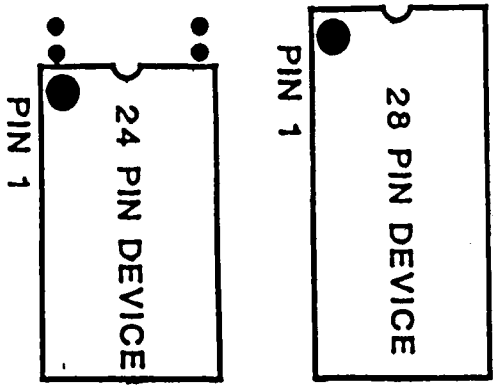


Figure 1

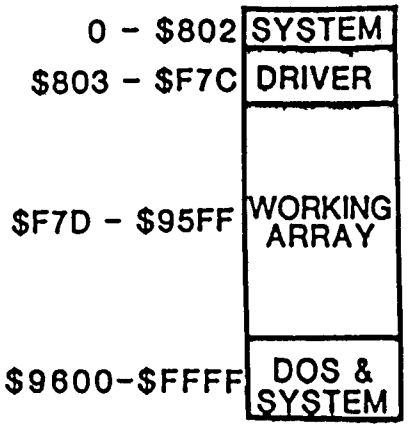


Figure 2

EPROM TYPE	LENGTH (in hex)	LENGTH (in dec.)	NORMAL PROGRAM (min. and sec.)	FAST BURN (min. & sec.)
2716	800	2048	1:42	0:13
2732	1000	4096	3:24	0:25
2764	2000	8192	6:50	0:50
27128	4000	16384	13:40	1:40
27256	8000	32768	27:18	3:20
27512	10000	65536	54:36*	6:40*

* plus the time needed to load 2nd bank

Table 1

PROmGRAMER™

INSTRUCTIONS

INTRODUCTION

If you understand EPROMs and the mechanics of programming, set the switches, boot the disk, and ignore the rest of these instructions. If you are in a rush to use the PROmGRAMER, or are not interested in the technology behind this product, you may skip the next section, and GOTO "SETTING THE SWITCHES".

WHAT IS AN EPROM

Taking our APPLE as an example, we know that some of its memory is volatile, that is, when we turn off the power, the computer "forgets" what it had in it. This type of memory is called RAM (Random Access Memory), or, more correctly, R/W (Read-Write) memory, since you can write information into it, and read it out again. However, there is some memory that is not volatile. We know that as soon as we turn on the computer, some programs are available to us, even without a disk. Examples of these programs include APPLESOFT and the system monitor. These programs are stored in a non-volatile memory called ROM (Read Only Memory). [We leave it as an exercise to the reader to figure out what a WOM (Write Only Memory) is good for].

A major problem with ROMs is that they are manufactured with the program in place, and it is impossible for the user to change the memory. The cost to make just one is in the vicinity of $3,000 (U.S.). Obviously, this is too high to be practical, unless you make many of them, since each additional one is only a few dollars.

Along came the PROM (Programmable Read Only Memory). This device is manufactured with many tiny fuses in it. Where a fuse exists, a logical ONE would be read. To change it to a logical ZERO, a high pulse of current is sent through the fuse, melting it. The obvious problem with this is that you can program the device once. If you make a mistake, buy another one, and try again.

Which brings us to the EPROM (Erasable Read Only Memory). This device is also manufactured with each "cell" reading a logical ONE. If you want a logical ZERO, you have to program it in. A rigorous explanation is beyond the scope of these instructions, but, in short, this is done by "pumping" a charge into an insulator. This is the job of the PROmGRAMER. The wanted address and data are set up and a relativly high voltage is applied to the chip. This causes an insulator to temporarily change into a conductor. A charge of electrons is injected into the middle of the conductor, and, when the high voltage is removed, these electrons now find that they are trapped, surrounded by an almost perfect insulator. This "floating" charge affects circuitry which "sees" a logical ZERO where such charge exists.

OTHER NOTES OF INTEREST

If you EXIT the program to APPLESOFT, you may restart it by typing "&", and RETURN. If you go to the MONITOR, you may restart by typing 803G, or by typing CONTROL Y, then RETURN.

At any time, you may abort listings or programming by typing ESC or RETURN. You may pause listings or programming by pressing any other key, and pressing a key again to continue.

You may ERASE CHECK a portion of an EPROM. For instance, typing E 400 7FF will check the indicated bytes.

You may issue a PR#1 command, to turn on a printer.

The source code for the driving program is included on the disk. It is written for the S-C Assembler (available from S-C Software, Post Office 280300, Dallas, TX 75228. Telephone (214) 324-2050). You may experiment with, and modify the program.

Although the source code and driving programs are copyrighted, the disk is not copy-protected. You are urged to make back-up copies, for your own use, of course.

On the off chance you misplace these directions, a copy is on the disk, in an APPLEWRITER text file.

While DOS commands can be used from within the program, you will have to remember that LOADing or RUNning a program will destroy the driving program, as will BLOADing or BRUNning a program between $803 and $F7C.

To maintain compatibility with earlier versions of this program, V (for verify) will do the same as C, K = E, and R = L

PROGRAMMING THE 27256

For some reason, most people who get into programming EPROMs start their files at $2000. That is a nice round number, it won't interfere with the PROmGRAMER driving program, and it does seem to be a standard, so why not continue to use it? Well, we'll tell you why. If you start at $2000, the file for a 27256 will end at $9FFF. We all remember (don't we?) that DOS resides starting at $9600, so we have a conflict. In this type of conflict, no one wins. We must avoid overwriting DOS. If we start the file at $1000, it will end at $8FFF neatly avoiding problems, so let's set a standard: All files will start at $1000 (naturally, you may start at another location if it is necessary for a particular job).

For programming Toshiba EPROMS, see "DIFFERENT EPROMS, page 2.

One other thing; under DOS 3.3 the maximum file length you can save to disk is $7FFF bytes, one byte less than that needed for a 27256 (amazing how often things like that seem to happen). We can do one of two things, save one byte less than we need, or change DOS. If you type POKE -22172,255, DOS will now accept files up to 64 Kbytes long. The "HELLO" program on the disk does this for you. We suggest that you do not INIT any disks after making this change, because your DOS is no longer standard (until you re-boot).

as shown in figure one. Pin #1 must be toward the top of the ZIF socket. There is a note at the top of the card reminding you of this, and two dots on the left side of the socket showing the location of pin #1. Note that if you have a 28 pin device (2764, 27128, 27256, or 27512), pin #1 is on the upper left side of the ZIF socket. If you have a 24 pin device (2708, 2716, or 2732), pin #1 is the third pin down from the top on the left of the ZIF socket. Thus, if you have a 24 pin device in the socket, there will be two holes visible above the EPROM on each side of the ZIF socket.

INSTALLATION

With power off, plug the PROmGRAMER into any slot (except zero) of your APPLE][,][+, or //e. Turn on the power, boot the disk, and the "HELLO" program on the disk will load the driver program (B.PROMGRAMER). NOTE: If you have not yet done so, make a copy of the disk, and put the original in a safe place. You may use any standard copy program.

PROGRAMMING THE EPROM

Programming the EPROM basically consists of taking an area of memory, and transferring it to the EPROM. The PROmGRAMER takes care of the mechanics, and it is up to you to make sure to get the correct data into the EPROM. We will show this by example, but first, a word about the driver program (the one that makes the PROmGRAMER work).

The program loads into memory at $803. (The dollar sign signifies that the number following is in hexadecimal). The length of the program is $77A, bringing you to $F7C. The rest of memory (up to $9600 where DOS begins) is available to you as the working array. A memory map is shown in figure 2.

Boot the disk, and the "HELLO" program will load the driver program into memory. The program will then ask you which slot the PROmGRAMER is in, and the type of EPROM you are using. (Type in a number between 1 and 8). You will then be presented with a menu of choices. Anthropomorphically speaking, you have to tell the computer which area of memory you are using (called the WORKING ARRAY). You do this by specifing the starting address (symbolized by SSSS), and the ending address (EEEE). You also have to tell the computer where you want to start programming the EPROM. You do this by giving is a relative address (PPPP) relative to the start of the EPROM, where the start is 0000. All these addresses are in hexadecimal.

To program an EPROM, you should have a binary file ready on disk with the necessary information. The length of the data you can fit in depends on the specific EPROM you are using. Table ONE on the back cover, shows EPROM types, the length of the file that can fit in, and programming times.

For our first example, let's try programming a 27128 with a unique pattern that tests every possible combination. Looking at the above table, we can see that the maximum length of the file is $4000 bytes. On the disk that we sent you is a program called "TEST PATTERN". This just consists of a sequence of numbers from 0 to $FF, repeating as necessary to fill the $4000 byte range. First, we set the switches, plug in the EPROM, and plug it into a slot (REMEMBER - POWER OFF!). Turn on the power, booting the

disk, and type the slot number.

Since we are using a 27128, type the number 5, which is the type number for a 27128.

DOS commands can be typed within the PROmGRAMER driving program, so we type;

BLOAD TEST PATTERN

This pattern will load starting at location $1000.

We will now "burn" the EPROM with the memory range from $1000 to $4FFF, starting at EPROM location 0, so we type:

B 1000 4FFF 0000

Note that it is necessary to put in the blank spaces, and four digits per number. (Actually, as we shall explain later, it is only necessary to type the letter "B", then return, as these parameters are the default numbers for the 27128).

Now, make yourself a cup of tea, and find something to pass the time. Each byte takes 50 milliseconds to program (0.05 seconds), so we have a wait of about .05 seconds times 16384 = 819.2 seconds, or 13 minutes 39.2 seconds. You can follow the progress of the programming, as the current EPROM address is printed on the lower right side of the screen.

After programming is complete, the PROmGRAMER will check if the job was done correctly. It will compare each byte of memory with the corresponding location in the EPROM, and verify that both are the same. If so, there will be a four note audio signal, and a visual message. If not, the computer will give you a six note audio signal, and start printing the addresses of the offending byte(s). You may abort the listing at any time by hitting the ESC or RETURN key.

While the PROmGRAMER is programming, there is a high voltage on one of the pins. As damage may occur if the wrong voltage is removed first, DO NOT insert or remove EPROMs while the PROmGRAMER is in the programming state. Since the programming voltage is removed in the read or idle state, there should be no problem inserting or removing EPROMs, even with power on. However, you MUST turn off power before removing the card, or dire consequences will follow.

READING EPROMS

Turn off the computer to assure yourself that the test pattern in memory has disappeared. Turn the computer back on, and initilize the program. Type 5 (the EPROM type), and;

L

This command will load memory locations $1000 through $4FFF with the information in the EPROM starting at EPROM location zero. To display the memory, type D. You should see the test pattern appear. (You may pause the listing by pressing any key (except ESC or RETURN), and continue it with another keypress). The information you are looking at was stored inside the EPROM. At this point, you can be sure that the PROmGRAMER and your technique are working correctly.

NORMAL OR FAST BURN?

You have a choice of two programming algorithms (an algorithm is the program that the computer uses.) The first, which we call "BURN", gives a 50 ms pulse for every address. This is the pulse length recommended by most EPROM manufacturers. Most EPROMs, however, will program sucessfully with a much shorter pulse. The "FAST BURN" algorithm will give a 2 ms. pulse, and read the byte back. If it does not read back correctly, it will get another 2 ms. pulse, and so forth until the byte reads back correctly. The PROmGRAMER will then give an "overprogramming" pulse lasting 4 ms. As most bytes will program with the first pulse, programming time is drastically reduced. For most EPROMs, however, this method is not guarenteed by the manufacturer (although off-the-record, most of the engineers say this method works fine.) It's up to you which method to use. Our recommendation is to use the FAST algorithm, unless it makes you uncomfortable. Incidentally, you will find that in almost all cases, the programming time will be shorter than that listed in the table, because both algorithms will skip any byte of $FF, as that is the same as an unprogrammed EPROM byte.

