROMS</b>		
27512	64k	FFFF
27513	4x16k (64k)	3FFF in 4 pages (0.5meg)
27011	8x16k (128k)	3FFF in 8 pages (1meg)

## APPENDIX E

### PARTS LIST

#### IC'S

2EA	U1,U2	LM723	14 PDIP	VOLTAGE REGULATOR
1EA	U3	LM324	14 PDIP	QUAD OP AMP
1EA	U9	74LS00	14 PDIP	2-INPUT NAND GATE
1EA	U6	7406	14 PDIP	HEX INV BUF W/ HV/OPEN COLLECTOR
1EA	U4	74LS86	14 PDIP	QUAD 2-INPUT EXCLUSIVE OR-GATE
1EA	U7	74LS139	16 PDIP	DECODER/DEMULTIPLEXER
3EA	U5,U8,U10	74LS374	20 PDIP	OCTAL D-TYPE FF W/TRI-STATE

#### DIODES

13EA	D1-D5,D9-D16	1N4002	100V 1A RECTIFIER DO-41
1EA	D8	1N4739	9.1V 1 WATT ZENER DO-41 *
2EA	D6,D7	1N4740	10V 1 WATT ZENER DO-41 *

#### LED'S

1EA	LED1	3MM	GREEN
1EA	LED2	3MM	RED
1EA	LED3	3MM	AMBER/YELLOW

#### CAPACITORS

1EA	C3	220 $\mu$ F,25V	ELECTROLYTIC RADIAL
1EA	C2	330 $\mu$ F,25V	ELECTROLYTIC RADIAL
2EA	C1,C4	470 $\mu$ F,16V	ELECTROLYTIC RADIAL *
7EA	C5,C7,C9	0.1 $\mu$ F,100V	CERAMIC RADIAL, 0.1" SPACING
	C10,C11,C14,C16		
5EA	C6,C8,C12	.001 $\mu$ F,100V	CERAMIC RADIAL, 0.1" SPACING
	C13,C15		

#### RESISTORS

2EA	R1,R2	3.0 $\Omega$ ,5%, $\frac{1}{4}$ W	AXIAL LEAD
6EA	R3,R4,R5	330 $\Omega$ ,5%, $\frac{1}{4}$ W	AXIAL LEAD
	R7,R9,R10		
4EA	R8,R19,R21	3.9K $\Omega$ ,5%, $\frac{1}{4}$ W	AXIAL LEAD
	R23		
1EA	R6	10K $\Omega$ ,5%, $\frac{1}{4}$ W	AXIAL LEAD
7EA	R15,R16,R17	47K $\Omega$ ,5%, $\frac{1}{4}$ W	AXIAL LEAD
	R18,R20,R22,R24		
1EA	R12	4.53K $\Omega$ ,1%, $\frac{1}{4}$ W	AXIAL LEAD
1EA	R13	7.87K $\Omega$ ,1%, $\frac{1}{4}$ W	AXIAL LEAD
1EA	R14	21.5K $\Omega$ ,1%, $\frac{1}{4}$ W	AXIAL LEAD
1EA	R11	43.2K $\Omega$ ,1%, $\frac{1}{4}$ W	AXIAL LEAD

