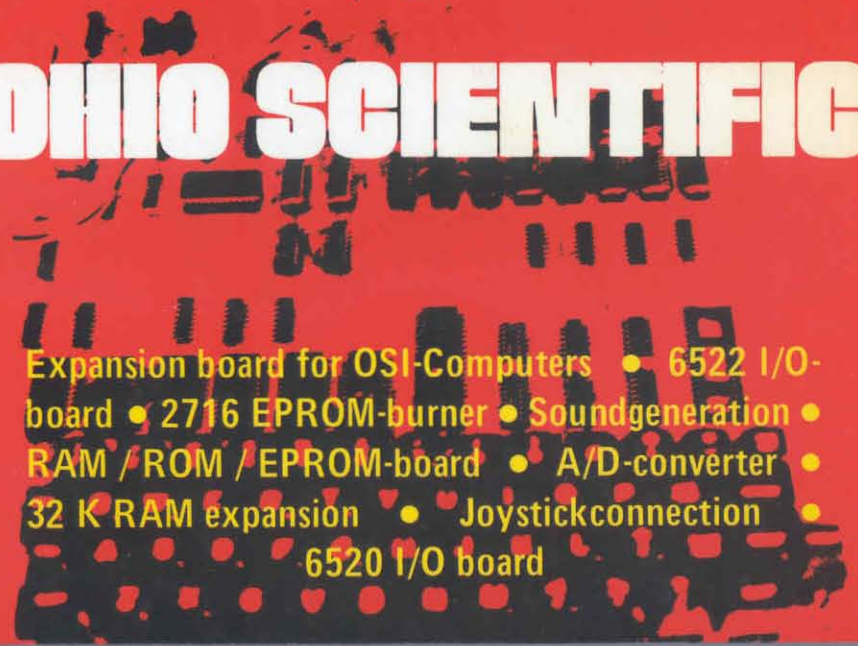
S. Roberts

THE THIRD BOOK of OHIO SCIENTIFIC



Expansion board for OSI-Computers • 6522 I/O-board • 2716 EPROM-burner • Soundgeneration • RAM / ROM / EPROM-board • A/D-converter • 32 K RAM expansion • Joystickconnection • 6520 I/O board

ACKNOWLEDGEMENT

1. OHIO Scientific: The C4P/C1P Users Manual
   OHIO Scientific, 1333 South Chillicother Road, Aurora, OH 44202
2. BETA Computer Devices, P.O.Box 3465, Orange, CA 92665
3. ELCOMP Microcomputer Magazine, Post Box 437
   D-8000 Munich 75, W.-Germany
4. AY-3-8910/8912 Programmable Sound Generator Data Manual, General Instrument, Microelectronics Division, 600 West John Street, Hicksville, NY 11802
5. R6500 Programming Manual Rockwell International, P.O.Box 3669, Anaheim, CA 92803
6. Reference is made to APPLE II throughout this book.
   APPLE II is a trademark of APPLE Computers in Cupertino, California
7. Fairchild TTL-Data Book from Fairchild, 464 Ellis Street, Mountain View, CA 94042
8. Fairchild DATA Sheet µA 9708

We also want to thank Ekkehard Floegel
Franz Ende
John Kozero
David Wilkie
and Winfried Hofacker
for their help in completing this book.

This book is published as a service to all OHIO Scientific users. No liability is assumed with respect of the use of information herein. Reproduction or publication of the content in any manner, without express permission of the publisher, is prohibited.

ISBN 3-921682-77-0

Published by: Ing. W. Hofacker GmbH
Postbox 75 04 37
D-8000 Munich / West-Germany

US-Distributor: ELCOMP Publishing, Inc.
Postbox 1194
Pomona, CA 91769

Printed in West-Germany – Imprime' en RFA

THE THIRD BOOK of OHIO SCIENTIFIC

Introduction

I. Introduction

What OHIO Scientific user has never considered tailoring his machine to his own needs.

In this book we will show, how many ways there are to expand and modify the hardware of OHIO SCIENTIFIC computers, using circuits you probably never thought about.

We describe how to hook up a motherboard that expands your system with four slots using connections like the APPLE-Bus or any similar configuration and one slot like the S44-Bus. Into these slots you can plug various boards with your specific circuits on it.

We describe how to burn EPROMs with your OHIO SCIENTIFIC computer, how to build up a soundboard, how to use an analog-digital converter, how to hook up a parallel printer and much more.

Table of Content

ELCOMP–1 EXPANSION MOTHERBOARD DESCRIPTION

The ELCOMP-1 is a motherboard for expansion of single-board computers. It provides an efficient expansion area consisting of four APPLE-compatible (see note 1) slots, and one S44-E slot. The ELCOMP-1 interfaces easily with most single-board microcomputers, including the OSI Superboard II, ATARI, Commodore VIC, Commodore PET, and AIM/KIM Systems.

A 40-conductor ribbon cable, with appropriate connector, couples the ELCOMP-1 with the host microcomputer. ELCOMP –1 requires the complete address and data bus, phase-2 clock, read/write-not, and master reset signals from the host system. Address, data, and clock signals are routed from the host system interface connector (J6) directly to Apple connectors (J1 – J4), and to S44-E connector (J5). U46 provides phase–2–not to the S44-E Bus as required.

Most of the circuitry on the ELCOMP -1 motherboard is devoted to address decoding for the APPLE slots, and to control of the

OSI Superboard-II data direction line (DD–NOT).U6–15 goes true (low) for memory addresses from C000 to C7FF. This signal, combined with true (HI) phase-2 clock and true (HI) memory-read signal, cause U1–13 to pull down on the OSI Superboard–II data-direction line. This reverses the direction of the Superboard's data-bus tranceivers, and permits data to flow from external memory (on ELCOMP-1) onto the Superboard's internal data bus. If the Superboard is to read memory outside the C000–C7FF range, then extra decoding gates must be added to pull down on data-direction over the appropriate address range. A prototyping area is provided on the ELCOMP-1 for this reason.

The signal at U6–15 also enables decoders U3 and U2. The eight outputs of U3 indicate which 256 byte page between C000 and C7FF is being addressed. Four of these outputs are unused, the others provide I/O select signals to APPLE connectors J1 through J4. These signals are low when true.

APPLE SLOT #	I/O SELECT ADDRESSES
J1	C1XX (C100–C1FF)
J2	C2XX (C200–C2FF)
J3	C3XX (C300–C3FF)
J4	C4XX (C400–C4FF)

TABLE 1 I/O SELECTS

U2 decodes memory address from C000 to C1FF into eight 16-byte segments, indicated by the eight active-low outputs of U2. The signals act as APPLE-bus device selects, and are coupled to the four APPLE connectors summarized below.

APPLE SLOT #	DEVICE SELECT ADDRESS
J1	C01X
J2	C02X
J3	C03X
J4	C04X

TABLE 2 DEVICE SELECTS

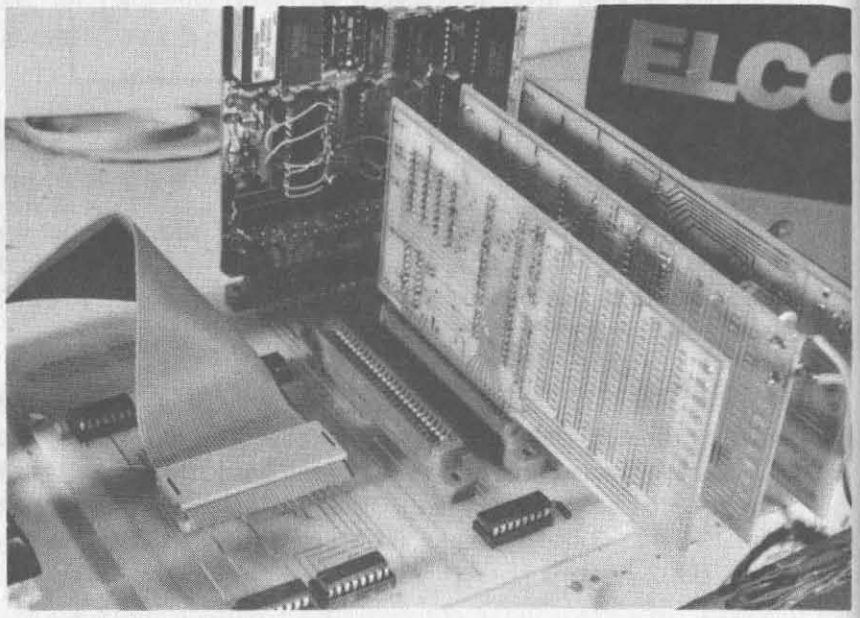
An I/O strobe signal is generated for memory addresses between C800 and CFFF. This signal is available at U6–14. Note that the Superboard data bus buffers are not reversed for memory reads in this address range.

The ELCOMP-1 is compatible with a variety of APPLE-boards, including the EPROM programmer, parallel I/O, ROM card, RAM card, sound-effects card, and prototyping card offered by ELCOMP (also in bare-board form). ELCOMP-1 is ideal for use with the popular S-44 32K dynamic RAM card offered for use with 1-Mhz 6502 systems.

The ELCOMP-1 connects immediately to the Superboard-II with only a 40-pin ribbon cable. The ELCOMP-1 is designed to use separate power supplies, but may be modified to draw power from the Superboard. The only modification necessary to the Superboard consists of adding a jumper from the reset-not pin of the 6502 to the bus-output-connector, pin 31. This will supply a power-up reset signal to I/O chips, etc.

To use ELCOMP-1 with other computers, it is only necessary to add an adapter containing the equivalent pin-out of the OSI-Superboard bus-output-connector. ELCOMP will offer boards for the microcomputers mentioned above.

Note 1:
These APPLE slots will not support cards requiring dynamic RAM refresh signals from the APPLE bus, and will not support applications requiring use of the APPLE language card.



Motherboard, holding three expansion cards and 32K RAM-board

Part I

The ELCOMP-1 Expansion Board allows compatibility with many 6502-based systems which access to a growing variety of compact, low-cost APPLE expansion parts (excluding products requiring dynamic RAM-refresh, APPLE monitor or language card features).