DEFAULTS

(Quick definition; a DEFAULT is a parameter chosen by the program, but can be overridden by the user.) Most of us will program EPROMs at the same address. Since it is a pain to have to enter a string of numbers each time we want to program an EPROM, the software has built-in defaults. These numbers are chosen to give the most utility to the most people. But we all know what happens when a manufacturer decides what's good for you. They are usually wrong. There will be times that you will not want to use our standard defaults. That's fine with us. The default parameters assume that you want to program the entire EPROM, from a file loaded at $1000. If you press "B", and return, the default parameters will be printed, just as if you had typed them in. If you wish to use different parameters, just type them in after the command letter. Note that the default parameter for the 27512 is for the lower half of the EPROM. To program the upper half, you will have to type in the parameters.

If you will be exclusively using a particular slot and/or EPROM, you may save a copy of the program with these as defaults, so you will not have to select the slot or EPROM type each time you start the program. The byte at $806 usually holds $00, which signifies that a choice of slot has to be made. You may put the slot number times 16 in this location. The EPROM type is in location $807, where $00 signifies that no choice has been made, or a value from $B1 to $B8. You may save these parameters by running the program, making your slot and EPROM choices, then type "BSAVE (name), A$803, L$77A. BRUNning this program will bypass the choices for slot and EPROM. If you wish to change the EPROM type, you may press "Z", which will restart the program.

These EPROMs are programmed one byte at a time, i.e. eight bits per programming pulse. The charge is as permanent as we wish to make it (Well, not absolutly permanent. Most EPROM manufactures will guarantee the charge retention for only(!) ten years). We can change this insulator back into a conductor, however, by shining a short-wavelength ultra-violet lamp through the little window on the EPROM. This causes the insulator to allow the charges to "leak" rapidly, thus erasing the EPROM. This usually takes about 20 minutes. Now, unless you know what you are doing, please read the rest of these instructions before you try programming your first EPROM.

SETTING THE SWITCHES

On the PROmGRAMER is a set of ten switches. All switches should be in the down position (OFF) except for those shown by the white square above the switch. On the right edge of the white squares are some numbers representing the different types of EPROMs. This unit will program the single voltage supply type of EPROM from the 2708 to the 27512. The number 27 does not appear on the guide. For instance, to read or program a 27128, find the number 128 to the right of the guide. You will notice that the squares above switch numbers 1, 4, 7, and 9 are highlighted. These switches should be turned on (UP). NOTE: It is <u>especially</u> important that switches 8, 9, and 10 be in the correct position for the type of EPROM used. These switches control the level of programming voltage, and failure to have them in the correct position may result in catastrophic failure of the EPROM.

DIFFERENT EPROMS

There are several EPROM types other than those listed on the guide. It is important to remember that the PROmGRAMER can only program the single voltage-type EPROMS. Some of the earlier EPROM types need three voltages to operate. These types should be considered obsolete, and the PROmGRAMER will not program them. In particular, TI (Texas Instruments) had an EPROM called a 2716 which uses three voltages. Their 2516 is the equivalent of everyone elses 2716. The Intel type 2716, which everyone else makes, can be programmed by the PROmGRAMER.

Toshiba makes a 27256 which is the same as the others, except that it uses a 21 volt programming voltage, instead of the 12.5 that the others use. Use the settings as shown, except switches 8 and 10 should be DOWN, and switch 9 should be UP.

Due to space limitations, we did not list all EPROM types that you can use. If you have a <u>single</u> voltage 2708, use the settings for the 2716. If you have a <u>2764A</u>, use the settings for the 2764, except switch 8 should be up, and switch 9 down.

PLUGGING IN THE EPROM

When you have the switches in the correct position for the the type of EPROM you will be using, you may plug the EPROM into the ZIF (Zero Insertion Force) socket. This is the large socket to the left of the guide. The lever will securely hold the EPROM when it is parallel to the PROmGRAMER board. Move the lever perpendicular to the board to insert or remove the EPROM. IMPORTANT NOTE: If the EPROM is inserted into the socket incorrectly, you may cause (you guessed it!) catastophic failure of the EPROM. Pin #1 of the EPROM is on the same edge as the small notch,

PROGRAMMING THE 27512

The 27512 can hold 64K of memory, which is the entire memory of the standard APPLE][. Obviously, most of that is something we would not want to program, so we must do it in two steps. Assuming the files are saved as "FIRST HALF" and "SECOND HALF", we would proceed somewhat as follows:

```
[initalize program
BLOAD FIRST HALF, A$1000

F 1000 8FFF 0000 (or just "F", since these are the defaults)

   [after about 4 minutes]

BLOAD SECOND HALF, A$1000

F 1000 8FFF 8000
```

CREDITS

The hardware design of the PROmGRAMER is by BOB BRICE. The Software was written by BOB SANDER-CEDERLOF. DOS 3.3 is a copyrighted program by APPLE COMPUTER. PROmGRAMER is a trademark of SOUTHERN CALIFORNIA RESEARCH GROUP.

A MESSAGE TO quikLoader OWNERS

We're sure that you would like the convenience of having the driver program on the quikLoader, available for <u>instant</u> loading without the bother of having to find the correct disk. Would we let you down? On the program diskette is a file called "QUIKLOADER". It is all set up to load in location $1000, and is complete with overhead files to store in a 2716. To store it in a larger chip, "BLOAD" it at the correct address (i.e. 2732 - A$1800, 2764 - A$2800, etc.)

YOU'RE ON YOUR OWN, NOW

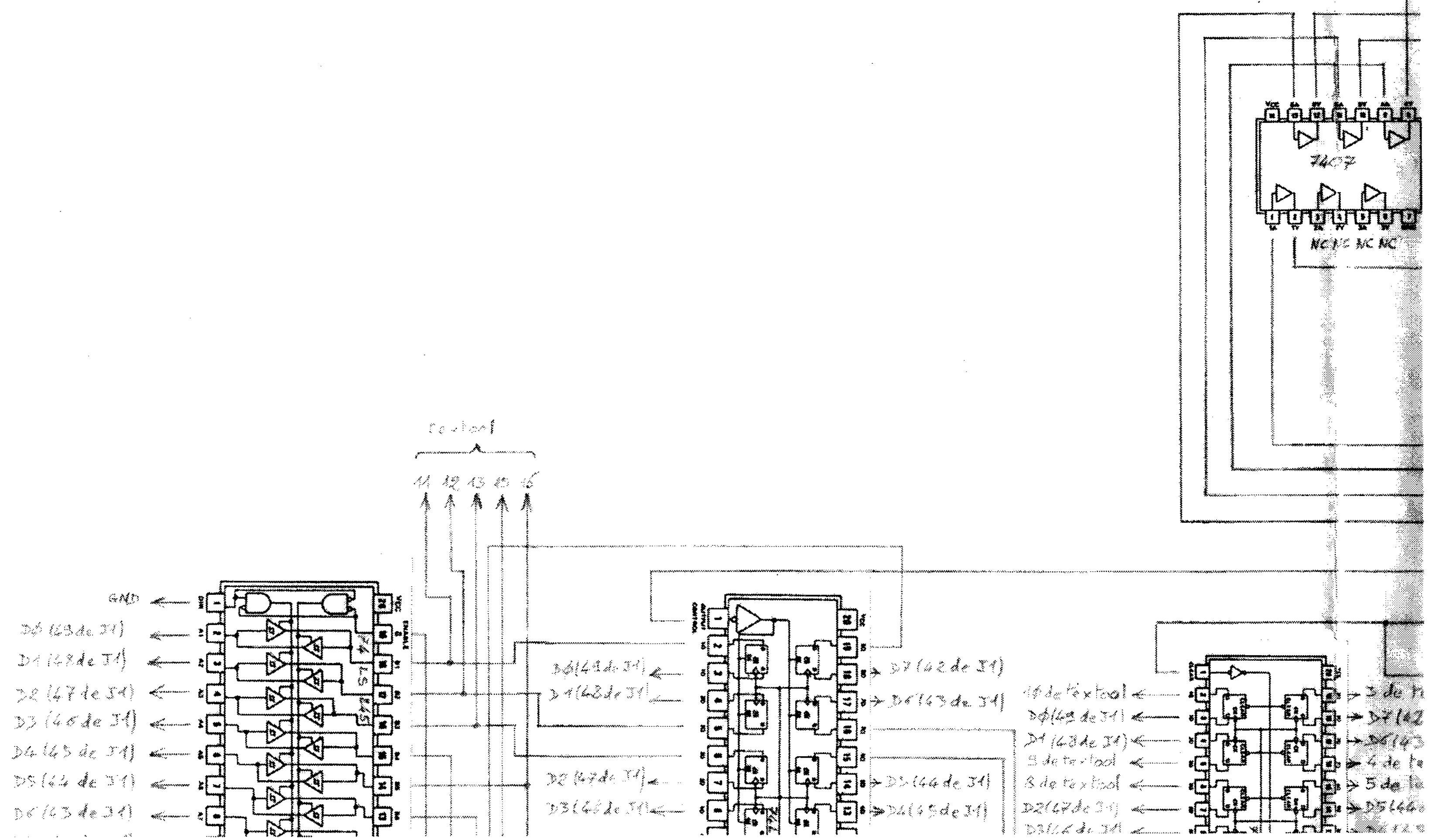
Well, not quite, we are available to give you any necessary assistance. Call us at (805) 529-2082. While we cannot be expected to teach you to program, we will be happy to help you with any PROmGRAMER questions.

WARRANTY

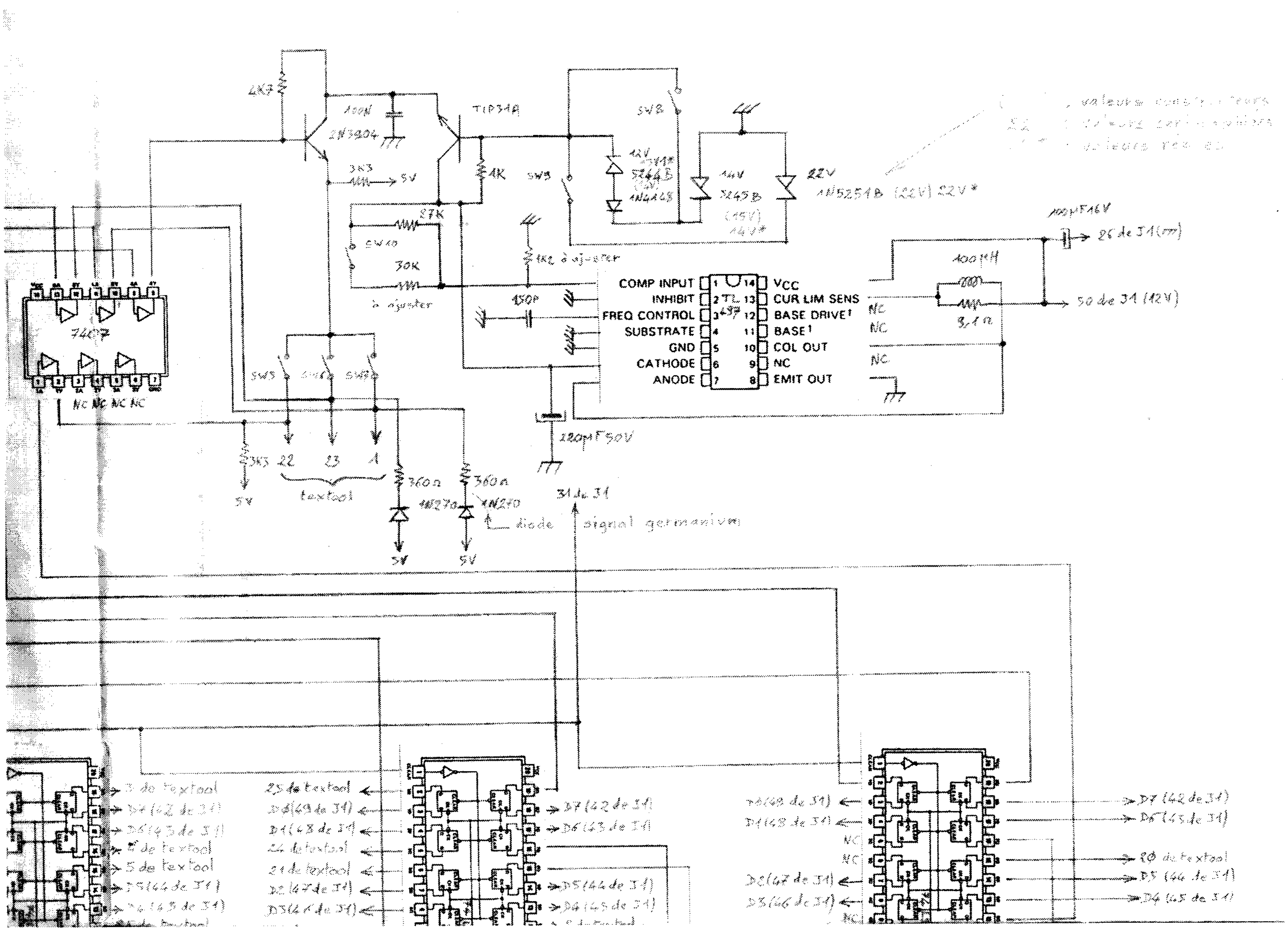
THE PROmGRAMER is warrented for six months for defects in parts or workmanship. If you have any problem within this time, return it postpaid to us, and we will repair or replace it. Our address is:

SOUTHERN CALIFORNIA RESEARCH GROUP
Post Office Box 593
Moorpark, CA 93020

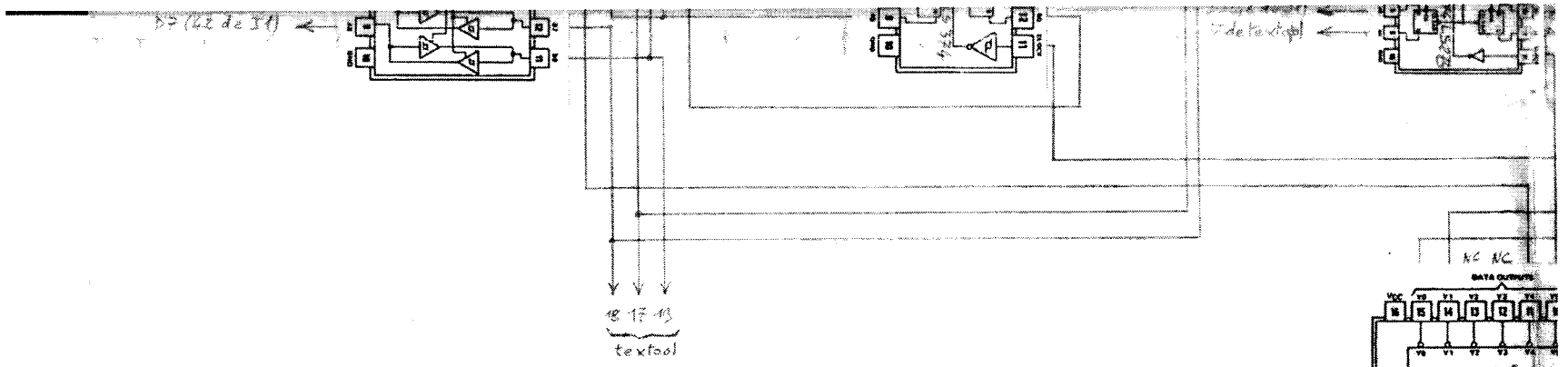
Telephone (805) 529-2082



7407
NC NC NC NC
textool
11 12 13 15 16
GND
D0 (49 de J1)
D1 (48 de J1)
D2 (47 de J1)
D3 (46 de J1)
D4 (45 de J1)
D5 (44 de J1)
D6 (43 de J1)
74 LS 245
D7 (42 de J1)
D6 (43 de J1)
D5 (44 de J1)
D4 (45 de J1)



4K7
100N
2N3904
3K3
5V
TIP31A
1K
27K
SW10
30K
à ajuster
SW8
SW9
12V
1N4148
14V
5245B
(15V)
22V
1N5251B (22V) 22V*
valeurs constructeurs
valeurs réelles
1K2 à ajuster
100µF16V
25 de J1
100µH
50 de J1 (12V)
COMP INPUT
INHIBIT
FREQ CONTROL
SUBSTRATE
GND
CATHODE
ANODE
VCC
CUR LIM SENS
BASE DRIVE
BASE
COL OUT
NC
EMIT OUT
TL497
150P
220µF50V
7407
NC NC NC NC
SW5
SW6
SW7
3K3
22
23
1
textool
360Ω
1N270
diode signal germanium
31 de J1
3 de textool
D7 (42 de J1)
D6 (43 de J1)
4 de textool
5 de textool
D5 (44 de J1)
25 de textool
D0 (49 de J1)
D1 (48 de J1)
24 de textool
21 de textool
D2 (47 de J1)
D3 (46 de J1)
D4 (45 de J1)
20 de textool



28 b. texfool ⟵ 5V (25 de J1)

5 x 0,1 µF de découplage des CI

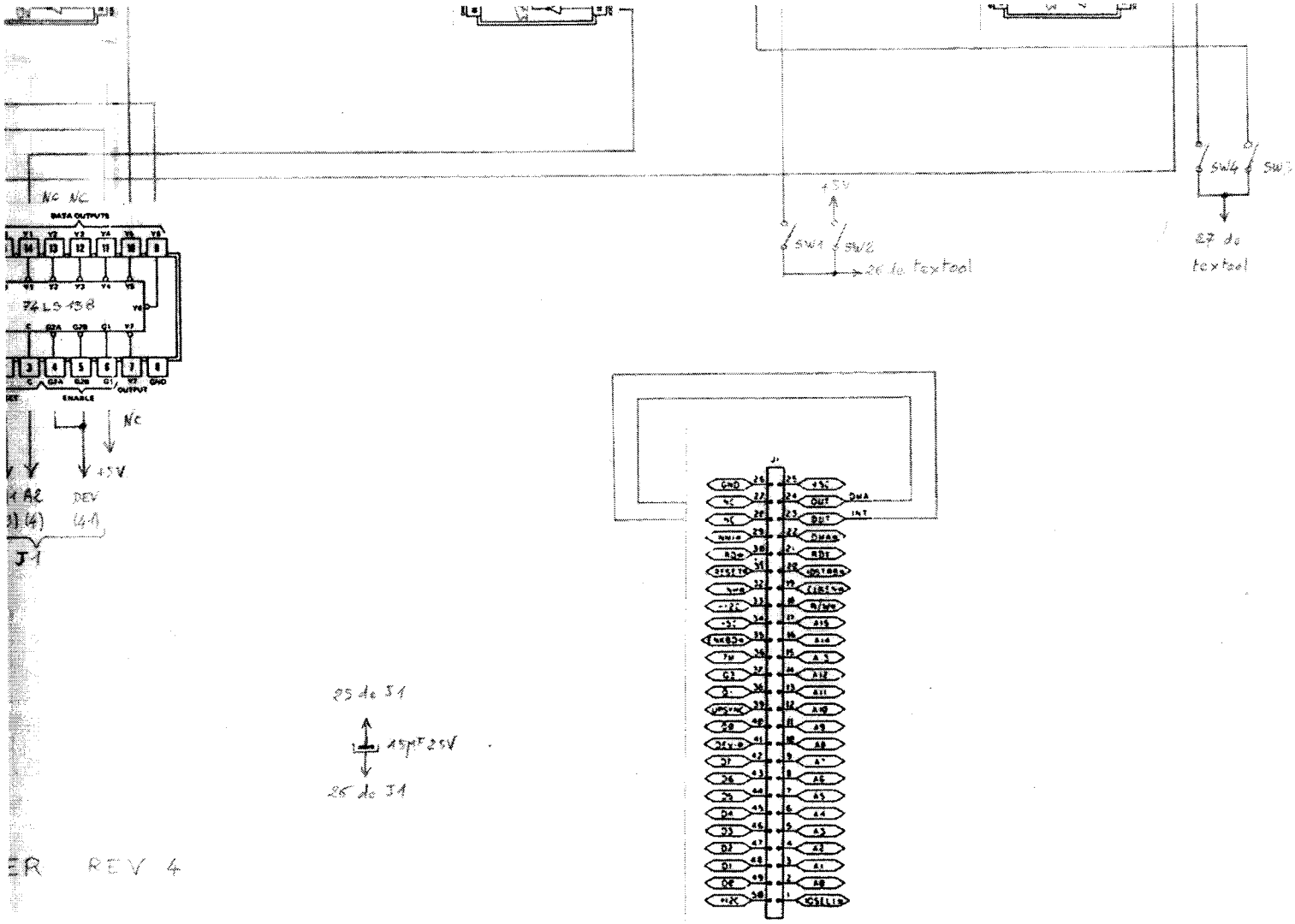
0 - FFFF	27512	1	0	1	0	1	0	0	1	0	1
0 - 7FFF	27256	1	0	1	0	0	0	1	1	0	1
0 - 3FFF	27128A	1	0	0	1	0	0	1	1	0	1
0 - 3FFF	27128	1	0	0	1	0	0	1	0	1	0
0 - 1FFF	2764	1	0	0	1	0	0	1	1	0	0
0 - 1FFF	2764	1	0	0	1	0	0	1	0	1	0
0 - 0FFF	2732A	0	1	0	1	1	0	0	0	1	0
0 - 0FFF	2732	0	1	0	1	1	1	0	0	0	0
0 - 07FF	2716	0	1	0	1	0	1	0	0	0	0
		1	2	3	4	5	6	7	8	9	10

12,5V	± 0,5V
	± 0,3V
	± 0,5V
21V	± 0,5V
12,5V	± 0,5V
21V	± 0,5V
	± 0,5V
25V	± 1V
	± 1V

1: ON
0: OFF

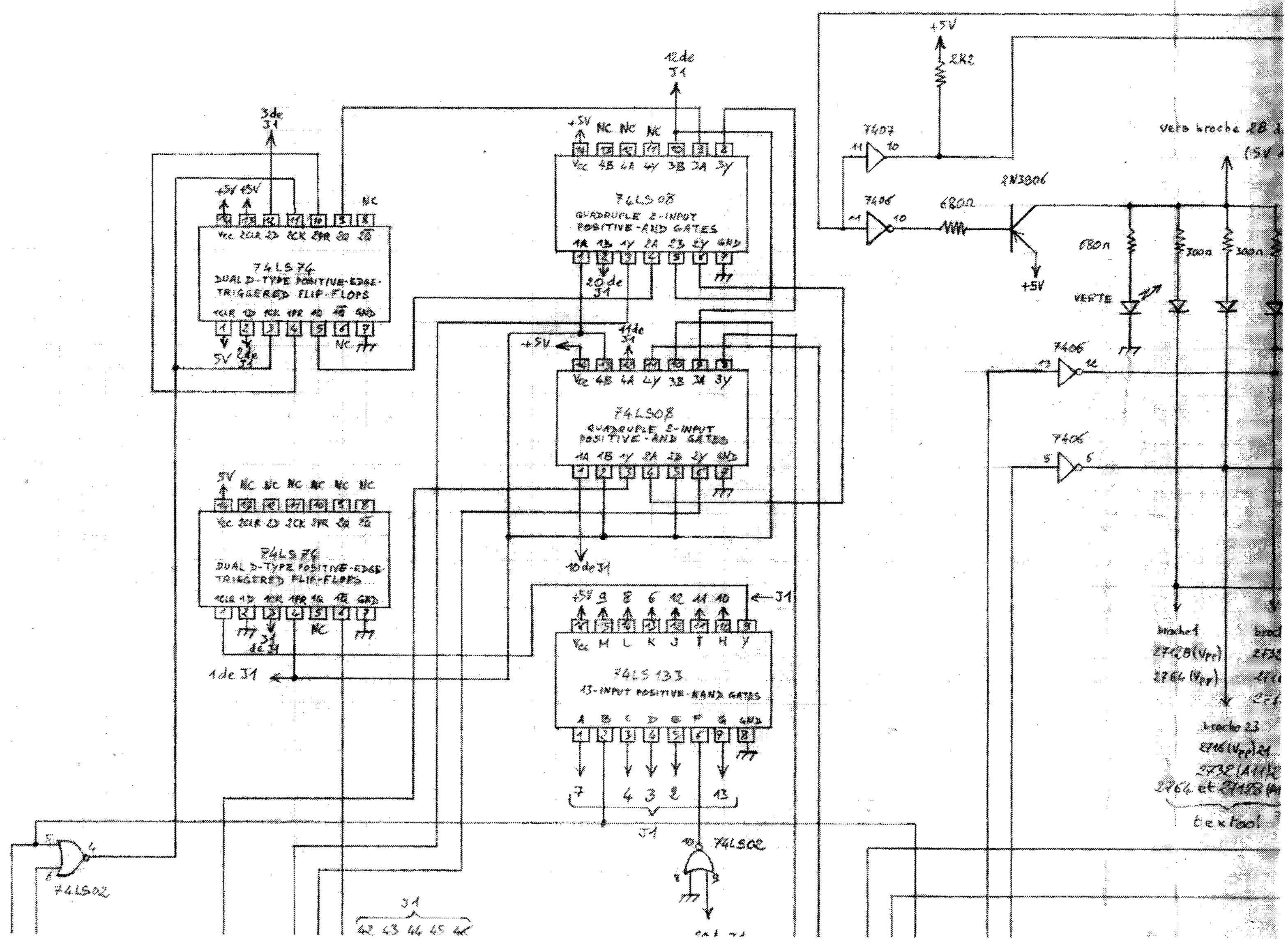
Tableau de programmation du bloc INTER-DIL