#### SOCKET

1EA 28 PIN ZIF

#### CONNECTOR

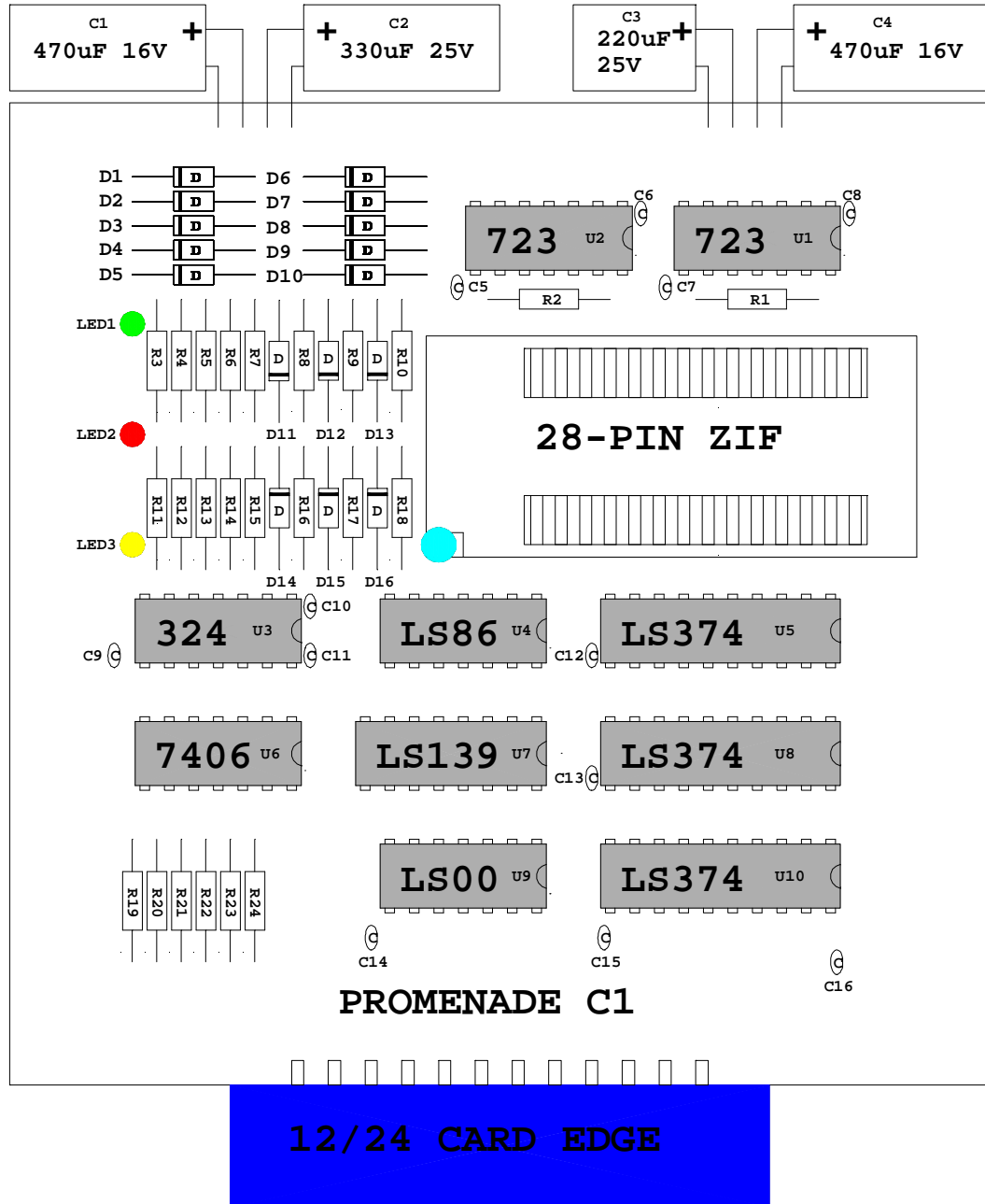
1EA 12/24 CARD EDGE FEMALE

\* = SEE PAGE FOLLOWING THE BLOCK DIAGRAM



## APPENDIX F

### BLOCK DIAGRAM



The ZENER diodes (1N4739 and 1N4740 rated at 1W) get hot during operation. I have replaced the diodes on my programmer with a higher rated diode (1N5346B and 1N5347B rated at 5W). Also raise them up off the board as much as possible, the leads act as heat sinks.

The 470 $\mu$ F 16V electrolytic is under rated. I have replaced mine with a 470 $\mu$ F 25V electrolytic.

The PCB layout labeled OFFSET was created for use on the C128/128D. It will allow the programmer to be inserted without having to remove the RGBI connector. This will allow the use of the 80-COL screen.

The OFFSET layout uses a right angle edge card connector. That is the only difference (in parts) from the original programmer.