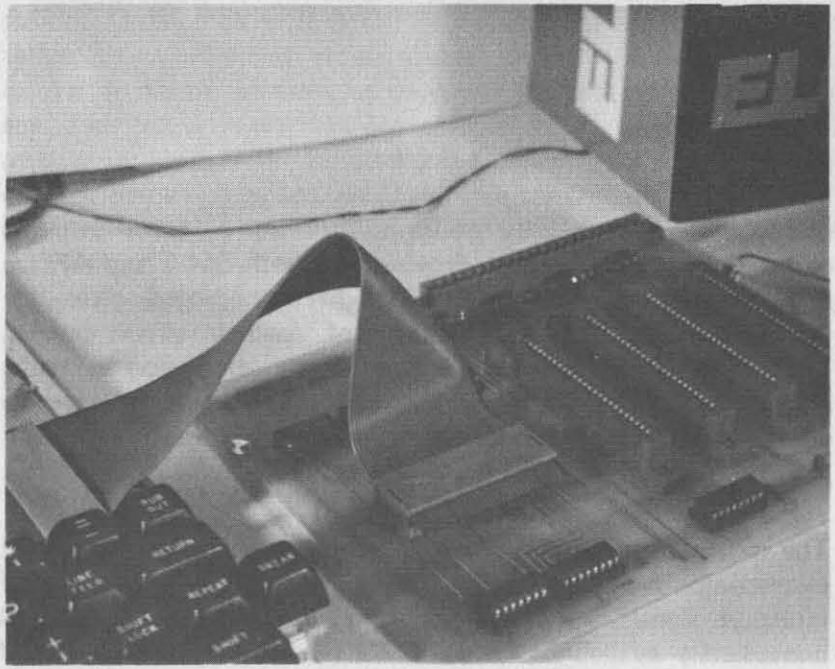
Use of a small motherboard assembly provides small system expansion at a cost consistant with a single board 6502 microcomputer. Many computer hobbyists wish to implement a small interface or memory expansion, but would be happy to avoid the overhead of constructing their own motherboard interface and bus assembly. ELCOMP-1 meets this need.

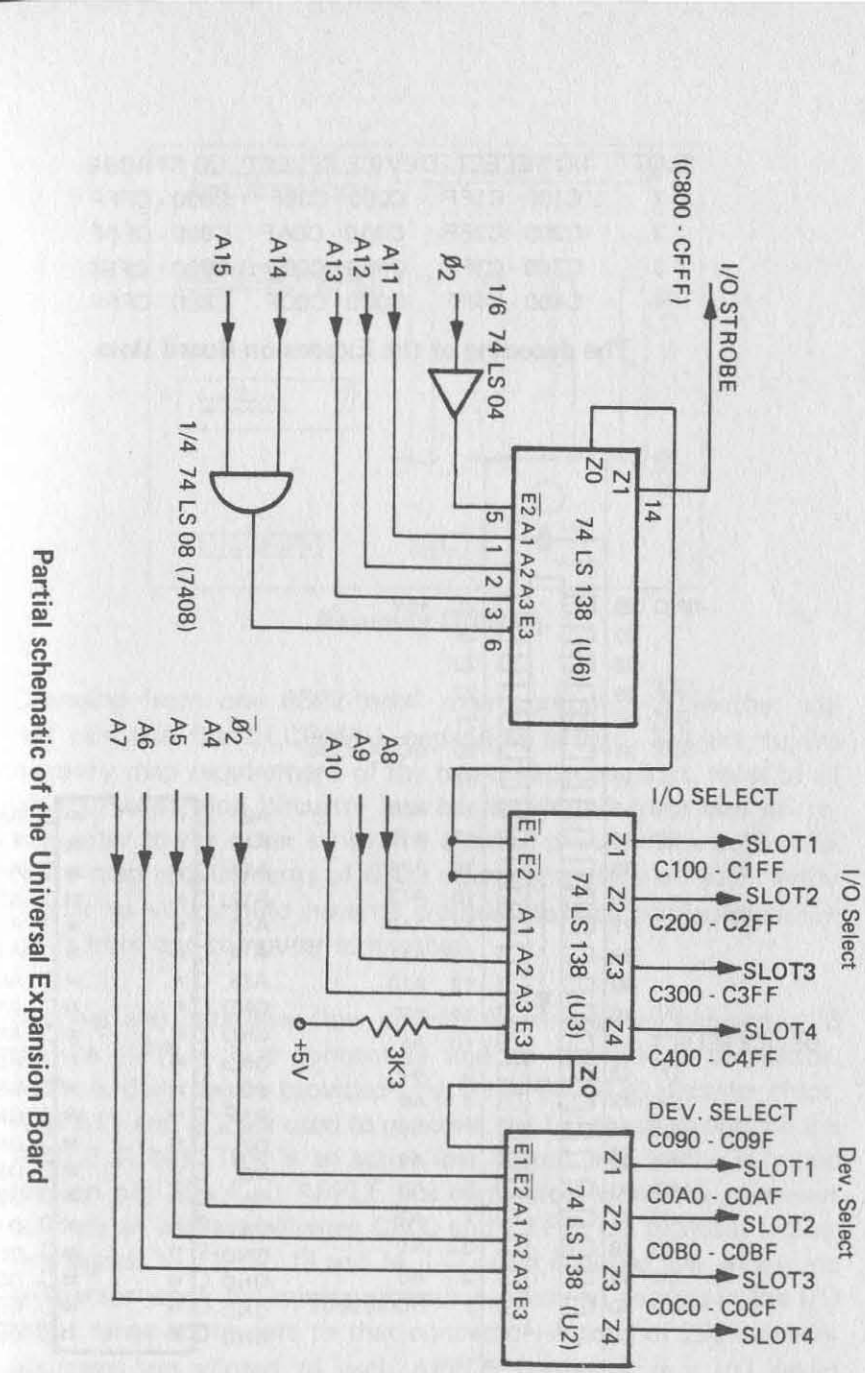
Many expansion items are available for the APPLE, OSI, AIM, KIM, SYM and other 6502 computers, but very few of these provide the flexibility of a bus structure. Those systems which do provide a bus structure are generally available fully assembled. Such systems often cost more than the original single board microcomputer. ELCOMP-1 avoids this excessive cost because it is provided in a minimum KIT form. The computer hobbyist is free to choose the case, power supply, buffering and number of expansion connectors necessary to meet his needs. A typical hobbyist activity might include the addition of a disk interface and/or printer interface to a single board microcomputer system. To support the software requisite to the disk expansion, the hobbyist would typically desire the addition of some amounts of random access memory. Provision of both S44-E and APPLE bus connectors on the ELCOMP-1 motherboard gives the hobbyist access to a wide variety of available circuit boards. The hobbyist may take advantage of the expansion elements already available on the market, and may reserve his creative efforts for software and hardware of particular interest.

The motherboard is easily interfaced to a variety of 6502-based microcomputers. The 6502's simple control bus (consisting of interrupt signals, $\phi 2$ clock, and read/write not signals) is immediately available for all 6502 based microcomputers and

may be emulated with signals available from other microprocessor systems. The ELCOMP-1 may be directly interfaced with an Ohio Scientific Superboard-II using only a 40-conductor ribbon cable. Adapter assemblies are available for most other 6502-based single board microcomputers (including ATARI, Commodore VIC, PET/CBM, AIM, KIM, SYM, etc.).

A special cable assembly connects the internal bus of the APPLE II microcomputer to the ELCOMP-1 motherboard. By extending the APPLE bus outside the APPLE enclosure, ELCOMP-1 enhances the development and service ability of APPLE II circuit boards. The following tables summarize connections to the Superboard interface, the four APPLE slots, and the S44-E connector. Lines not described in the following table are not supported by the ELCOMP-1 motherboard. APPLE compatible circuit cards which require use of APPLE monitor ROM features, APPLE language card features, or dynamic memory refresh feature circuitry of the APPLE II computer may not be used with the ELCOMP-1 motherboard.

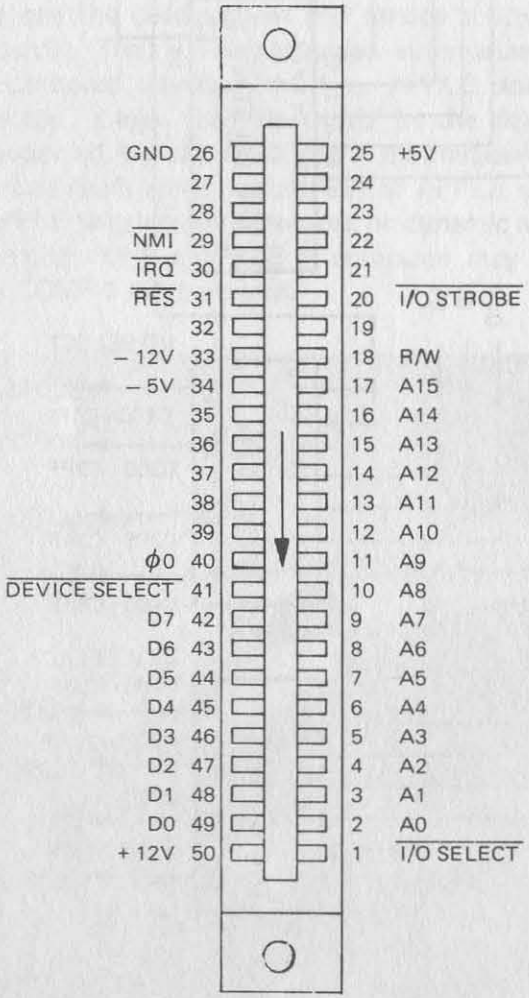




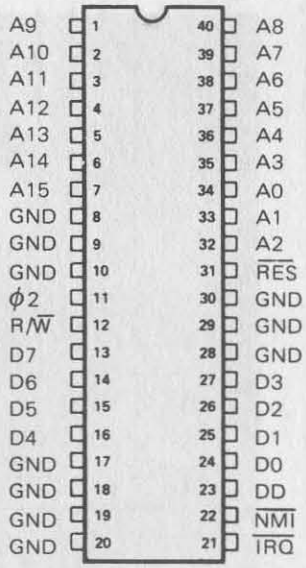
Partial schematic of the Universal Expansion Board