PROMGRAMER REV

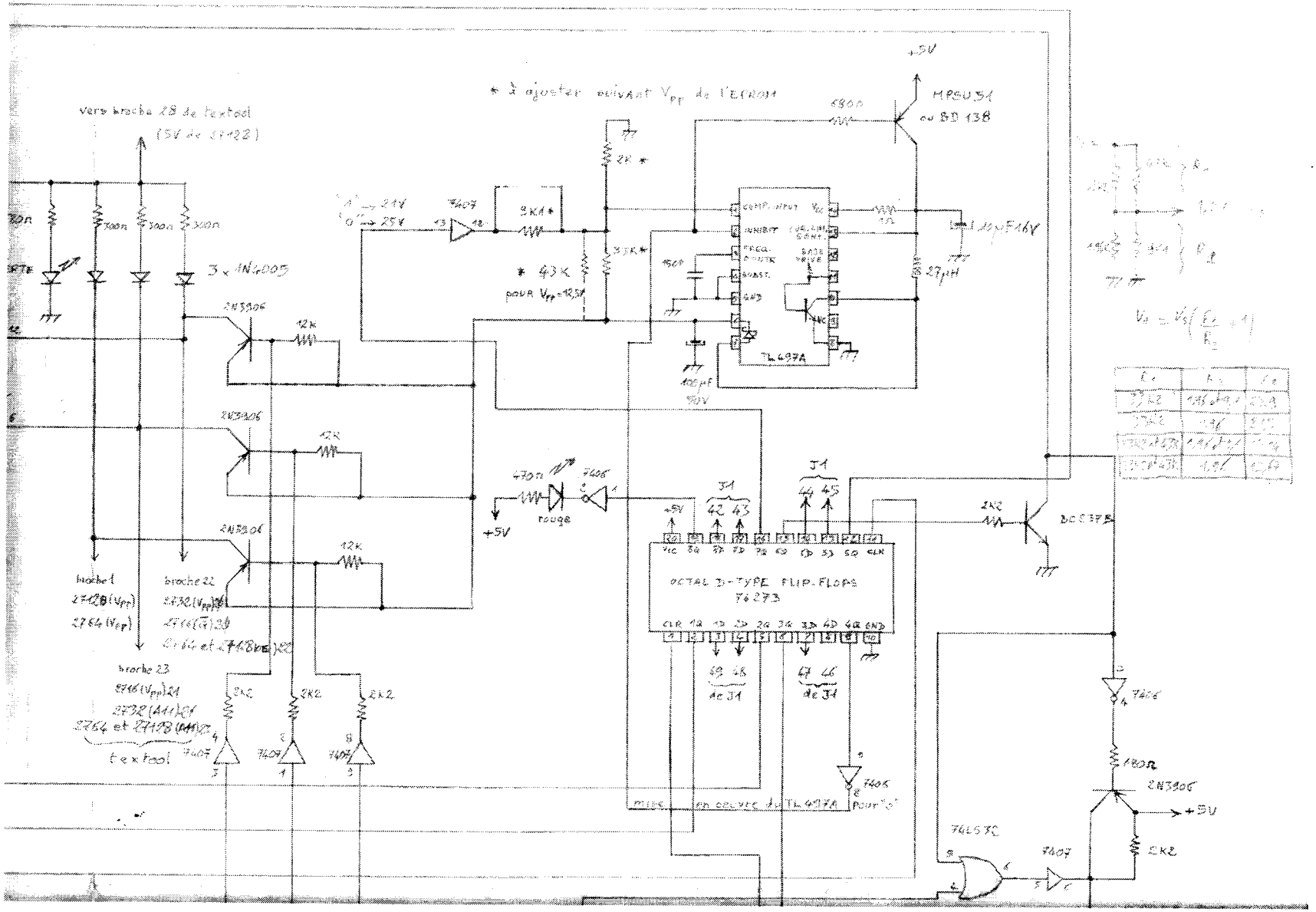

NC NC
74LS138
ENABLE
NC
+5V
SW1
SW2
SW4
SW3
27 do textool
J1
25 do 31
15pF 25V
25 do 31
+5V
DMA
INT

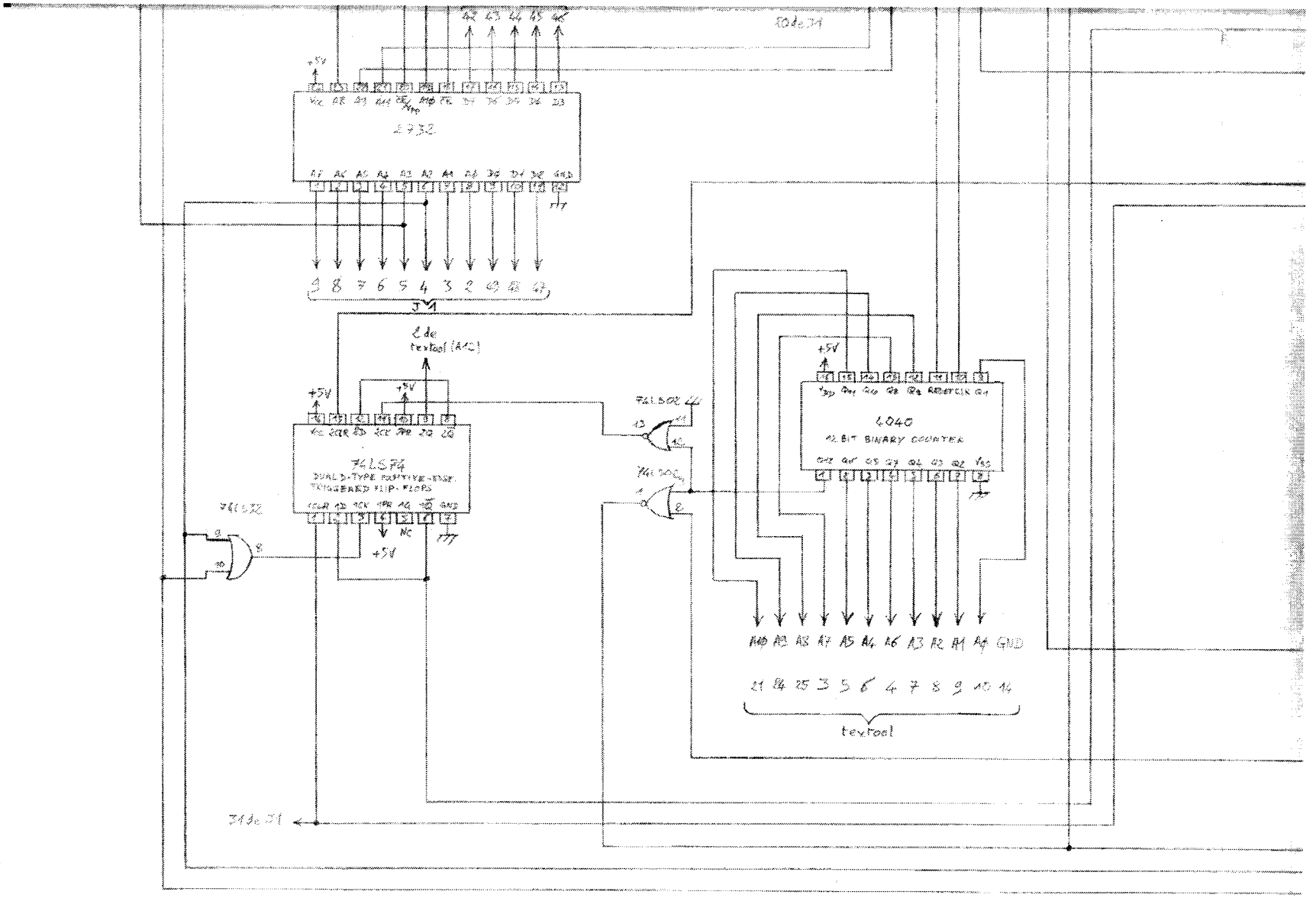
ER REV 4

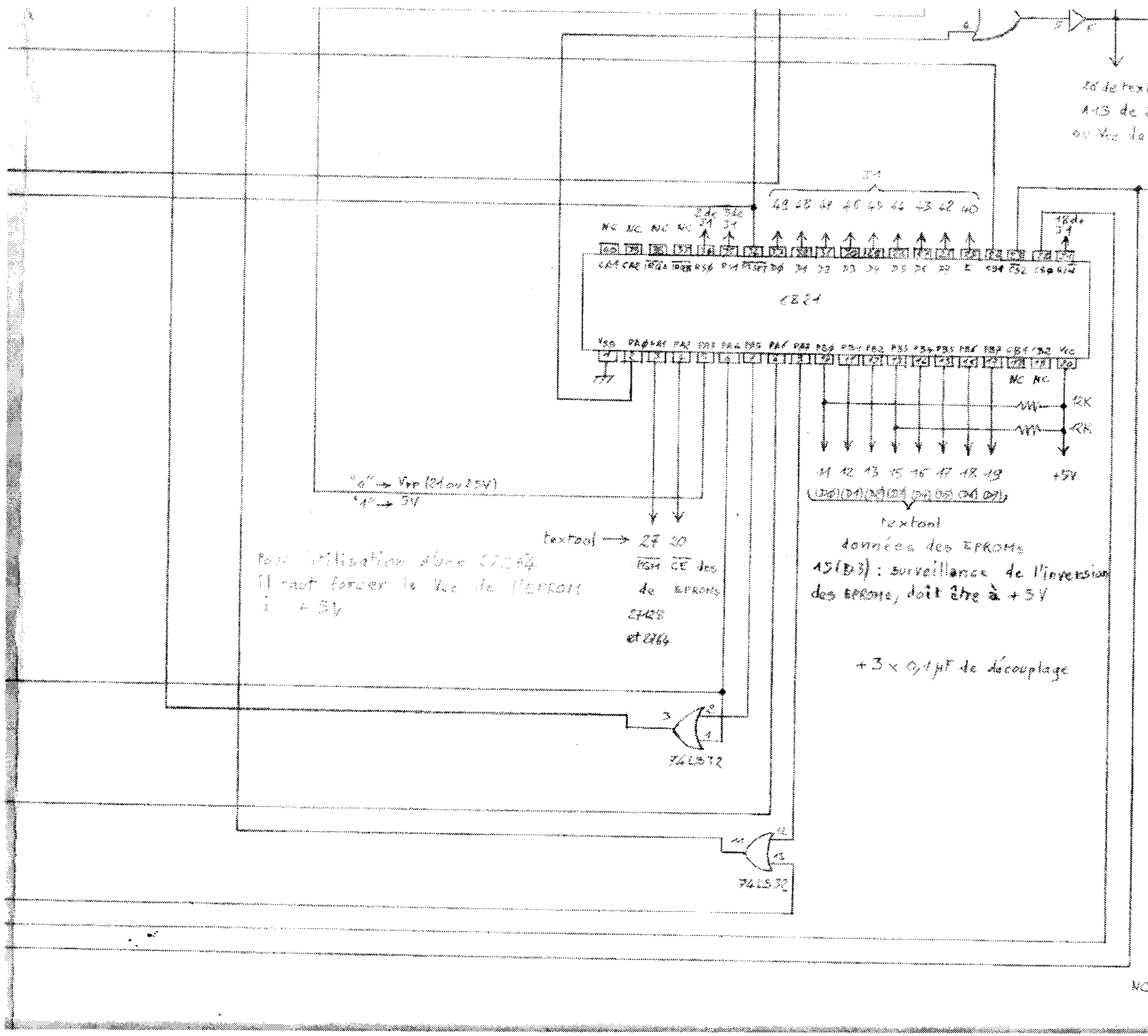
CARTE PROGRAMMATRICE D'EPROMS "LASE



2716 - 2732 - 2732A - 2532 - 2764 - 27128




+5V
2732
4040
12 BIT BINARY COUNTER
74LS74
DUAL D-TYPE POSITIVE-EDGE-TRIGGERED FLIP-FLOPS
74LS32
74LS00
textool (A4)
textool
NC