In the SPECIAL EPROM's section of the manual (page 17) paragraphs 2 and 3, explains how to program a 27512 EPROM. In order to read/program the lower half (32K) of the EPROM you must manually ground pin 1 at the programming socket. To read/program the upper half you must remove the ground, which is the normal programmer state.

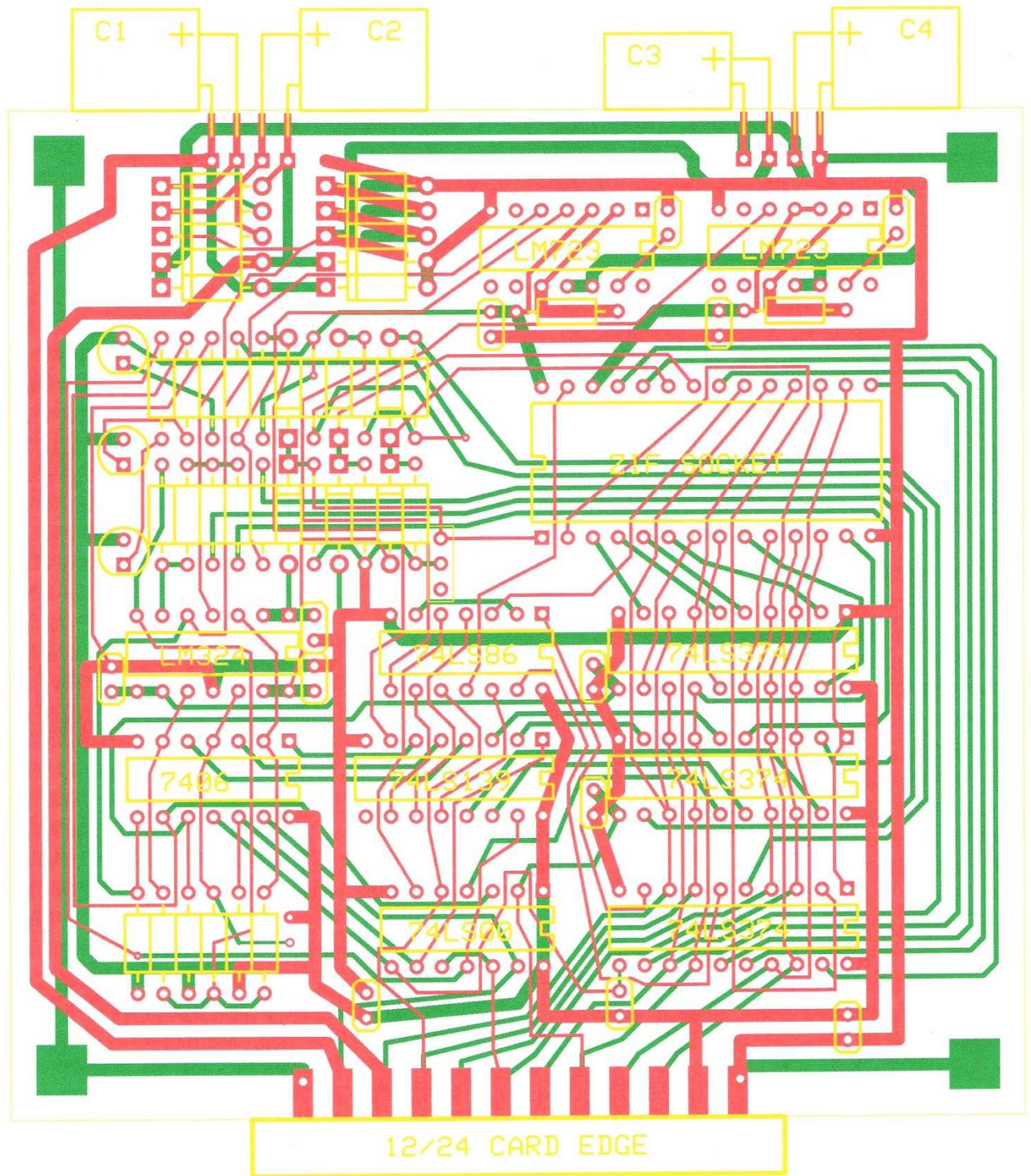
Notice on the schematic near the ZIF socket, there is a jumper (J2) that will ground pin 1 when needed. It has been added to the Printed Circuit Board layout also in case you make your own PCB.