SLOT	I/O SELECT	DEVICE SELECT	I/O STROBE
1	C100 - C1FF	C090 - C09F	C800 - CFFF
2	C200 - C2FF	C0A0 - C0AF	C800 - CFFF
3	C300 - C3FF	C0B0 - C0BF	C800 - CFFF
4	C400 - C4FF	C0C0 - C0CF	C800 - CFFF

The decoding of the Expansion Board slots

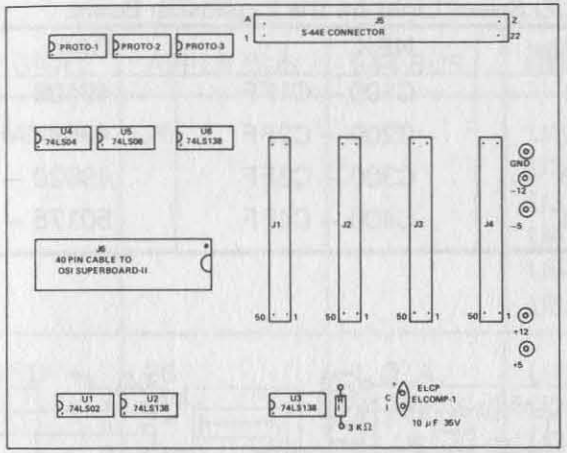


Connectors on the Expansion Board



40-pin connector

on the Superboard



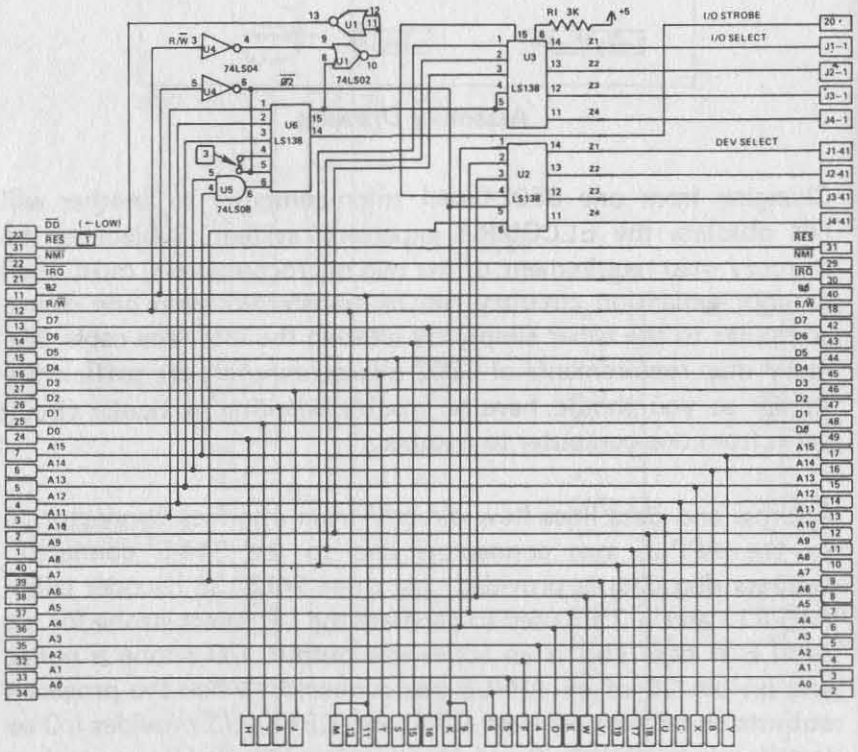
Assembly Drawing

Changing from one 6502-based microcomputer to another will not obsolete the ELCOMP-1 expansion system. Subject to the memory map requirement of the two microcomputers, most of all of your expansion circuitry can be transferred from one microcomputer to the other simply by altering the interface cable. Memory map requirements of 6502 microcomputers are sufficiently similar so you should have no trouble adapting particular circuit cards from one computer to another.

Address and data lines flow directly from interface connector J6 to the APPLE bus connectors and to the S44-E connector. Address decoding is provided by three 74LS138 decoder chips. A15-A11 and ϕ 2 are used to generate the I/O select strobe for the APPLE II bus. This is an active low output. I/O strobe is pulled low on pin 20 of all APPLE bus connectors when the processor outputs an address between C800 and CFFF. U3 provides I/O select signals to J1, J2, J3 and J4. I/O select is pulled low on a given connector when the microprocessor outputs an address in the I/O select range appropiate to that connector. A total of 256 memory addresses are alloted to each APPLE connector for I/O select purposes. A summary of I/O select addresses for each of the four APPLE connectors on the ELCOMP-1 motherboard is provided in the table below.

I/O Select Lines on the Expansion Board

Slot	HEX	DEZ
1	C100 – C1FF	49408 – 49663
2	C200 – C2FF	49664 – 49919
3	C300 – C3FF	49920 – 50175
4	C400 – C4FF	50176 – 50431



Complete schematic of the motherboard

POWER AND SPARES			
SIGNAL	APPLE BUS	S44 BUS	CHIPS
+5V	25	3, 20	U1–14 U2–16 U3–16 U4–14 U5–14 U6–16
GND	26	1, 2, A, 21, 22, Z	U1–7 U2–8 U3–8 U4–7 U5–7 U6–8
–5V	34	–	–
–12V	33	–	–
+12V	50	–	–
N/C	19, 21, 22, 23, 24, 27, 28, 32, 35, 36, 37, 38, 39	12, 14, N, P, R	

NOTES:

[1] $\overline{DD}$ is pulled low by ELCOMP-1 if and only if the CPU tries to read memory between $C000 and $C7FF. This line controls data direction for OSI-C1P off-board data bus transceivers.

[2] These gates add on rev-b. Cut trace to U2–6 on rev-a, and add the inverter and gate shown.

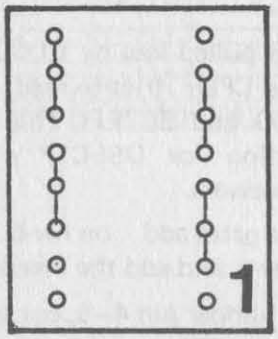
[3] Add jumper pin 4–5, cut trace pin 5–6.

Device select signals for the APPLE II bus are generated by U2. The 16 memory addresses are alloted each APPLE bus connector for device select purposes. A summary of device select addresses for J1 to J4 on the ELCOMP-1 motherboard is provided in the table below.

DEVICE SELECT LINES ON THE EXPANSION BOARD

Slot	HEX	DEZ
1	C090 – C09F	49296 – 49311
2	C0A0 – C0AF	49312 – 49327
3	C0B0 – C0BF	49328 – 49343
4	C0C0 – C0CF	49344 – 49359

A special circuit is provided to control the data direction line on the Superboard II. Data direction is an active low signal controlled by memory or I/O devices external to the Superboard II. The Superboard II provides a pair of 8T28 bus transceivers used only to buffer the data bus to and from external devices. Normally, these bus transceivers are transmitting, or outputting, from the Superboard. If the Superboard attempts to read a memory location within the range of an external memory or I/O device, the net device must pull down on the data direction line. This action reverses the direction of the 8T28 bus transceivers, and permits data to flow from the external device onto the Superboard. In a standard configuration the ELCOMP-1 motherboard pulls down the Superboard data direction line for memory read operations in the address range from C000 – C7FF. If memory or I/O-devices are to be used outside this address range of the Superboard II, then it is necessary to replace the 8T28 buffers with the jumper assemblies shown before.



Top view of the

sockets

(use two TC-platforms)

Alternative connection to any microcomputer is facilitated by a prototyping area provided on the ELCOMP-1 motherboard. For example regarding the data direction line on the Superboard II, additional decoding circuitry is to be added to control the data direction line over a broader address range. A further use of this prototyping area might be in combining the memory read, memory write signals of a Z80 or 8080 microprocessor system into a single read/write, as generated by the 6502.

ELCOMP-1 is designed to operate from an independent power supply. This is in keeping with the intent of expanding a small single board system. Connections are provided from +12, –12, +5, –5 and ground. Active components on the ELCOMP-1 motherboard require +5 volts. Other power supply voltages may be added or deleted to suit the users application. The lack of address and data bus buffering on the ELCOMP-1 requires an expansion card we designed with careful consideration to bus loading. Operation and clock rates more than 1 MHz or the interface cabling more than 15 inches long is not recommended.

Assembly of the ELCOMP-1 motherboard is fast and simple. If the ELCOMP-1 is to be used with the S44-E connector only, then only U4 is required. If the APPLE bus is to be used, then all six integrated circuits must be installed on the ELCOMP-1 motherboard. Sockets are recommended for all integrated circuits, and for the interface connector J6.

Parts List

Quantity	Description
3	1 of 8 Decoder / Demultiplexer 74LS138
1	Quad 2-Input NOR-Gate 74LS02
1	Hex-Inverter 74LS04
1	Quad 2-Input AND-Gate 74LS08
4	50-pin 2.54 mm Double Sides P.C.
1	44-pin S44-connector, female tail edge connector
1	40-pin socket
1	3K3 1/4 Watt resistor
2	0.1 µF Capacitor Ceramic
1	10 µF, 35V Tantal Capacitor

Note that the location of pin 1 on each integrated circuit is marked on the printed circuit board. Also see the assembly drawing below, for location and orientation of all components.

The ELCOMP-1 is compatible with a variety of APPLE II circuit cards and with S44-E circuit cards. The ELCOMP-1 is ideal for interfacing the data computer's 32K dynamic RAM card to Superboard II. ELCOMP operates a variety of expansion and prototyping cards for use for the ELCOMP-1 expansion motherboard.