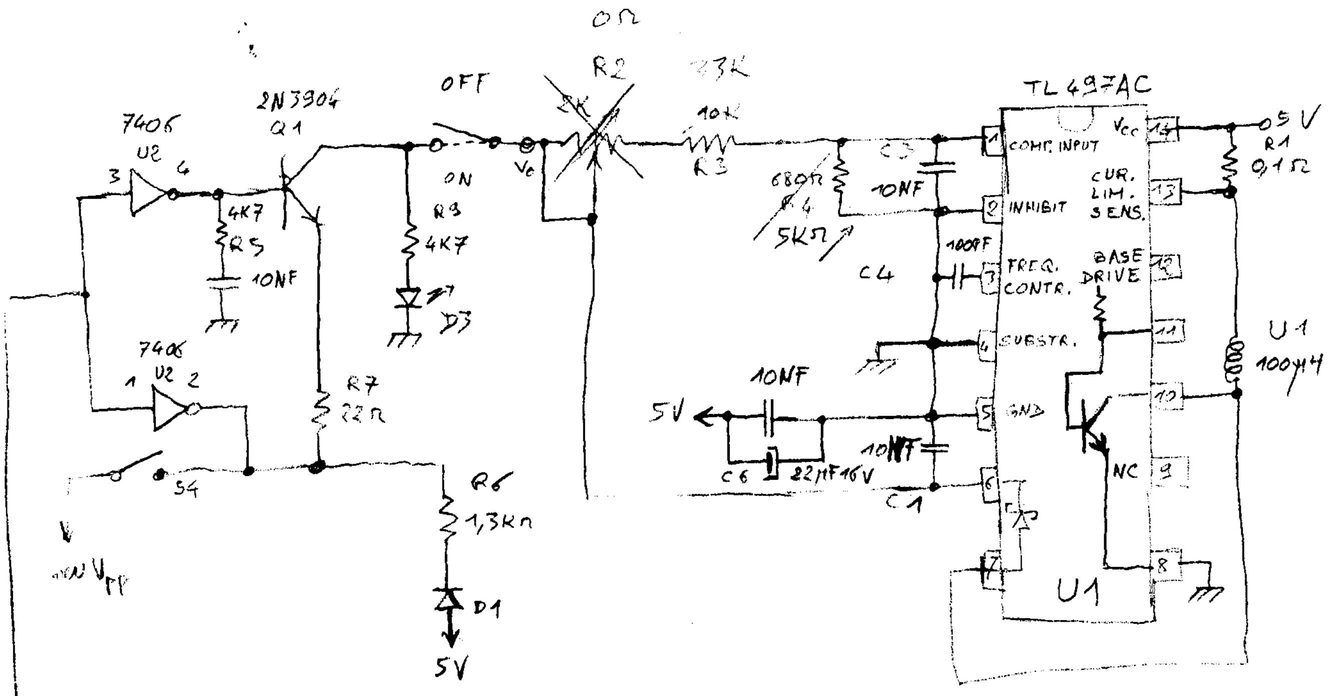


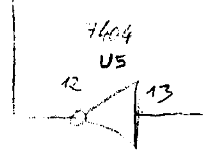
J1

GND	26	25	+5V
NC	27	24	NC
NC	28	23	NC
NC	29	22	NC
NC	30	21	NC
$\overline{RESET}$	31	20	$\overline{IOSTB}$
NC	32	19	NC
NC	33	18	R/$\overline{W}$
NC	34	17	NC
NC	35	16	NC
NC	36	15	NC
NC	37	14	NC
NC	38	13	A11
NC	39	12	A10
Φ0	40	11	A9
$\overline{DEV1}$	41	10	A8
D7	42	9	A7
D6	43	8	A6
D5	44	7	A5
D4	45	6	A4
D3	46	5	A3
D2	47	4	A2
D1	48	3	A1
D0	49	2	A0
NC -12V	50	1	$\overline{IOSEL1}$

	broche Vpp
2716	21
2732	20
2764	27

R : 1/4 W





$$V_o = \left(1 + \frac{R_3+R_2}{R_4}\right) V_{ref}$$

V_{ref} = 1,22 V théor.

1,45 V prat.

avec R_2 = 5K

prat. 22,8V ≤ V_o ≤ [illegible] V

1er [illegible] 20,7V = [illegible] ≤ 26,5 V [illegible] R_2 = 8,2 KΩ

R_2 = 4,7 KΩ

2732 mesurer le Vpp aux bornes du textool :

0V [illegible] 14 et

21V ou 25V en 20 ou 21 à vide ou 27

2716

R_4 → 4,7 KΩ ajust.

R_3 → 33 KΩ

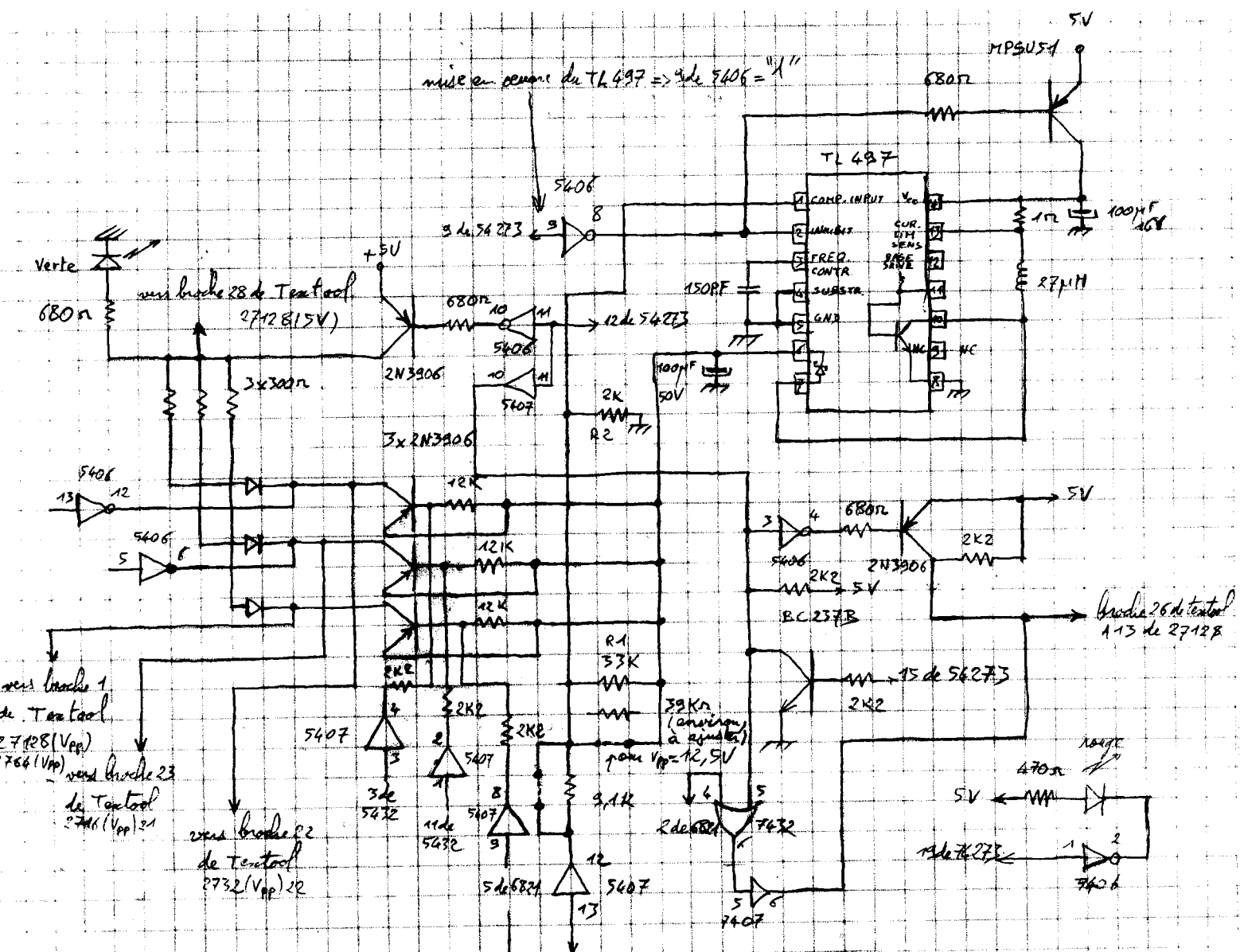
R_2 → porté 0Ω

U_1 → 27 µH

R_1 → 1 Ω

porter C_1 ou C_5 à 100µF

C_4 → 150PF



EPROMS	2716	2732	2732A intel	2764	2764A	27C64 intel	27128	27128A intel
V_{pp} nominal	25V	25V	21V	21V	12,5V	12,5V	21V	12,5V
tolérance	± 1V	± 1V	± 0,5V	± 0,5V	± 0,3V	± 0,5V	± 0,5V	± 0,5V
valeurs max absolues	28V	28V	22V	22V	14V	14V	22V	14V
N° de broche	21	20	20	1	1	1	1	1