If you have any questions you can reach me at:  
EMAIL: [newbury@planetkc.com](mailto:newbury@planetkc.com)

Daniel C. Newbury

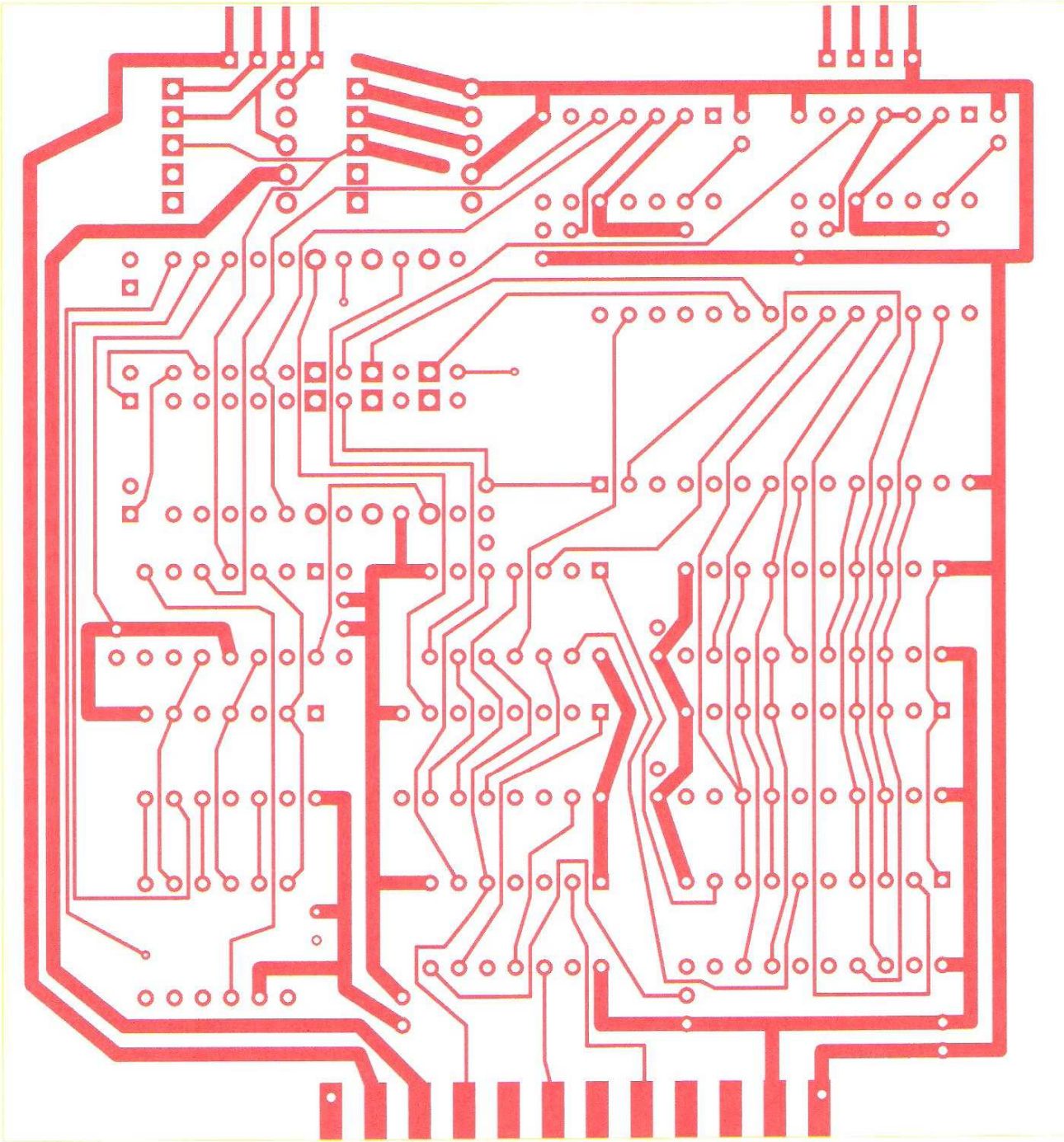
## APPENDIX G

Top and Bottom View

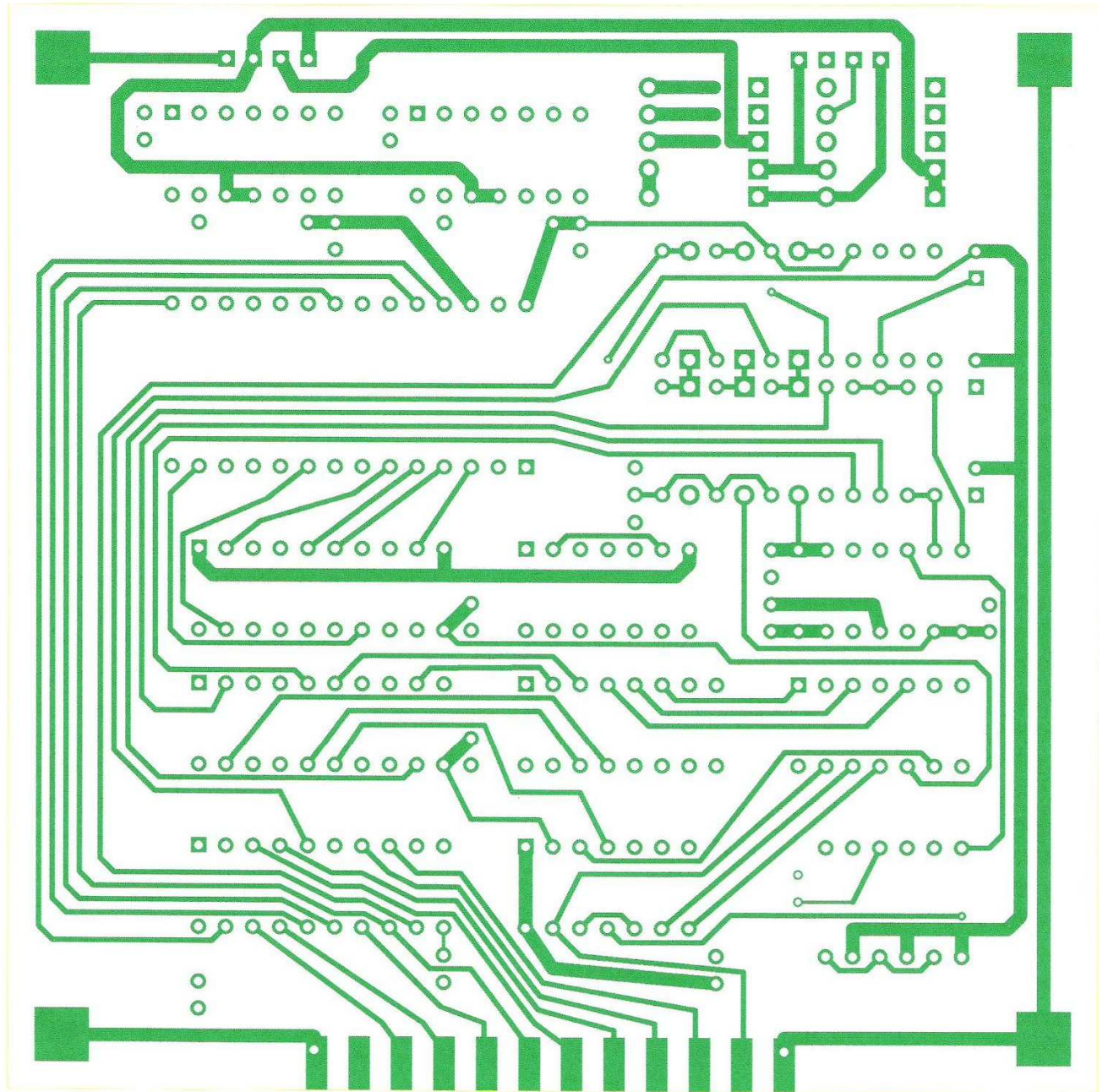