	S–44E		S–44E
1	GND	A	GND
2	GND	B	A14
3	VCC	C	A12
4	A3	D	A4
5	A1	E	A5
6	A0	F	A2
7	D1	H	D0
8	D2	J	D3
9	$\overline{\phi 2}$	K	$\overline{\phi 2}$
10	$\overline{RES}$	L	RES
11	$\phi 2$	M	$\phi 2$
12	R/$\overline{W}$	N	NC
13	R/$\overline{W}$	P	R/$\overline{W}$
14	NC	R	NC
15	D6	S	D5
16	D7	T	D4
17	A9	U	A11
18	A10	V	A7
19	A8	W	A6
20	VCC	X	A13
21	GND	Y	A15
22	GND	Z	GND

Pinout of the S-44 connector on the expansion board

The 6522 VIA I/O card

The 6522 versatile interface adapter is an extremely powerful I/O chip for 6502 based microcomputer systems. Yet, many 6502 systems have no 6522 chip available, and no provision for adding one. Model 605 experimenter card provides a 6522, 256 bytes of static RAM for user I/O routines, and a large prototyping area. Applications to the 6522 include: EPROM burners, printer controllers, A to D and D to A converters, cassette tape motor control, sound generation, and other miscellaneous control interfaces. In the following chapter we will describe the operation of the 6522 VIA in detail, complete with several application examples.

The 6522 versatile interface adapter constitutes a significant improvement over the original 6820 PIA.The 6502 adds two internal timers, a serial IN/OUT and parallel IN, serial OUT shift register, and latched data inputs on the peripheral ports. Enhanced handshaking permits high-speed data and transfer between multiple CPUs.

The primary feature of the 6522 VIA is a pair of 8-bit bidirectional data ports. Each of the 16 bidirectional lines may be programmed as either an input or output. Some of the lines may be interfaced with the internal interval timers providing the ability to generate specific time intervals, or to measure specific time intervals. The 6522 is controlled by an internal file of 168-bit registers. A table of these registers follows.

Access to the 16 internal registers is controlled by four register select lines, RS0, RS1, RS2, RS3. Interface to the 6502 microprocessor system is via the normal 8-bit data bus, 16-bit address bus (chip select 1, chip select 2), ϕ 2 clock, R/W, interrupt request not output, and master reset. Note the reset input clears all internal registers, placing all bidirectional data lines into the input state, and disabling all special timer and shift register features.

Peripheral port A consists of eight bidirectional data lines which may be individually programmed to act as inputs or outputs, under the control of data direction register A (DDRA). The logical value of output pins is controlled by an output register, and values pre-

Register Number	RS Coding				Register Desig.	Description	
	RS3	RS2	RS1	RS0		Write	Read
0	0	0	0	0	ORB/IRB	Output Register "B"	Input Register "B"
1	0	0	0	1	ORA/IRA	Output Register "A"	Input Register "A"
2	0	0	1	0	DDRB	Data Direction Register "B"	
3	0	0	1	1	DDRA	Data Direction Register "A"	
4	0	1	0	0	T1C-L	T1 Low-Order Latches	T1 Low-Order Counter
5	0	1	0	1	T1C-H	T1 High-Order Counter	
6	0	1	1	0	T1L-L	T1 Low-Order Latches	
7	0	1	1	1	T1L-H	T1 High-Order Latches	
8	1	0	0	0	T2C-L	T2 Low-Order Latches	T2 Low-Order Counter
9	1	0	0	1	T2C-H	T2 High-Order Counter	
10	1	0	1	0	SR	Shift Register	
11	1	0	1	1	ACR	Auxiliary Control Register	
12	1	1	0	0	PCR	Peripheral Control Register	
13	1	1	0	1	IFR	Interrupt Flag Register	
14	1	1	1	0	IER	Interrupt Enable Register	
15	1	1	1	1	ORA/IRA	Same as Reg 1 Except No "Handshake"	

SY6522 Internal Register Summary

sented to input pins may be latched into an input register under control of a handshake line (CA1).

Peripheral port B also consists of eight bidirectional data lines. They are controlled by data direction register and a manner analogous to the control of peripheral port A. For port B has the addes feature that outputs are configured with high current drivers, enabling them to directly drive Darlington output transistors. Peripheral port B and its handshake lines exhibits several special features, operating under the control of the 6522's 16 internal registers.

Some programming examples: DDRA and DDRB control the data direction of the 16 bidirectional data bits in ports A and B, respectively. There is a one to one correspondence between bits and the data direction registers and the 16 bidirectional data bits. If the bit is set on the data direction register, then the corresponding bit in port A or port B will be configured as an output. If a bit in the data direction register is reset, the corresponding bit in port A or port B will be configured as an input.
Example:
To configure all lines of port A as outputs the 6502 host system should execute the following instructions:

```
LDA  # $FF
STA  DDRA
```

The same result could be done from BASIC by executing the following statement:

```
POKE DDRA,255
```

In the reset state all 16 bidirectional data lines are configured as inputs. Those microprocessor system configures outputs, as required, by loading DDRA and DDRB with the needed values.

The exact memory address of the 6522's internal register file depends upon the external decoding circuitry used to drive chip select 1 and chip select 2. However, if you use the ELCOMP-1 expansion board and the model 605 6522 VIA board, then the addresses of the 16 internal registers of the 6522 will be as follows:

SLOT 1 HEX	SLOT 1 DEC	SLOT 2 HEX	SLOT 2 DEC	SLOT 3 HEX	SLOT 3 DEC	SLOT 4 HEX	SLOT 4 DEC	Register Desig.	Description Write	Description Read
C090	49296	C0A0	49312	C0B0	49328	C0C0	49344	ORB/IRB	Output Register "B"	Input Register "B"
C091	49297	C0A1	49313	C0B1	49329	C0C1	49345	ORA/IRA	Output Register "A"	Input Register "A"
C092	49298	C0A2	49314	C0B2	49330	C0C2	49346	DDRB	Data Direction Register "B"	
C093	49299	C0A3	49315	C0B3	49331	C0C3	49347	DDRA	Data Direction Register "A"	
C094	49300	C0A4	49316	C0B4	49332	C0C4	49348	T1C-L	T1 Low-Order Latches	T1 Low-Order Counter
C095	49301	C0A5	49317	C0B5	49333	C0C5	49349	T1C-H	T1 High-Order Counter	
C096	49302	C0A6	49318	C086	49334	C0C6	49350	T1L-L	T1 Low-Order Latches	
C097	49303	C0A7	49319	C0B7	49335	C0C7	49351	T1L-H	T1 High-Order Latches	
C098	49304	C0A8	49320	C0B8	49336	C0C8	49352	T2C-L	T2 Low-Order Latches	T2 Low-Order Counter
C099	49305	C0A9	49321	C0B9	49337	C0C9	49353	T2C-H	T2 High-Order Counter	
C09A	49306	C0AA	49322	C0BA	49338	C0CA	49354	SR	Shift Register	
C09B	49307	C0AB	49323	C0BB	49339	C0CB	49355	ACR	Auxiliary Control Register	
C09C	49308	C0AC	49324	C0BC	49340	C0CC	49356	PCR	Peripheral Control Register	
C09D	49309	C0AD	49325	C0BD	49341	C0CD	49357	IFR	Interrupt Flag Register	
C09E	49310	C0AE	49326	C0BE	49342	C0CE	49358	IER	Interrupt Enable Register	
C09F	49311	C0AF	49327	C0BF	49343	C0CF	49359	ORA/IRA	Same as Reg 1 Except No "Handshake"	

Exact memory addresses for the 6522 registers

The above table shows the exact memory address of each individual register of the 6522 assuming that it is plugged into slot 1 to 4 of the expansion board. It shows the exact memory address in hexadecimal, the exact memory address in decimal, the name of the register, and the description of the register for each of the 16 registers in the chip.

Programming example

Problem:
Program a bargraph display using LEDs and the 6522.

Solution:
Connect the anode of one LED to each of the eight bidirectional data lines of port A. Connect the cathode of the LED to ground through 220Ω resistor. Repeat this for each of the eight bidirectional data lines, as shown below.

Test program:
Board in slot 4.
Go into the monitor and look into location C0C0–C0C3.

```
RESET (BREAK)
C0C0 FF
C0C1 FF
C0C2 00
C0C3 00
```

Test of Ports

Put FF into location C0C2 and C0C3 and then

```
C0C0 00
C0C1 00
C0C2 FF
C0C3 FF
```

Then write

```
C400 00
C401 AA
C402 55
C403 FF
```

Test of RAM

This content should read the same.

The program shown below drives the bargraph display, assuming the 6522 board has been inserted in slot 4 of the ELCOMP-1 expansion board.

With the 6522 in slot 4 the first address of the 6522 register file will be 49344 decimal. The data direction register for port A (DDRA) has an address equal to the base address plus 3. The output register for port A has an address equal to the base address plus 1. This addressing is noted in lines 10 – 50 of the program, and the address values are assigned to variables in line 70 and 80. Line 90 begins a FOR...NEXT loop which counts its way to the eight bits of port A. The remark statements in lines 160–250 describe the values assigned to the variables bit and data,as well as the number of LEDs turned on, as the program steps through the FOR...NEXT loop.

```
10   REM BARGRAPH DRIVER
20   REM 6522 IN SLOT 4 OF ELCOMP
30   REM FIRST SLOT 4 ADDRESS = 49344
40   REM DDRA = BASE ADDRESS + 3 = 49347
50   REM PORT A (ORA) = BASE ADDRESS + 1 = 49345
60   REM
70   DDRA = 49347
80   PA = 49345
90   FOR BIT = 0 TO 8
100  DATA = (2 ↑ BIT)–1
110 POKE PA, DATA
120  REM KILL SOME TIME
130  FOR DELAY = 0 TO 1000 :NEXT DELAY
140  NEXT BIT
150  GOTO 90
160  REM VALUE OF BIT  DATA    NUMBER OF LED's ON
170  REM    0           0            0
180  REM    1           1            1
190  REM    2           3            2
200  REM    3           7            3
210  REM    4          15            4
220  REM    5          31            5
230  REM    6          63            6
240  REM    7         127            7
250  REM    8         255            8
```



```
10  REM  BARGRAPH DISPLAY
20  REM  BOARD IN SLOT 4
30 DDRA = 49347:TA = 49345
40  POKE DDRA,255
50 A = 1
60  POKE TA,A
70  GOSUB 200
80 A = A * 2
90  IF A = 256 THEN A = 1
100  GOTO 60
200  REM  TIME DELAY
210  FOR I = 1 TO 50
220  NEXT I: RETURN
```

Programming example:

Using the timers

Interval timer T1 consists of a 16-bit counter and two 8-bit latches. The latches are used to store a pre-set value for the counter. Subsequent to loading from the 8-bit latches, the counter decrements at the system clock rate (ϕ 2 clock). When the counter reaches a count of zero, an internal interrupt flag will be set, and the IRQ pin will be set low, if interrupts are enabled. Timer 1 can be programmed to do several things at this point.Two bits in the auxiliary control register (ACR) determine timer 1 operating mode. If bit 7 in the auxiliary control register is set (1), then the 6522 will reverse the polarity of the signal at bit # 7 of port B. Each time the timer 1 reaches the count of zero. If bit 7 of the auxiliary control register is reset (0), then bit 7 of port B will be disabled. If bit 6 of the auxiliary control register is set, then upon reaching a count of zero, timer 1 will automatically reload with its preset value and continue counting. This is referred to as the "FREE-RUN" mode. If bit 6 of the auxiliary control register is reset, then timer 1 will halt upon reaching a count of zero. As always, an interrupt will be output, if enabled. In this "one-shot" mode, timer 1 will not run again until reloaded by the host microprocessor.