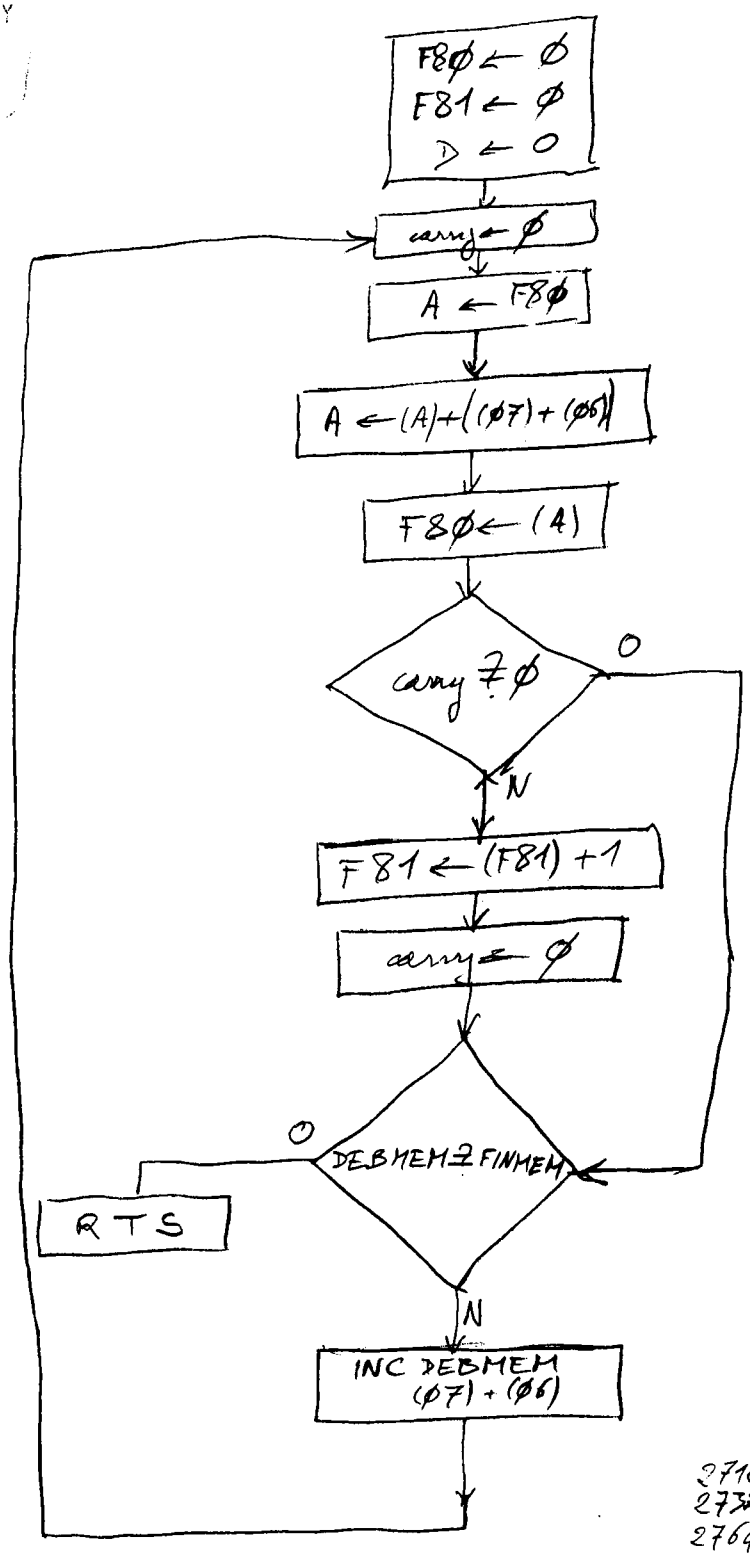
```
0F90- A9 00 8D 80 0F 8D 81 0F
*F90L

0F90-   A9 00       LDA   #$00
0F92-   8D 80 0F    STA   $0F80
0F95-   8D 81 0F    STA   $0F81
0F98-   D8          CLD
0F99-   18          CLC
0F9A-   AD 80 0F    LDA   $0F80
0F9D-   A0 00       LDY   #$00
0F9F-   71 06       ADC   ($06),Y
0FA1-   8D 80 0F    STA   $0F80
0FA4-   90 04       BCC   $0FAA
0FA6-   EE 81 0F    INC   $0F81
0FA9-   18          CLC
0FAA-   A5 06       LDA   $06
0FAC-   C5 08       CMP   $08
0FAE-   D0 07       BNE   $0FB7
0FB0-   A5 07       LDA   $07
0FB2-   C5 09       CMP   $09
0FB4-   D0 01       BNE   $0FB7
0FB6-   60          RTS
0FB7-   E6 06       INC   $06
*L

0FB9-   D0 02       BNE   $0FBD
0FBB-   E6 07       INC   $07
0FBD-   4C 99 0F    JMP   $0F99
0FC0-   00          BRK
0FC1-   00          BRK
0FC2-   FF          ???
0FC3-   FF          ???
0FC4-   00          BRK
0FC5-   00          BRK
0FC6-   FF          ???
0FC7-   FF          ???
0FC8-   00          BRK
0FC9-   00          BRK
0FCA-   FF          ???
0FCB-   FF          ???
0FCC-   00          BRK
0FCD-   00          BRK
0FCE-   FF          ???
0FCF-   FF          ???
0FD0-   00          BRK
*
```

$06 octet bas
$07 octet haut
} DEBMEM

$08 octet bas
$09 octet haut
} FINMEM



2716 : 1ØØØ → 17FF
2732 : 1ØØØ → 1FFF
2764 : 1ØØØ → 2FFF
27128 : 1ØØØ → 4FFF

			1 2 3 4 5 6 7 8 9 10		
	27512	(28)	- 1 0 1 0 1 0 0 1 0 1	12,48V	13,5V ± 0,5
	27256	(27)	- 1 0 1 0 0 0 1 1 0 1	12,58V	12,5V ± 0,5
	+ 27128A	(26)	- 1 0 0 1 0 0 1 1 0 1	12,58V	12,5V ± 0,5
0,6V	27128	(26)	- 1 0 0 1 0 0 1 0 1 0	21,7V	21,0V ± 0,5
0,6V	2764	(28)	- 1 0 0 1 0 0 1 0 1 0	21,3V	21,0V ± 0,5
0,6V	~ 2732A	(24)	0 1 0 1 1 0 0 0 1 0	21,2V	21,0V ± 0,5
0,6V	2732	(24)	0 1 0 1 1 0 0 0 0 0	24,8V	25,0V ± 1
0,7V	+ 2716	(23)	0 1 0 1 0 1 0 0 0 0	25V	25,0V ± 1
			1 2 3 4 5 6 7 8 9 10		
	~ 27c64		1 0 0 1 0 0 1 1 0 0	12,5V	

Offset 1000

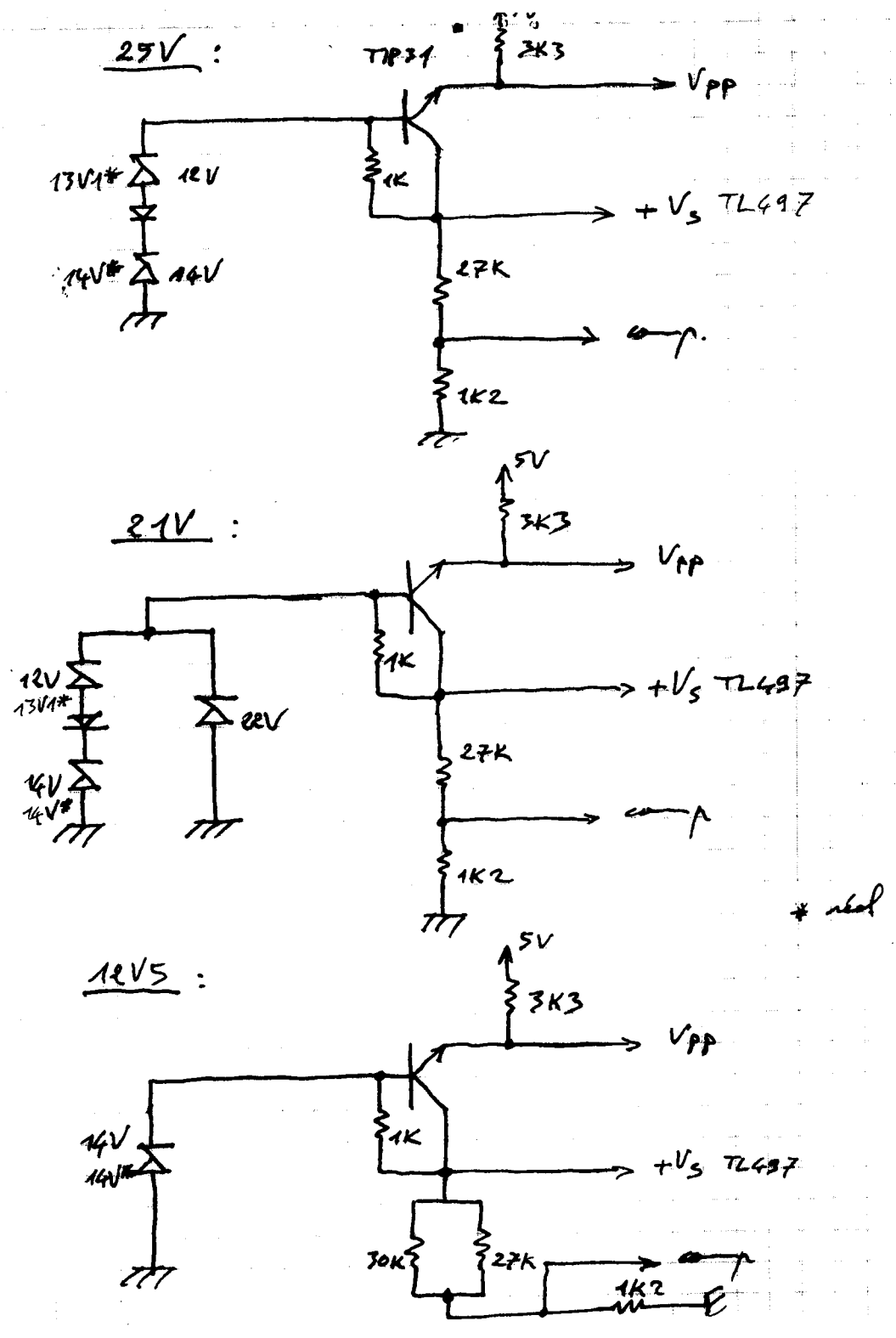
0F	2FFC	E6	1763 00
86	2FFD	49	2FFB "
56	2FFE	E0	
B7	2FFF	00	
20			
03			
86			

01C5 : FF an FB (11C5)

1FFC = E6

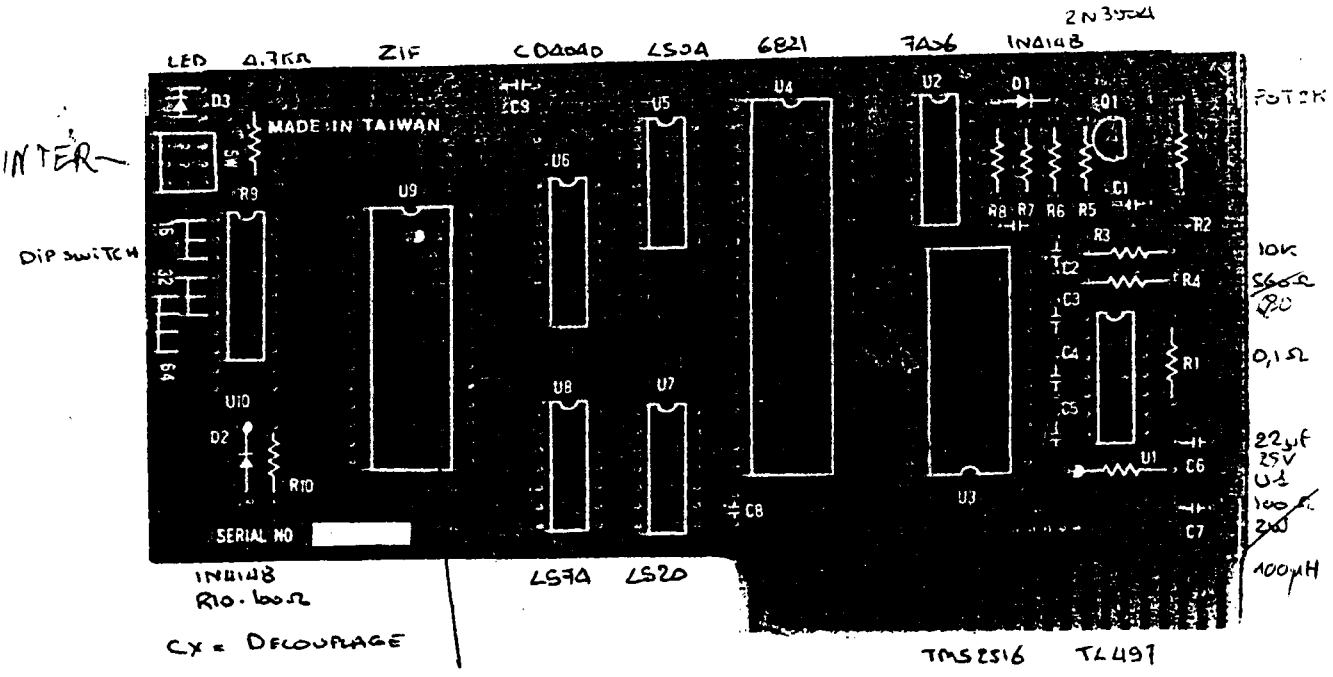
1A1B	F7
1A1C	F7
1A1D	F7
2042	FD

0 - 1FFF	2764	1 0 0 1 0 0 1 0 1 0	} 21V ± 0,5
0 - 0FFF	2732A	0 1 0 1 1 0 0 0 1 0	
0 - 0FFF	2732	0 1 0 1 1 0 0 0 0 0	} 25V ± 1
0 - 07FF	2716	0 1 0 1 0 1 0 0 0 0	