Top Trace View

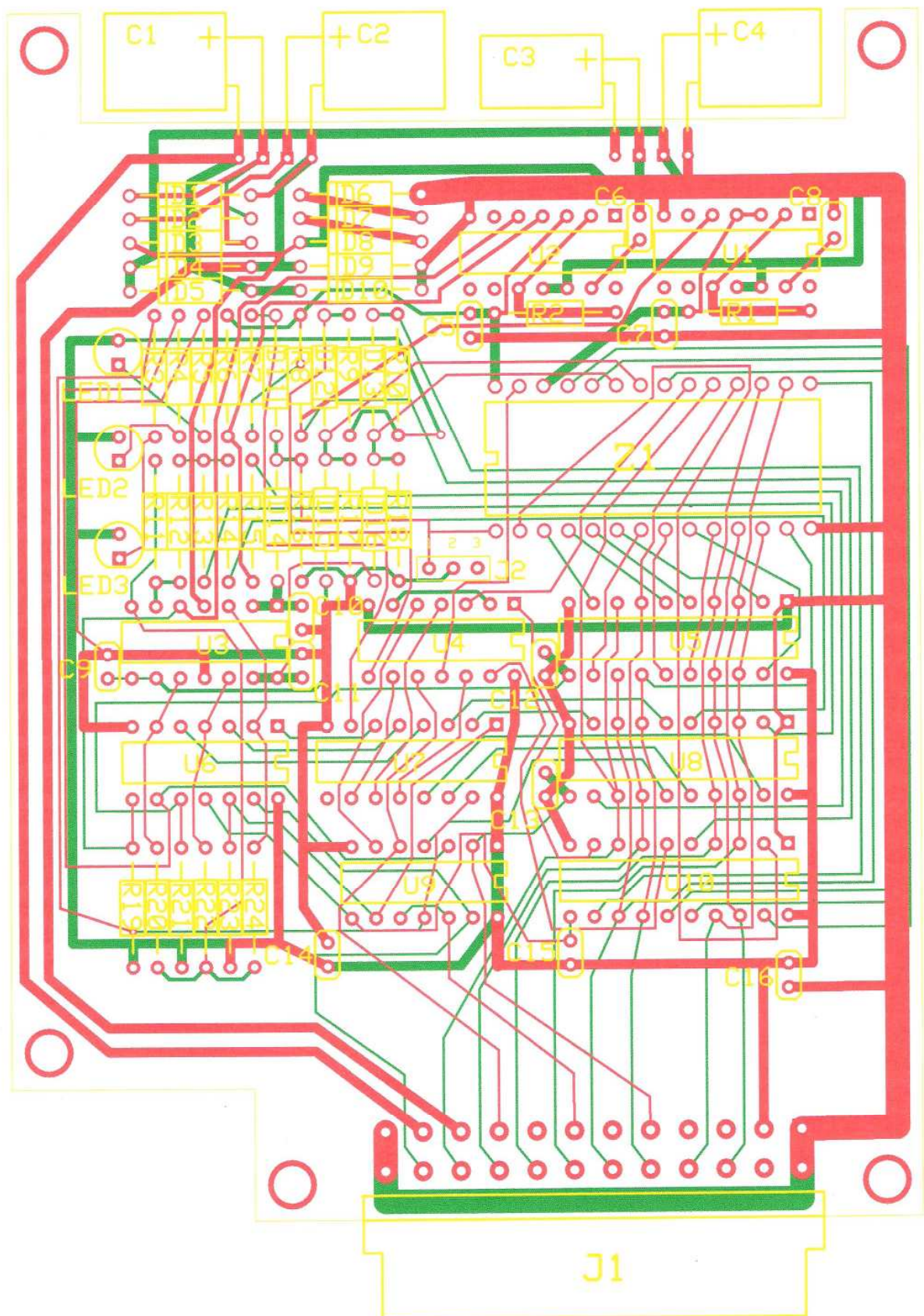


Bottom Trace View as seen from Bottom