Four of the 6522's 16 internal registers directly affect the operation of timer 1. These registers are summarized in the table below:

Register Select Bits

RS3	RS2	RS1	RS0	Operation
L	H	L	L	Write into low-order latch.
L	H	L	H	Write into high-order latch Write into high-order counter. Transfer low-order latch into low order counter. Reset T1 interrupt flag.(IFR6)
L	H	H	L	Write low-order latch.
L	H	H	H	Write high-order latch. Reset T1 interrupt flag. (IFR6)

Note that the processor does not write directly into the low-order counter (T1C-L). Instead, this half of the counter is loaded automatically from the low-order latch when the processor writes into the high-order counter.

Note: The last two entries of the above table load the pre-set latch with new values without affecting the current contents of the counter, or the count in progress. Note that execution of the second and fourth entries in the above table cause the internal interrupt flag to be reset. If timer 1 is operating in "one shot" mode, than a new count would not be initiated unless the second entry in the above table were executed. Note that the port B, bit 7 output enable from timer 1 will overwrite bit 7 in the port B data direction register.

Problem:
Program a square wave with 100 ms period.

Solution:

```
LDA   # $C0; select operation mode
STA   ACR
LDA   # $50; load least significant byte of counter
STA   T1C-1; counter
LDA   #$C3;  load high byte
STA   T1C-H; counter
```

Execution of the last command of the above program transfers the pre-set value from the two 8-bit latches into the low byte and the high byte of the 16-bit counter, clears the interrupt flag, and allows timer 1 to begin decrementing. When timer 1 reaches the count of zero it would invert the state of port B bit 7, reload itself from the 8-bit latches, and begin counting down once again.

Port B, pin 7 will therefore demonstrate a square wave signal, whose period will be 100 ms.
A BASIC program to achieve the same result follows:

```
10 REM PULSE GENERATOR
20 REM PUT YOUR 6522 INTO SLOT 4
30 REM ELCOMP-1 EXPANSION BOARD
40 ACR = 49355
50 T1CL = 49348: T1CH=49349
60 POKE ACR,192: REM C0 IS OPERATION MODE
70 POKE T1CL,80: REM 50 HEX
80  POKE T1CH,195: REM C3 HEX
```

```
10   REM   SQUAREWAVE GENERATOR
20   REM   BOARD IN SLOT 4
30 ACR = 49355
40 T1CL = 49348:T1CH = 49349
50   POKE ACR,192
60   POKE T1CL,78
90   POKE T1CH,199
100   END
```

A single shot pulse could be generated, instead of a square wave, by selecting operation mode 80 hex instead of C0 hex. A different square wave frequency could be programmed by substituting different values for the 8-bit latches (program lines 70 and 80).

Operation of timer 2 is similar to operation of timer 1, except that timer 2 must be operated in one shot mode.

Timer 2 may also be used to count external events, rather than operate as a timer clock triggered by $\phi 2$ clock. In this mode the 6522 counts the negative going pulses on port line PB6. Operation mode of timer 2 is controlled by bit 5 of the auxilliary control register. If bit 5 of the auxiliary control register is reset (0), then

timer 2 operates in one-shot mode. If bit 5 of the auxiliary control register is set (1), then timer 2 counts negative going pulses at port B bit 6.

A very large interval counter may be configured by connecting PB7 to PB6. Timer 1 is operated in its free running mode, with PB7 inverted each time timer 1 reaches a count of 0. Timer 2 is operated in the event counter mode. Each time that a zero count of timer 1 causes PB7 to switch from logic high to logic low, timer 2 will be clocked. The contents of timer 2 will therefore indicate elapsed time. A timer will be disabled by using the auxiliary control register to disable the timer 1 output to PB7.

Use of the 6522 shift register

Register 10 of the 6522 chip is a shift register. This register may be used to convert parallel data to a serial data stream or vice versa. 3 bits of the auxiliary control register select the various shift register operating modes. The 6522 provides for three serial input modes and four serial output modes. These modes differ as to the source of the shift register clock.

With auxiliary control register bit 6 set equal to zero, the shift register is either disabled or placed in one of its three serial input modes. If auxiliary control register bits 3 and 2 are also zero, then the shift register is disabled. If ACR3 = 0, and ACR2 = 1, then serial data will be clocked into the shift register under control of timer number 2. IF ACR3 = 1, and ACR2 = 0, then serial data will shift in at the system clock rate.

ACR4	ACR3	ACR2	
0	0	0	Shift Register Disabled
0	0	1	Shift in under Control of Timer 2
0	1	0	Shift in at system clock rate
0	1	1	Shift in under Control of External Input Pulses

In mode 001, the shift register clock is controlled by the lower byte of timer 2. The shift register clock is made available at CP1. The frequency of the shift register clock is therefore dependent upon the system clock rate, and the contents of a lower byte of timer 2. The shift in operation is enabled each time the microprocessor system reads or writes the shift register. The internal

shift register interrupt flag is set following the 8-shift register clock.

In shift register mode 010, $\phi 2$ is used as the shift register clock. As above, the shift in operation is initiated when the microprocessor reads or writes the shift register. The shift register interrupt flag is set after the ninth circle of $\phi 2$. At this time, clock output pulses at CP1 stop.

In shift register mode 011, an external device X is the shift register clock. CP1 is configured as an input, and provides this external clock to the 6522 shift register. The shift register interrupt flag will be set each time 8 bits have been loaded into the shift register. However, unlike modes 001 and 010, the shift register will not stop inputting data. The shift register interrupt flag will be reset if the microprocessor reads or writes the shift register.

Note that in all shift register modes serial data is clocked first into the least significant bit of the shift register. On each successive clock pulse, data is shifted towards the most significant bit of the shift register. Note also that serial data is loaded into the shift register, while $\phi 2$ is low, immediately following the rising edge of the shift clock. Serial input data must be stable during this time period.

In each of the four shift register output modes CB2 is configured as an output. Shift register bit 7 is the first bit output to CB2. At the same time that bit 7 is output to CB2, it is also shifted back into bit 0 of the shift register. Therefore, shift register data does not disappear out of CB2, rather, it circulates to the shift register. As with the shift-in modes shifting occurs, while ϕ 2 is low, immediately subsequent to the rising edge of the shift clock.

Shift out mode 100 is a free-running mode. The frequency of the shift clock is controlled by the contents of timer 2. The shifting process never stops, and the eight shift register bits are clocked out of CB2 repetitively.

The shift rate of serial output mode 101 is also controlled by timer 2. Unlike mode 100, each read or write of the shift register

causes eight bis to be shifted onto CB2. The shift clock is output under CB1. Following eight shift clocks, shifting halts, and the shift register interrupt flag is set.

Shift register output mode 110 is very similar to mode 101. However, the shift clock is now directly controlled by $\phi 2$, and is independent of timer 2.

Shift register output mode 111 provides for data output under control of an external clock. The external clock is applied to the 6522 via CB1, which is configured as an input. The shift register interrupt flag is set each time that eight bits have been output onto CB2, however, although the shifting process is not halted.The shift register interrupt flag is reset each time the microprocessor reads or writes the shift register.

Note that in both the serial input and serial output modes, data flows from shift register bit 0 towards shift register bit 7. Therefore, shift register bit 7 is the first bit out during a serial output operation, and shift register bit 0 is the first bit loaded during a serial input operation.

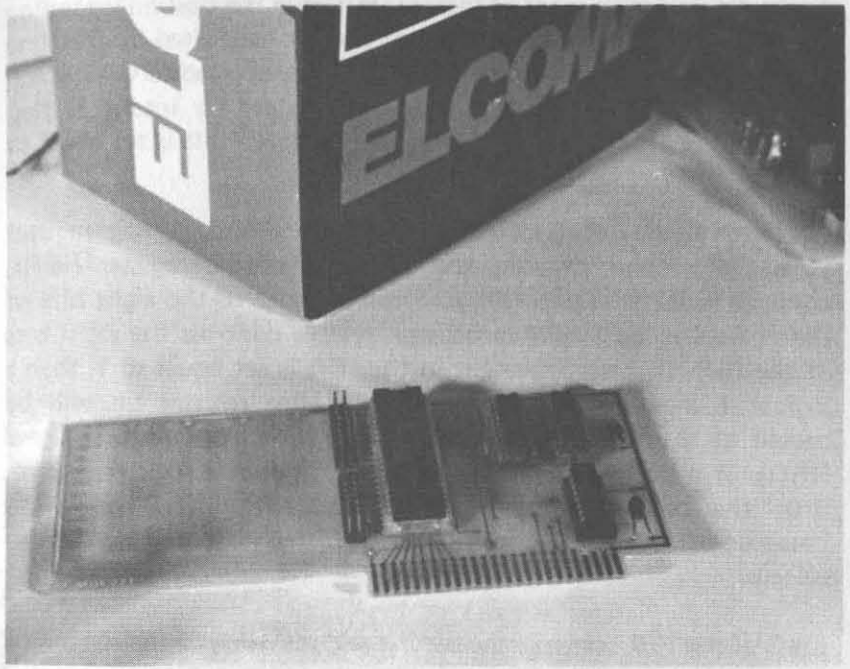
Programming example:

Problem:
Simulate a pulse generator with variable frequency and duty circle.

```
LDA   #$FF      ; SET TIMER FOR
STA   T2L-L     ; LOWEST FREQUENCY
LDA   #$10      ; OPERATION MODE
STA   ACR       ; FREE RUNNING
LDA   #$0F      ; OUTPUT 4 x 0 and 4 x 1
STA   SR        ;
JMP   Monitor   ; JUMP BACK TO MONITOR
```

In this example the 6522 shift register is configured in serial output mode 100. The pulse output is taken from CB2. Frequency of the wave form is determined by timer 2. The smaller the value in timer 2, the greater the frequency with which timer 2 reaches 0, providing an increased shift clock frequency. The duty cycle of the pulse is determined by the contents of the circulating shift

register. Loading four adjacent shift register bits with logical 1's, and loading the remaining shift register bits with logical 0's, we produce a pulse with 50 per cent duty cycle. Setting additional shift register bits, we increase the duty cycle of the wave form.