25V :
TIP31
3K3
Vpp
13V1*
12V
1K
+Vs TL497
14V*
14V
27K
1K2
21V :
5V
3K3
Vpp
12V
13V1*
22V
1K
+Vs TL497
14V
14V*
27K
1K2
* réel
12V5 :
5V
3K3
Vpp
14V
14V*
1K
+Vs TL497
30K
27K
1K2

CARTE "PROGRAMMATEUR D'EPROM"

HASMEP



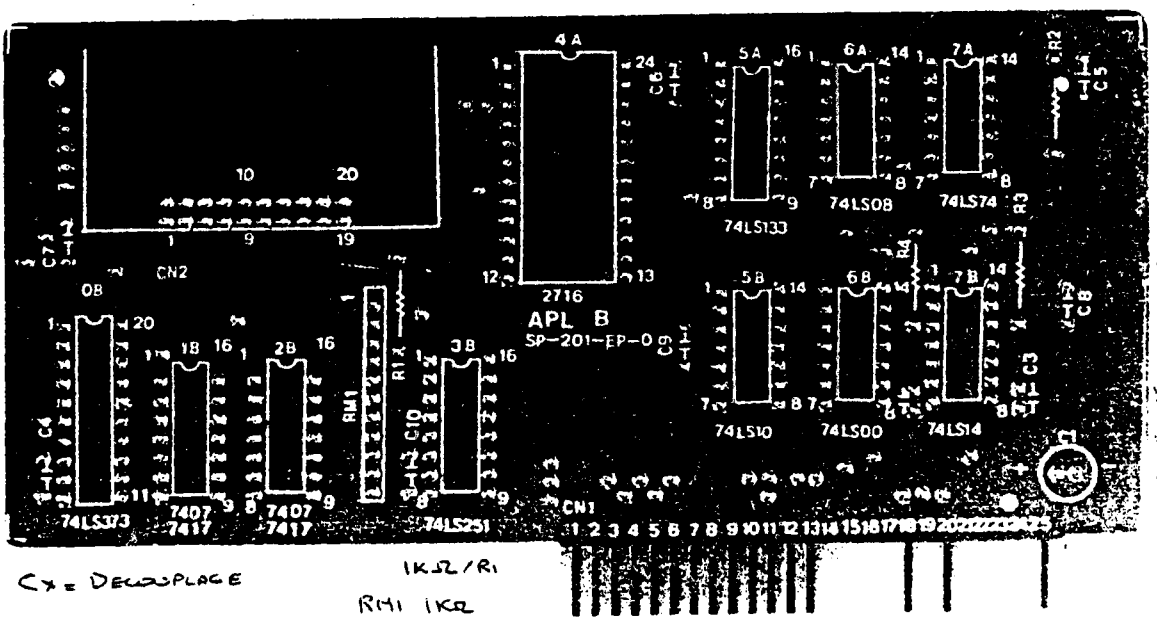
TEXTOOL

$C_1 - C_2 - C_3 \cdot C_5 - C_7 - C_8 - C_9$

10NF

C_4 : 100PF

CARTE INTERFACE IMPRIMANTE PARALLELE

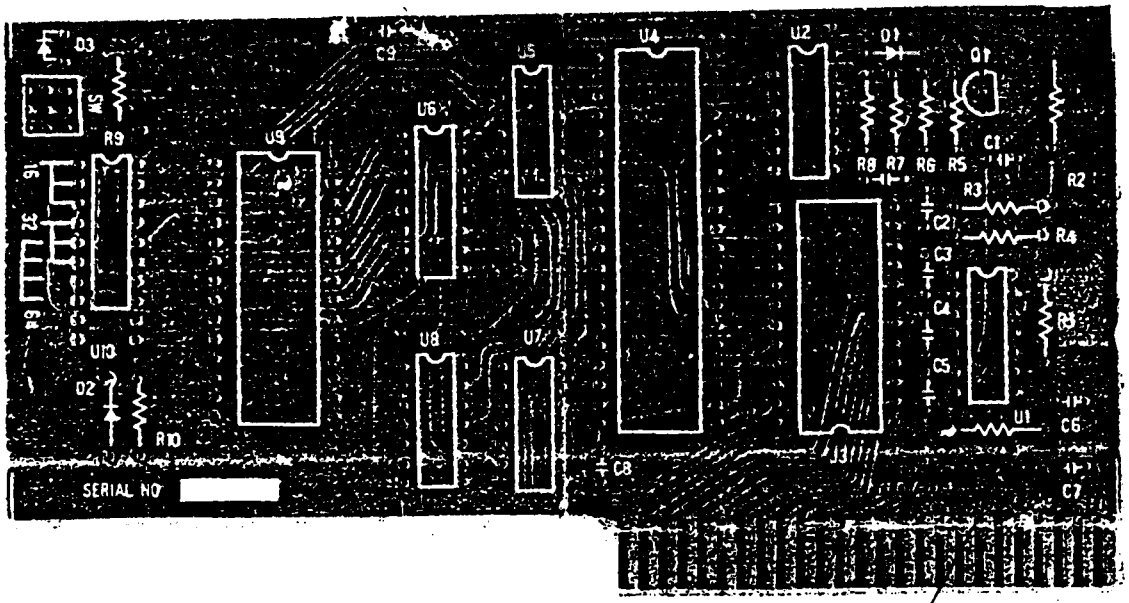


CX = DECOUPLAGE

1KΩ/R1

RM1 1KΩ

HASMG



100 µH

EPROM - BURNER

~~R 1 = 1 Ohm~~
R 2 = 4,7K (trimer)
~~R 3 = 10 k~~
~~R 4 = 680 Ohm~~
~~R 5 = 4,7 k~~
~~R 6 = 120 Ohm~~
~~R 7 = 27 Ohm~~
~~R 8 = 4,7 k~~
~~R 9 = 4,7 k~~
~~R10 = 100 Ohm~~

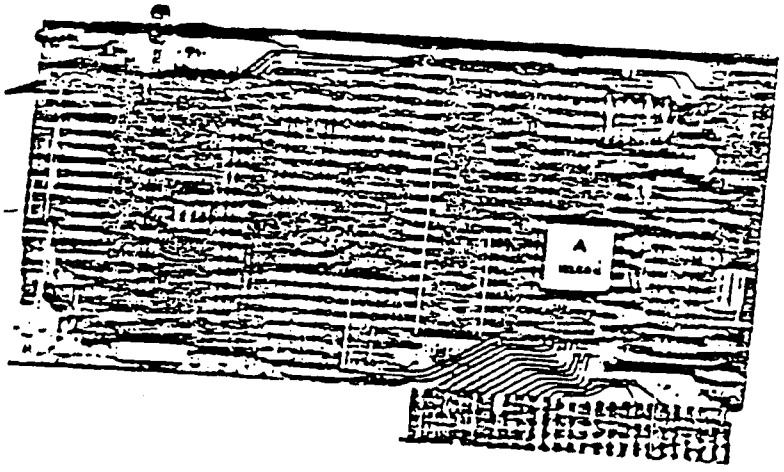
C 2,3,5,7,8,9 = 100 nF + Cx
~~C 1 = 47 nF/35 V~~
~~C 4 = 100 pF~~
~~C 6 = 10 uF/16 V~~

Q 1 = 2N3904
U 1 = TL497AC
~~U 2 = 7406~~
U 3 = Programm Eprom 2716
U 4 = 6821
~~U 5 = 74 ls 04~~
U 6 = CD 4040
~~U 7 = 74 ls 20~~
~~U 8 = 74 ls 74~~
U 9 = 28 pol. Textool

D 1,2 = ~~1N4148~~
D 3 = LED

AP-64e EPROM WRITER

FOR APPLE II, II+, IIe, III.



HOBBY MODEI
THE BEST COMBINATION PROGRAM

Features:

1. Eprom: 2716, 2732, 2764.
2. Reliable.
3. Easy to use (no external equipment needed)
4. Auto blank check.
5. Auto verify.
6. Error indicated.
7. Voltage for program adjustable.
8. Type selection by DIP switch.
9. Fireware included.
10. Write, Read, Copy, Compare, Blankcheck, Monitor functions

Operations:

Introduction: The AP-64e Eprom Writer is a programer for Apple II, II+, IIe, III computer. It offers very fast and easy programing methods through the fireware in board.

Installation: To install the AP-64e Eprom Writer, simply plug it into any slot inside the apple(except slot Ø) as follows: SLOT 5 par ex

1. Turn off the power to the Apple. This is very important to prevent damage to the computer and the AP-64e.
2. The DIP switch on the board will difine what EPROM you are writing, to set DIP switch as following chart:

DIP switch position	1	2	3	4	5	6	7	8
2716	o	o	o	x	x	x	x	x
2732	x	x	x	o	o	o	x	x
2764	x	x	x	x	o	o	o	o

Please confirm DIP switch must be set right as abcve chart, otherwise EPROM could be damaged.

3. Remove the cover from the Apple. Inside along the rear edge of the circuit board, is a series of eight long, narrow sockets called "SLOTS". Insert the AP-64e into any slots except the left one with a gentle rocking motion until fully seated.
4. Now turn your Apple on. Type PR#n (n is slot number). or type IN#n (Integer Basic) . From the Monitor(whose promt character is *), you type Cs0@G or s"CTRL P" (s is slot number).

Functions: There are six functions for AP-64e EPROM WRITER:

1. WRITE: Program any a block of memory content into 2716,2732 or 2764.
2. READ: To move EPROM memory content to RAM location.
3. COPY: Duplicate content of master ROM into copy EPROMS.
4. COMPARE: To verify 2 blocks of memory between systems and EPROMS.
5. BLANK TEST: Test 2716,2732 and 2764 is blank or not.

6. MONITOR: To enter the Apple machine language, y
can examine, change, move, compare and r
the contents of memory in machine langua
of 6502 CPU.

er's guide: After you make installations of AP-64e EPROM WRI
the following will be come to the screen:

AP-64 EPROM PROGRAMMER

ENHANCED VELCOM WRITTEN BY CARL

2) 2716
4) 2732
8) 2764
?

Now you must key in the number 2,4 or 8 to choice how many bytes you want to program your EPROMS. 2 represents 2K bytes (0000-07FF). 4 represents 4K bytes (0000-0FFF). 8 represents 8K bytes (0000-1FFF). Please reconfirm whether the DIP switc already set or not before you key in any number, Errors coul be occured in the following steps if you make wrong selectio When everything is right, you will see the next lines come to the screen:

1) WRITE
2) READ
3) COPY
4) COMPARE
5) BLANKCHECK
6) MONITOR
?

Before you make any action, the first thing is to learn how to insert the EPROM into TEXTOOL:

1. To pull up the stick of textool that can release any ROM or EPROM which already installed in place of textool.
2. To put the right EPROM into the textool. Because the textoo is 28 pins DIP socket that is for 2764 EPROMs, and 2716,273 EPROMs are 24 pins, so the installation of 2716 and 2732 mus 2 pins below caculated from the up side of textool. It is v important to recheck the EPROM goes into the textool in righ pins, otherwise it might be caused a damage.

3. To push the stick of textool down to the PCB side. Now the EPROM will be tightly held in the textool socket.

Now you may key in a number from 1 to 6 to chcice any function we mentioned before. Details are as follows:

1) WRITE: When you type"1"that means you will programing any a block of memory content into EPRCMs. The next line come to the screen is:

START ADDRESS ? 2000 rangement en [illegible] pP

just key in a start address of memory in hex code and hit RETURN.

BLANK ? (Y/N)

A. If you want to "burn" a blank EPRCM, now insert the PEROM and type "Yes", will test EPRCM blank automatically.

RCM CHECK OK! (Means your eprom is ready to write)

SW ON

In case the EPROM has been programmed, Then

RCM CHECK ERR!! (Means your eprom cant write)

BLANK ? (Y/N)

Please change another one and do it again.