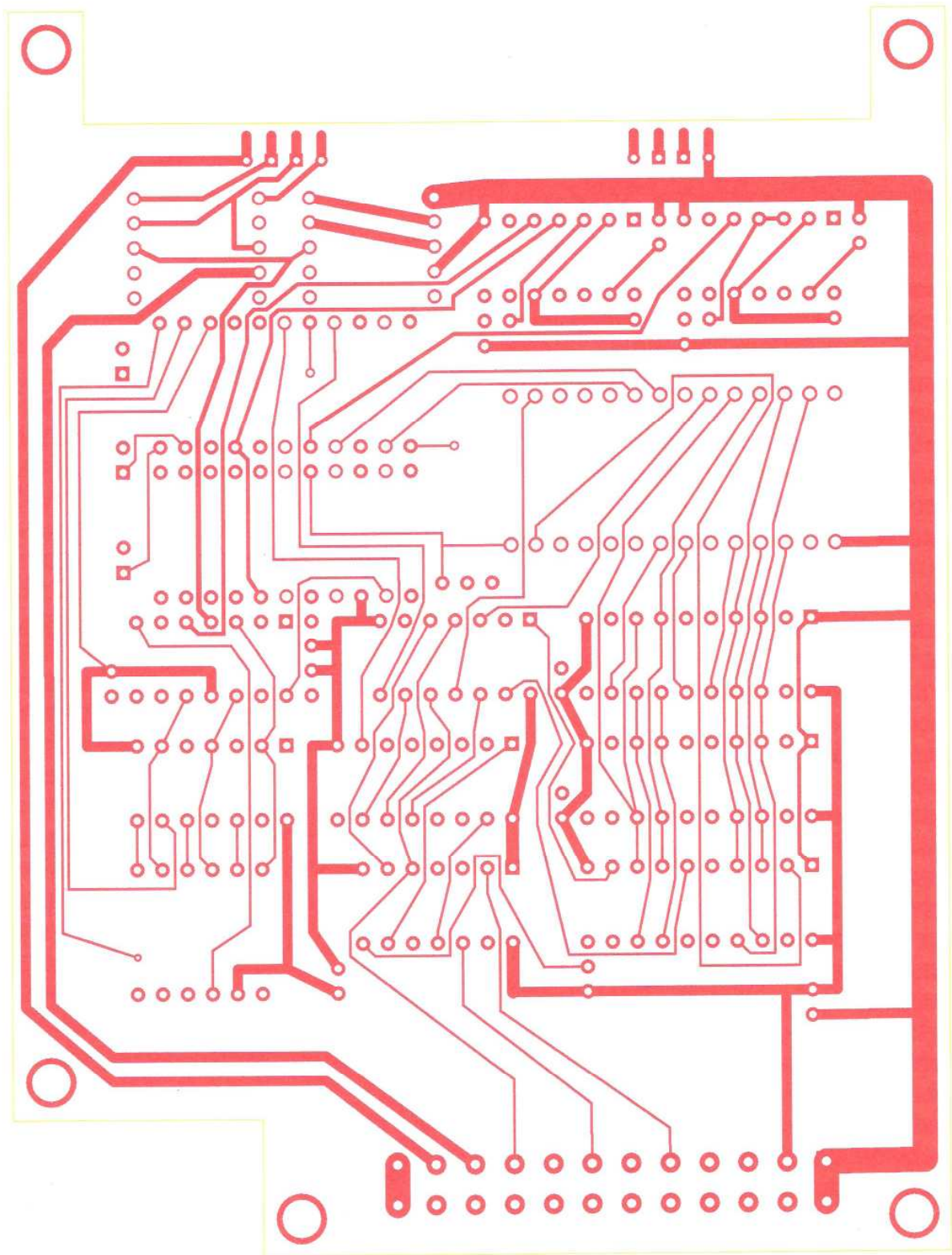




Top and Bottom View with C128 Offset

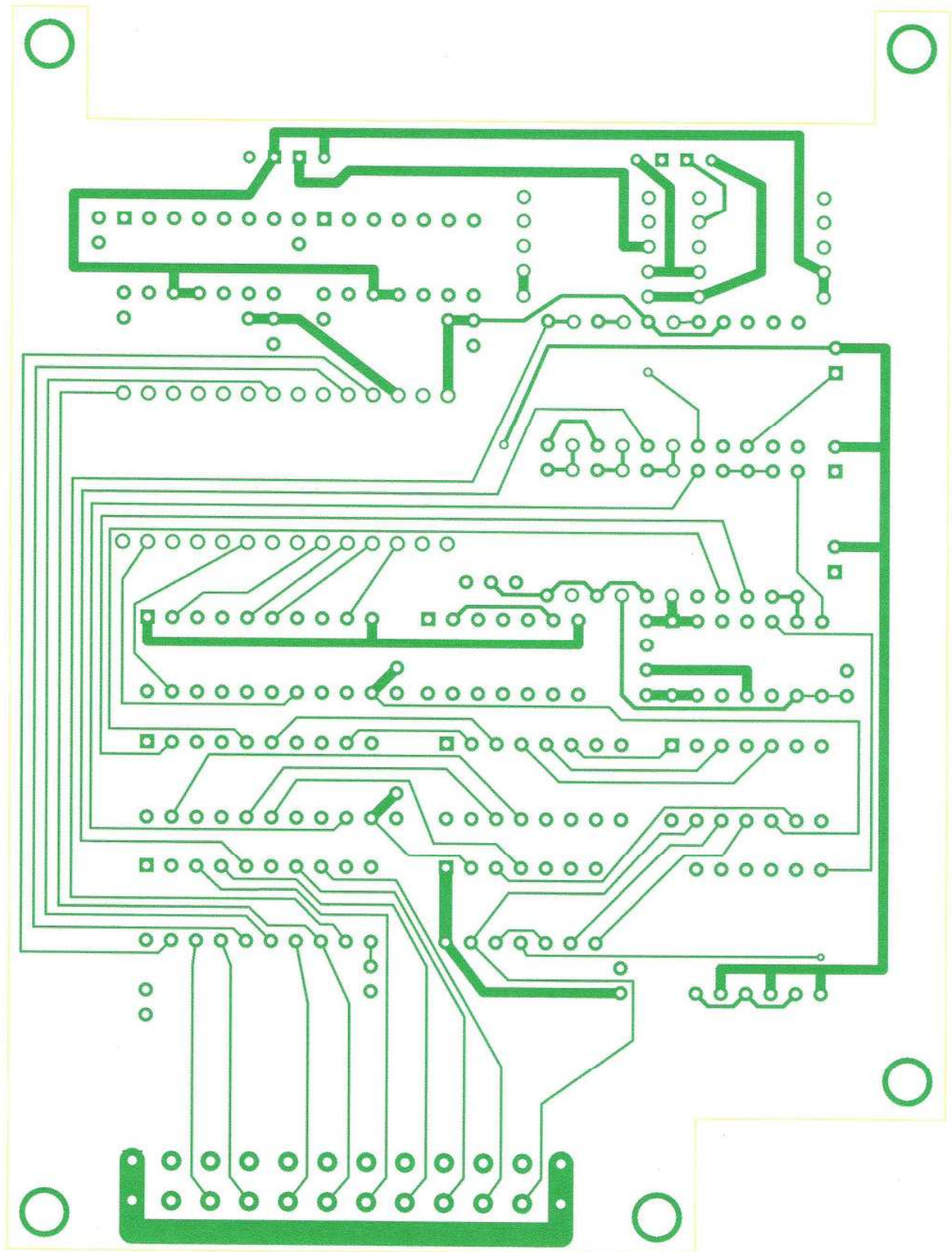


Top Trace View with C128 Offset





Bottom Trace View as seen from Bottom with C128 Offset





Page # 1