The 6522 VIA card with 1K of RAM and a large prototyping area

A BASIC program could also be used to solve the above problem.

```
10 REM FREERUNNING SWG
20 REM WITH VARIABLE DUTYCYCLE
30 REM PUT YOUR 6522 VIA BOARD IN SLOT 4
50 ACR = 49355 : T2L = 49352
60 SR = 49354
70 POKE T2L,255
80 POKE ACR,16
90 POKE SR,15
100END
```

A one-shot pulse of variable width could be produced from the above example if serial output mode 101 were selected. In this case the contents of the shift register would output ones,

the shifting process would halt, and the host microprocessor system would be interrupted.

Interrupt control

There are several sources of interrupts within the 6522. Depending upon the mode selected, interrupts may be generated by the time out of timer 1 or timer 2, by external signals applied to CA1, CA2, CB1, CB2, or by activities of the shift register. An interrupt flag bit is set equal to 1, if any of the previous seven bits is set equal to 1.

Together, these eight bits constitute the interrupt flag register (IFR). A second register, the interrupt enable register (IER), contains eight bits with a 1-to-1 relationship to the eight bits of the IFR. The host microprocessor system controls the eight bits of the IER. If a particular bit in the IER is set equal to 1, then a logical 1 in a corresponding interrupt flag register bit will be passed to the host microprocessor as an interrupt request signal (IRQ). If a particular bit in the interrupt enable register is zero, then the corresponding bit in the interrupt flag register is "masked-off", and will not interrupt the host microprocessor system.

Bit 7 of the IER controls the setting and resetting of the remaining seven bits in the IER.

Case 1: Suppose the microprocessor writes a byte of data to the IER, such that bit 7 of this byte is set equal to 1. In this case, each logical 1 written to bits 0 – 6 of the IER will cause the corresponding bit of the IER to be set equal to 1.

Case 2: Suppose the microprocessor sends a byte of data to the IER such that bit 7 of the data byte is set equal to zero. In this case, each one written to bits 0 – 6 of the IER will cause the corresponding bit in the IER to be cleared.

In either case, writing a 0 to bit 0–6 of the IER leaves the selected bits unaffected. The microprocessor system is also free to read the contents of the IER. In this case bit 7 will always be read as a logic 0.

Programming example:

Problem:

Write a program subroutine to generate a one- shot pulse at PB7. The processor should not return from the subroutine until the pulse is terminated.

```
MONOFL: LDA  #$80   ; SELECT MONOFLOPS
        STA  ACR            OPERATION
        LDA  #$50
        STA  TICL   ; SET TIME
        LDA  #$C7
        STA  TICH   ; SETS BIT 6 IFR=0
MO      LDA  IFR
        AND  #$40   ; MASK BIT 6 of IFR
        BEQ  MO
        RTS
```

Line 1

Selected timer 1 one-shot mode, enable output to PB7.

Line 3

Set timer 1 low order latch

Line 6

Write to higher order latch of timer one, write to higher order byte of timer 1 counter, transfer low order latch into low order counter, also clears timer 1 interrupt flag (IFR-bit 6) in interrupt flag register.

Line 7

Poll IFR to see if timer 1 ended.

Bits 0, 1, 3, and 4 of the IFR areusedfor data transfer between the processor and external devices. Each of these bits may be set and reset by using the peripheral control register (PCR).

PCR

Bit	7	6	5	4	3	2	1	0
	CB2			CB1	CA2			CAI

Bit 0 or the peripheral control register determines which edge of the power supply to CA1 will set bit 1 in the interrupt flag register. If this bit is a 0 the negative edges applied to CA1 will cause the CA1 interrupt flag to be set. If this bit is a 1 then positive edges at CA1 will set the CA1 interrupt flag. Reading or writing peripheral port A will reset CA1 interrupt flag.

If the shift register function is enabled, CB1 will act as an input or output for the shift clock. If the shift register function is disabled, then CB1 may be used in identical fashion to CA1. If PCR4 is a logic 1, the negative edges applied to CB1 will set the CB1 interrupt flag(Bit 4, IFR).
If PCR4 equals zero, then the CB1 interrupt flag will be set by positive edges applied to CB1. The CB1 interrupt flag may be cleared by reading or writing peripheral port B.

CA2 may act as an interrupt input, or as a control output. A table summarizing available CA2 modes follows.

PCR3	PCR2	PCR1	Mode
0	0	0	CA2 Negative Edge Interrupt (IFR0/ORA Clear) Mode -- Set CA2 interrupt flag (IFR0) on a negative transition of the input signal. Clear IFR0 on a read or write of the Peripheral A Output Register (ORA) or by writing logic 1 into IFR0.
0	0	1	CA2 Negative Edge Interrupt (IFR0 Clear) Mode -- Set IFR0 on a negative transition of the CA2 input signal. Reading or writing ORA does not clear the CA2 interrupt flag. Clear IFR0 by writing logic 1 into IFR0.
0	1	0	CA2 Positive Edge Interrupt (IFR0/ORA Clear) Mode -- Set CA2 interrupt flag on a positive transition of the CA2 input signal. Clear IFR0 with a read or write of the Peripheral A Output Register.
0	1	1	CA2 Positive Edge Interrupt (IFR0 Clear) Mode -- Set IFR0 on a positive transition of the CA2 input signal. Reading or writing ORA does not clear the CA2 interrupt flag. Clear IFR0 by writing logic 1 into IFR0.
1	0	0	CA2 Handshake Output Mode -- Set CA2 output low on a read or write of the Peripheral A Output Register. Reset CA2 high with an active transition on CA1.
1	0	1	CA2 Pulse Output Mode -- CA2 goes low for one cycle following a read or write of the Peripheral A Output Register.
1	1	0	CA2 Output Low Mode -- The CA2 output is held low in this mode.
1	1	1	CA2 Output High Mode -- The CA2 output is held high in this mode.

With the shift register function enabled, CB2 is dedicated as to serial input/output from the 6522. If the shift register function is disabled, then CB2 functions in a manner analgous to CA2.

PCR7	PCR6	PCR5	Mode
0	0	0	CB2 Negative Edge Interrupt (IFR3/ORB Clear) Mode -- Set CB2 interrupt flag (IFR3) on a negative transition of the CB2 input signal. Clear IFR3 on a read or write of the Peripheral B Output Register (ORB) or by writing logic 1 into IFR3.
0	0	1	CB2 Negative Edge Interrupt (IFR3 Clear) Mode -- Set IFR3 on a negative transition of the CB2 input signal. Reading or writing ORB does not clear the interrupt flag. Clear IFR3 by writing logic 1 into IFR3.
0	1	0	CB2 Positive Edge Interrupt (IFR3/ORB Clear) Mode -- Set CB2 input signal. Clear the CB2 interrupt flag on a read or write of ORB or by writing logic 1 into IFR3.
0	1	1	CB2 Positive Edge Interrupt (IFR3 Clear) Mode -- Set IFR3 on a positive transition of the CB2 input signal. Reading or writing ORB does not clear the CB2 interrupt flag. Clear IFR3 by writing logic 1 into IFR3.
1	0	0	CB2 Handshake Output Mode -- Set CB2 low on a write ORB operation. Reset CB2 high with an active transition of the CB1 input signal.
1	0	1	CB2 Pulse Output Mode -- Set CB2 low for one cycle following a write ORB operation.
1	1	0	CB2 Manual Output Low Mode -- The CB2 output is held low on this mode.
1	1	1	CB2 Manual Output High Mode -- The CB2 output is held high in this mode.

The auxiliary control register has been discussed in some detail at this point, however, two functions of the register remain unexplained. Bit 0 and 1 of the auxiliary control register (ACR), control the input data watching function through peripheral ports A and B. ACR0 is the port A input latch enable. Input latching is enabled by setting this bit equal to 0. Input data will be latched when the CA1 interrupt flag is set. Reading port A resets the CA1 interrupt flag. This arrangement permits simple handshaking with external devices. If several port A pins are configured as outputs, and if input latching is enabled, then reading port A will transfer eight bits from the data input latches to the processor, although input latch bits corresponding to pins configured as outputs do not necessarily represent the state of those output pins. Careful planing is required to combine input data latching with intermixed output pins.

ACR1 controls the input data latching function through peripheral port B. Exactly as ACR0 controls latching for peripheral port A.

Note, however, that port B data latches will indicate the level apply either to an input pin or the contents of the port B output register depending upon whether a pin is configured as an input or an output.

Stop watch

You also can use the Timer as a counter. In this mode the negative going edges of PB6 are counted. Bit 5 of the ACR-register sets the operation mode.

ACR5	Operation mode
0	Monoflops
1	Count negative pulses on PB 6

If you connect PB6 and PB7 and operate timer 2 as a counter and timer 1 as a freerunning pulse generator you can simulate a stop watch for measuring the execution time of your programs.

The program consists of two parts. A BASIC program and a machine language subroutine in the 1K RAM area on the 6522 VIA board.