B. Sometimes you only want to correct data that already programmed in EPRCM, So please type "No".

The blank check procedure will be ommitted and directlly display:

SW ON

Now you just turn on the switch which is on the front edge of AP-64e, and hit RETURN key. The writing function will start and address hex code should be show on the right down corner of the screen.

For 2716 it will take 95 seconds to fully programming, When the "burnning" is completed, the screen display:

COMPARE OK !!

SW OFF

Please turn the switch off and take the EPROM away from the TEXTOOL, a new fully programming EPRCM has been produced. Or the screen displayed as:

COMPARE ERR !!

SW OFF

that means your EPROM not be written as you need, please turn off the switch and change a new good EPROM and do again until you get a fully programmed EPROM.

2) READ: To instal the EPROM into the TEXTOOL and type 2, the follows will come to the screen:

START ADDRESS ? S

Also please key in a start address of memory in hex code. Where wish to read in from EPROMs. and hit RETURN. After one second, Read function finished:

READ (Show on the screen)

1) WRITE
2) READ
3) COPY
4) COMPARE
5) BLANKCHECK
6) MONITOR

Now you may key in 4 or 6 to check what you READ from EPROMs according the next procedures we will explained later.

3) COPY: To key in "3" means you want to burn a new copy EPROM from a old master ROM. You will see the follows from the screen:

SET MASTER ROM

Please do it carefully and hit RETURN

READ

SET BLANK ROM

Please take the master ROM away and put a blank EPROM in the textool, press RETURN:

BLANK ? (Y/N)

The next procedure is same as WRITE until you get a new copy EPROM.

4) COMPARE: To compare a block of memory and the master EPROM. Only to hit "4" for compare function, the following come to the screen:

START ADDRESS ? S

Please key in start address of memory location in where you wint to compare with the master EPROM

which is inserted in the TEXTOOL. and hit RETURN
SET MASTER ROM
please do it, or you already have done just hit RETURN,
COMPARE OK!
or
COMPARE ERR!!
CONTINUE ? (Y/N)
Just key in "Yes" for continue or "No" to stop it.
If you want to compare two or more EPROMs, the first step is to read data from eprom to the memory location of your system as READ function and then set another eprom as master rom AND do COMPARE function.

5) BLANKCHECK: To test EPROMS is blank or not. please key in "5" and then you will see the follows on the screen:
SET BLANK ROM
Please insert the EPROM you want to test and hit RETURN:
ROM CHECK OK! (If the EPROM is blank.)
or
ROM CHECK ERR!! (When the EPROM is not blank.)

6) MONITOR: To enter the Apple machine language when you type "6". You can examine, change, move, compare and run thecontents of memory in machine language. Please refer the Apple reference manual for more details.

Problems and Solutions: After using AP-64e EPROM WRITER, you will find out so many advantages such as faster, portable and more reliable. But some problems will come to bother you too. Hereafter we like to point out problems may be occurred and show you how to settle.

pour passer du basic au moniteur faire : CALL 151 ↵

1. The EPROMs can not be programmed: Since the EPROMs which could be already stored program, you can't write any more, except you erase and program again.
2. If you forget to key in ROM SIZE (represented by 2,4,8) the errors could be occurred when you use WRITE, READ, COPY functions, please reset your computer and boot AP-64e again.

pour revenir au DOS 3.3 faire CTRL RESET

3. Don't forget to turn the switch off when you use AP-64e from cold start or when screen indicate you "SW OFF".

2716 : 6000.6800 codes hexa (ou 6000.6800)
ou 6000 L désassemblage 20 premières lignes environ (ou 6502)
en BASIC : après avoir lu l'EPROM et rajouté éventuellement à partir de 6000 le protocole sauvegarde sur disquette ; charger DOS 3.3 puis mettre la disquette de sauvegarde
BSAVE NOM DE PROGRAMME, A$6000, L$800
1 pour 2716 1000 (2732), 2000 (2764), 4000 (27128), 8000 (27256)
impression des données : - lire l'EPROM
- taper PR#0
- taper PR#1
- passer en moniteur (CALL 151)
puis 6000.67FF ↵ (pour 2716 par ex) ou 2000L pour désassemblage des 24 premières lignes

Chargement d'un prog. EPROM sur disquette en mémoire centrale d'adresse $2000 par ex :

BLOAD INTERF. IMPR. FFB2, A$2000

SUMMARY OF COMMANDS

COMMAND	FORMAT	DESCRIPTION
C	C [chip type]	Sets up file to use different size chips. Default size is 27256. Use 2764 through 27512.
D	D	Move DOS 3.3 to RAM.
F	F [file name]	Name of file on disk that the AID looks for.
G	G $[addrs]	GO TO MOTHERBOARD. Turns off quikLoader.
H	H $[addrs]	HIGH RAM CONTROL. Causes STA at $C0[addrs]
I	I	Move Integer BASIC to high RAM. Not necessary to run normal integer programs.
J	J [name]	Jump to file named [name]. This can be a K, P, or L file. Does not turn off quikLoader.
K	K [name]	The file name as you want it to appear on the quikLoader Katalog.
L	L [name]	Sets a Label for the J command.
P	P [name]	Tells the quikLoader that this is the power-up application. NOTE - If [name] is left blank, the following file will run only at power-up, as it will not show on the Katalog.
R	R	Turns off quikLoader and performs motherboard Reset
S	S $[addrs]	Turns off quikLoader, does a Subroutine on the motherboard at [addrs], and goes back to the quikLoader. $45 - $49 = Acc, X, Y, status, and stack pointer.
V	V [name]	Verbatim. Takes the file [name] and puts it verbatim (that is, exactly as is) into the quikLoader primary routine. The file must be 6502 machine code. The accumulator value must be retained.

PROGRAMMER'S AID
simplified instructions

quikLoader PROGRAMMERS AID version 4.4

The PROGRAMMERS AID is an APPLESOFT program which will build quikLoader compatible EPROMS from disk-based programs. The instructions are in two parts. This first part consists of the most important, relativly easy directions, followed by a summary of all instructions. The second part covers these commands in more detail, plus some of the more advanced instructions, that might require some programming experience on the part of the user.

A copy of these instructions are on the enclosed disk in a text file named "INSTRUCTIONS"

INSTRUCTIONS

1) Important - To use some of the features of the AID, you must have the newest operating system installed in CHIP 0 of the quikLoader. The new version is labeled C0.256-5. If you do not have this version, the enclosed disk has a ready-to-burn file. Load your EPROM programmer with a 27256, and BLOAD the file (cataloged as C0.256-5). The file is set up to load at $1000. When programmed, turn off power to the computer, remove the quikLoader, and remove the old operating system. (If you had a two or more chip version of the operating system, it will also be necessary to make the jumper and cut the "bowtie" on the back of the quikLoader, by socket 0, as shown on the quikLoader itself.) Plug the new CHIP 0 in socket 0, making sure that the notch on the EPROM is toward the top of the card. Re-install the quikLoader, plug it into the computer, turn on power, and check for proper operation.

If you do not have the facilities for programming a 27256, you may obtain the new chip from us for $10.00, plus $2.50 for shipping.

Note that you do not need the new operating system for using the AID, but for using the chip made with the AID.

2) To use the programmers AID, you need a text file with the commands recorded in it. Most word processors (such as APPLEWRITER) will make the necessary text file. If you do not have access to a word processor, a very rudimentary text file generator is included on the disk. RUN "MAKE TEXT".

3) The format of the commands are COMMAND (at least one space) NAME. For example, the command KATALOG MYFILE will tell the AID to construct a quikLoader program that will show on the quikLoader katalog as "MYFILE". It is not necessary to use more than one letter of the command e.g. K MYFILE will work just as well.

4) The command FILE (NAME) tells the AID to look for a file on the disk to turn into a chip program. Putting the above two commands together, we now have the simplest possible program for building a chip:

```
K MYFILE
F TESTFILE
```

This program will look on the disk for a file named "TESTFILE", and will turn it into a text file which, when EXECed, will build a file containing the overhead, catalog, and program, on a ready-to-burn file. Note that it does not matter if the program "TESTFILE" is APPLESOFT, Integer, or machine language.

5) If you wish the computer to auto-start a program on chip, the chip must be plugged into socket 6 of the quikLoader. You also have to inform the quikLoader that this is the particular program that will be the "POWER-UP" program. Thus, we use the command POWER-UP, in place of K, to show the QLOS that this is the power-up application. IMPORTANT NOTE: If your power-up application needs to use DOS, you must first tell the quikLoader to load DOS. This command is "DOS" or "D".

6) You can put several, or even many, programs on each chip, subject to memory availability. One of these programs can be the N-RESET program, that is, the program will run when the number corresponding to the quikLoader socket number is pressed, along with control and reset. The first program on the list will be the "N-RESET" program.

7) Sometimes you will find it necessary to load one or more machine language programs before running an APPLESOFT program. To do this, list the machine language program(s) before the APPLESOFT program. For example, if you have an APPLESOFT program on disk named INVOICE that needs a machine language segment named CLOCK DRIVER, and you want the katalog name to show as "DATED INVOICE", the program will look like this:

```
K DATED INVOICE
F CLOCK DRIVER
F INVOICE
```

Again, if the "DATED INVOICE" program were to be the "POWER-UP" program, the commands would be as above except:

```
P DATED INVOICE
```

Naturally, if your APPLESOFT program BLOADS the machine language segment, the line that contains the BLOAD command should be removed from the program.

8) The AID assumes that you are using a 27256 EPROM. (anything smaller is wasteful of the quikLoader.) If you must use a different size, the command is "CHIP", e.g.:

```
C 27128
```

You are limited to the chips from the 2764 to the 27512.

9) If your program is too large to fit on a single chip, not to worry, the AID will take care of that. You may use several CHIP commands. If the programs take more space than you planned for, the AID will tell you that there is not enough room, giving you a chance to either call for an additional chip, or substitute a larger one.

10) Getting back to the beginning, DO NOT do any programming on the supplied disk. We suggest that you use one disk per chip set, at least at the begining. Onto a freshly INITted DOS 3.3 disk, transfer the APPLESOFT program AID, and the binary program CAT (A$8400, L$200). Using a text editor, build your commands, and save them to a text file with a file name of your choice on the same disk. (this name will be shown in the following paragraphs as (name)). Transfer all disk files that you will be using onto the same disk.

11) When you're ready, RUN AID. When the screen shows the words "INSERT SOURCE DISK", hit return. The screen will then show the words "ENTER SOURCE FILE NAME". Enter the (name) and hit return. The program will go through your text file, command by command, and find the necessary files. It will also make some additional files on the disk. After three passes, it will make a file called "MAKE (name)". When the words "SUCCESFUL ASSEMBLY" appear on the screen, hit reset to leave the program, then type "EXEC MAKE (name)". The computer will go through that text file and make a file called "(name).C0" (that's number zero, not letter "oh".)(If you end up with more than one chip file, they will be (name).C1, etc.) This file is a standard DOS 3.3 machine language file, starting at $1000, with the length necessary to fill the selected chip. Simply burn this into a chip, plug it into the quikLoader, and you should be all set.

12) A few additional notes:

When burning one or more chips in a set, the lowest chip number contains the GETSLOT.

Unused bytes are set to $FF, to cut down on programming time.

All files built using the AID are primary routines. This means that you do not waste the space between $C000 and $C0FF.

13) REVIEW BY EXAMPLE:

```
C 27256
C 2764
K QLFILE1
F PROGRAM1
K QLFILE2
F PROGRAM2
P QLFILE3
F MLFILEA
F MLFILEB
F MLFILEC
F PROGRAM3
```

Explanation: We expect to use two chips, a 27256 and a 2764. The first program that will appear on the quikLoader Katalog for this chip will be "QLFILE1". This program appears on the disk under the name PROGRAM1. This program will also be the N-RESET program of this chip.

The second program will appear on the quikLoader Katalog as "QLFILE2". It corresponds to the disk-based program "PROGRAM2".

The third (and last) program will be shown as "QLFILE3". The quikLoader will first load three machine language segments, "MLFILEA", "MLFILEB", and "MLFILEC". The quikLoader will then load the program "PROGRAM3". "QLFILE3" is also the POWER-UP program on this chip.