You can select the C400 hex area via the DIP switches on the 6522 board.
The elapsed time is determined in location C426 hex . It will be put in units of 1/100 seconds into memory locations C4FE and C4FF hex . At this location the BASIC program picks up the time value to print it out on the screen.

```
1   REM  TEST FOR TIME DELAY
2   REM  BOARD IN SLOT 4
3   REM  E.FLOEGEL NOV 80
10 START = 50188:FIN = 50214
15 HI = 50430:LO = 50431
100  CALL START:
110  GOSUB 1000
200  CALL FIN: GOSUB 990: END
990  PRINT "EXECUTION TIME=";
992 H% =  PEEK (HI):L% =  PEEK (LO)
994  PRINT (H% * 256 + L%) / 100;" SECONDS"
999  RETURN
1000 Q = 2.5:B = 1.2:C = 3.4
1002 E = 1 / Q
1005  FOR I = 1 TO 100
1010 A = (B + C) * Q
1020  NEXT I
1030  RETURN
```

```
C400-   20 0C C4    JSR   $C40C
C403-   4C 00 FE    JMP   $FE00
C406-   20 26 C4    JSR   $C426
C409-   4C 00 FE    JMP   $FE00
C40C-   A9 E0       LDA   #$E0
C40E-   8D CB C0    STA   $C0CB
C411-   A9 01       LDA   #$01
C413-   8D C8 C0    STA   $C0C8
C416-   A9 00       LDA   #$00
C418-   8D C9 C0    STA   $C0C9
C41B-   A9 EC       LDA   #$EC
C41D-   8D C4 C0    STA   $C0C4
C420-   A9 13       LDA   #$13
C422-   8D C5 C0    STA   $C0C5
C425-   60          RTS
C426-   AD C8 C0    LDA   $C0C8
C429-   8D FE C4    STA   $C4FE
C42C-   AD C9 C0    LDA   $C0C9
C42F-   8D FF C4    STA   $C4FF
C432-   38          SEC
C433-   A9 00       LDA   #$00
C435-   ED FE C4    SBC   $C4FE
C438-   8D FE C4    STA   $C4FE
C43B-   A9 00       LDA   #$00
C43D-   ED FF C4    SBC   $C4FF
C440-   8D FF C4    STA   $C4FF
C443-   60          RTS
```

```
*
C400- 20 0C C4 4C 00 FE 20 26
C408- C4 4C 00 FE A9 E0 8D CB
C410- C0 A9 01 8D C8 C0 A9 00
C418- 8D C9 C0 A9 EC 8D C4 C0
C420- A9 13 8D C5 C0 60 AD C8
C428- C0 8D FE C4 AD C9 C0 8D
C430- FF C4 38 A9 00 ED FE C4
C438- 8D FE C4 A9 00 ED FF C4
C440- 8D FF C4 60
*
```

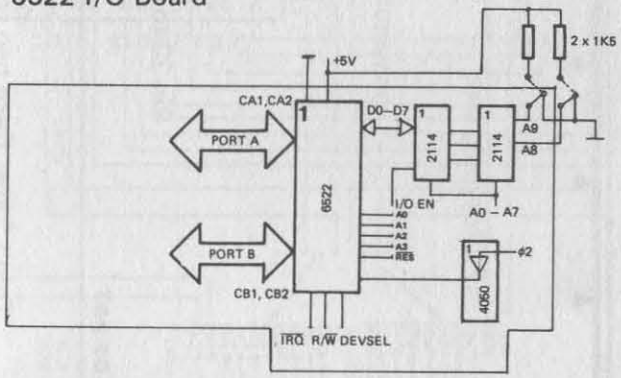

Assembly of the 6522 VIA-board

Assembly of the 605 6522 VIA experimenter card

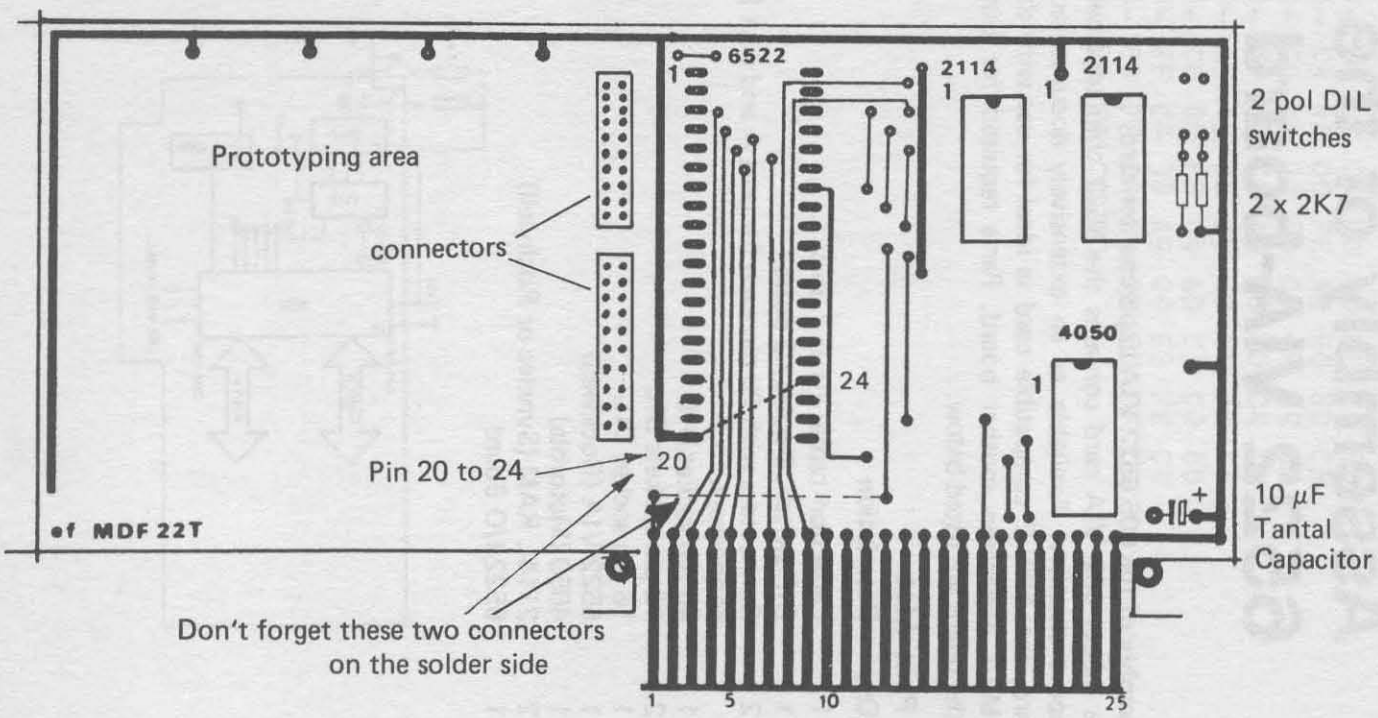
The model 605 VIA card contains the 6522 chip together with support circuitry. Available as an extensively documented bare board, the APPLE compatible card is ideal for use with the EL-COMP-1 expansion mother board. Parts required for assembly of the board are listed below.

Part List

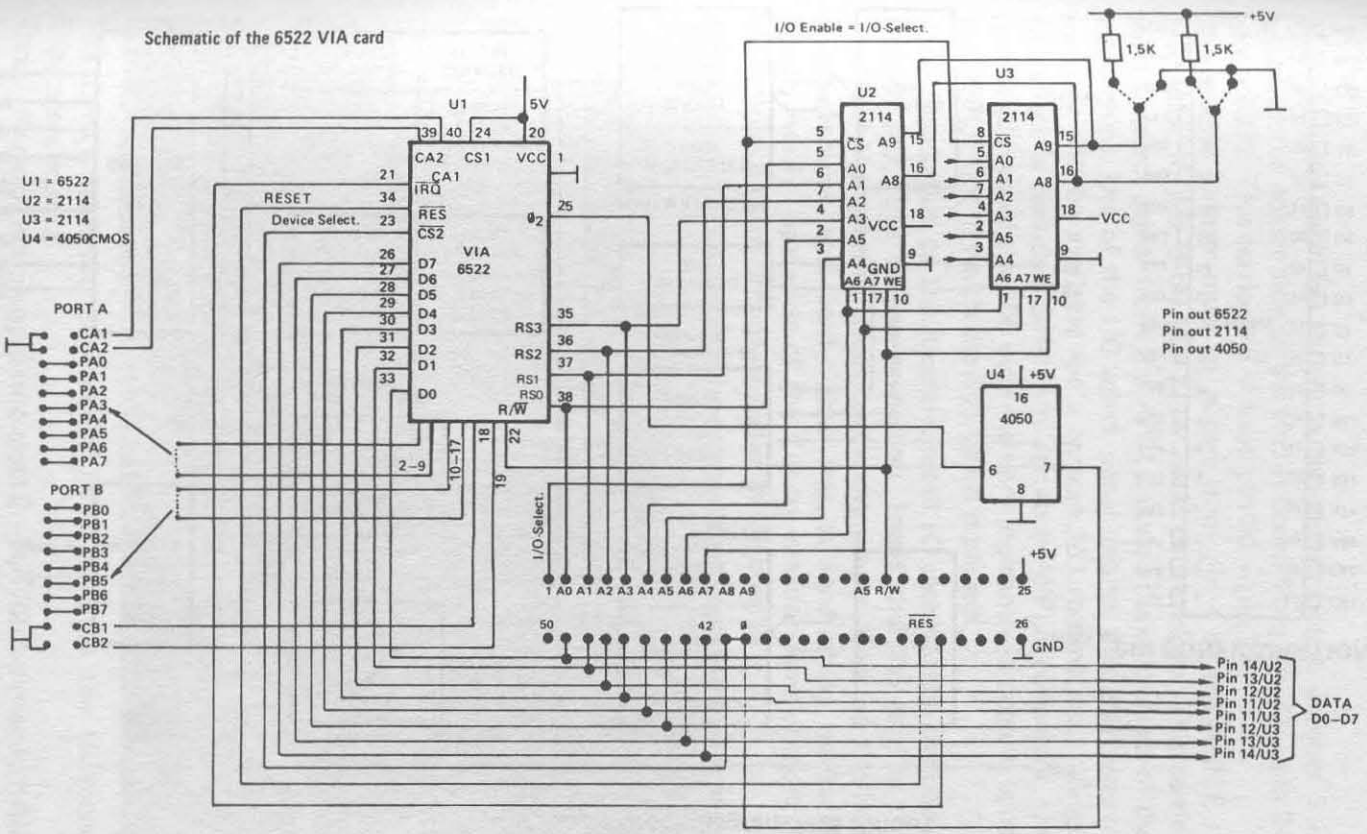
Qty	Description
1	Capacitor tantal 10 uF / 35 V
1	DIP-switch, 2 poles / 3 poles
2	Connectors with 20 pin each for port A and port B connectors.
1	40 pin socket DIL
2	18 pin sockets DIL
1	16 pin socket DIL
1	6522 VIA (Rockwell)
1	4050 (Motorola)
2	2114L RAM (Synelec or Rockwell)
1	6522 I/O Board

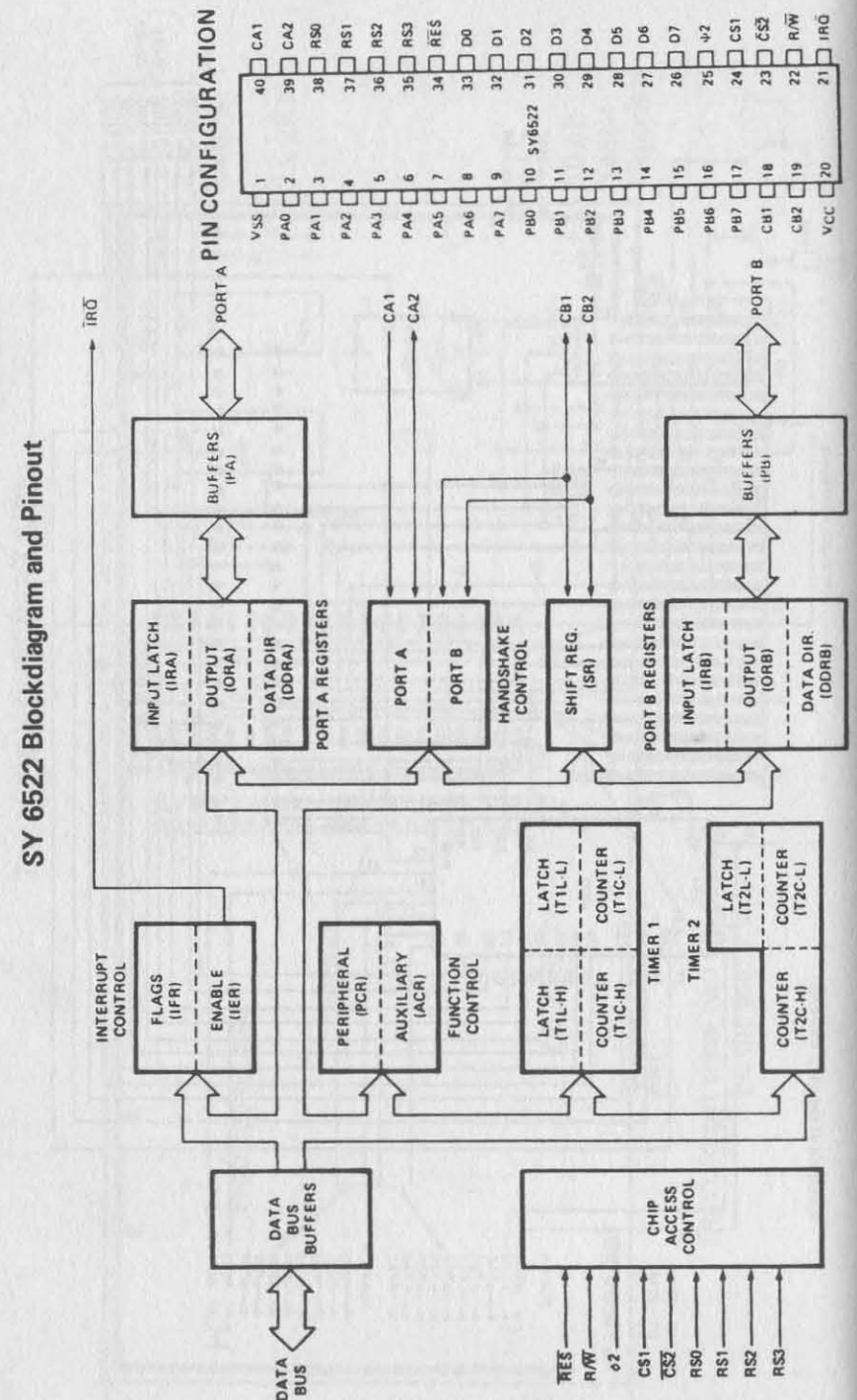


TOP VIEW (COMPONENT SIDE)



Schematic of the 6522 VIA card

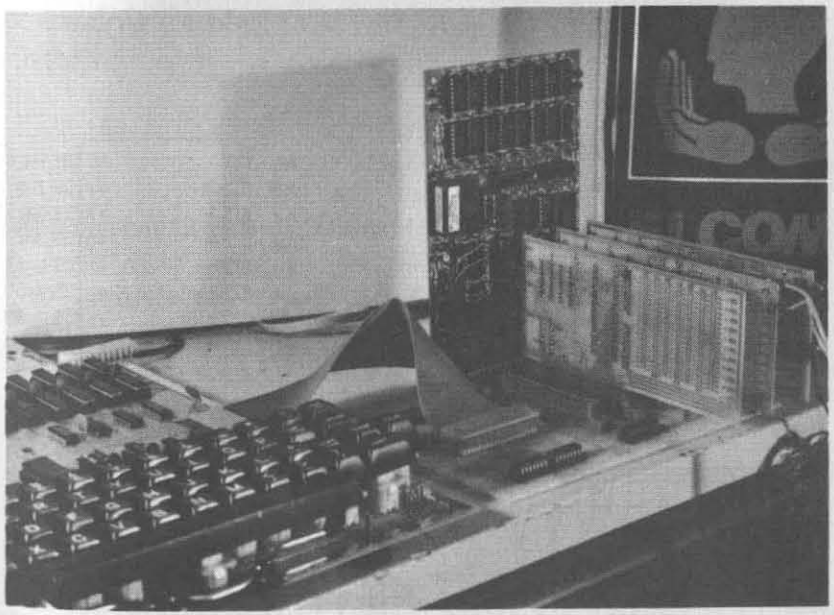




Two jumpers must be added to the circuit board. Jumper 1 is added to the solder side of the circuit board from pin 20 – pin 24 of the 6522. This carries + 5 V from pin 20 – pin 24.

The second jumper must be added from pin 1 of the APPLE II connector to pin 8 on the 2114 RAM chip sockets. This carries the I/O select signal from the APPLE bus to the chip select of the RAM chips. Use of the I/O select signal implies that 256 bytes of memory will be available for a custom I/O routine, etc. Since I/O select is decoded differently for each APPLE connector, the precise address of the 256 byte memory segment will depend upon the APPLE slot in which the 605 board is placed.

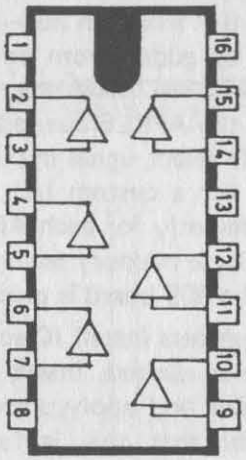
After installation of the jumpers install IC sockets with the 6522, C4050, and 2114 devices as desired. Insert the 605 card in the ELCOMP-1 expansion chasis and apply power. Boot up the host microcomputer and verify that this is functioning normally. Finally, verify operation of the VIA by executing one or more of the samples of programs given earlier in the chapter.



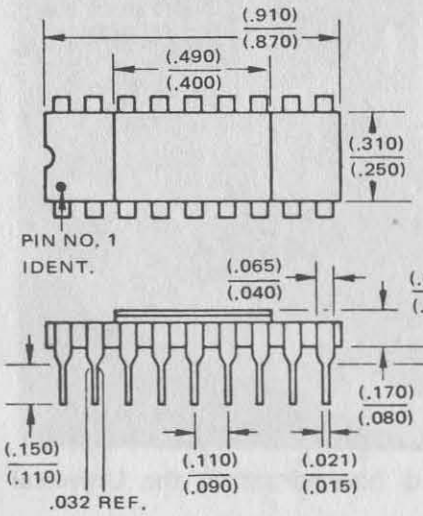
The OHIO Scientific Superboard hooked up to the Universal Expansion Board.

Three I/O-cards are plugged into slots 2–4. A 32 K dynamic RAM card in the S-44 slot gives you 40K of usable RAM.

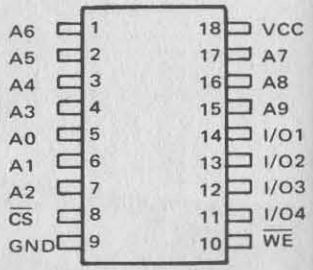
TOP VIEW 4050 C MOS BUFFER



R2114 1024 x 4 STATIC RANDOM ACCESS MEMORY



Package Dimensions



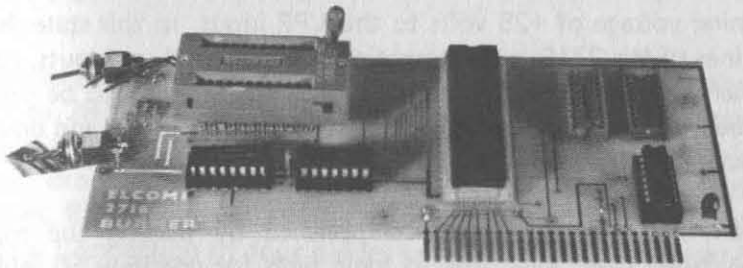
Pin Configuration

2716 EPROM Burner

Typical OHIO Scientific

2716 EPROM-Burner

The model 607 EPROM programmer may be used to program 2716 type EPROMs which use a single +5V supply. The model 607 is a ideal for use with the ELCOMP-1 expansion board.



The 2716 has five states. These are:

- READ
- STAND-BY
- PROGRAM
- PROGRAM VERIFY and
- PROGRAM INHIBIT

A given state is selected according to the signals present on three inputs. These are:

- CHIP ENABLE (pin 18),
- OUTPUT, ENABLE (pin 20, and
- PROGRAMMING VOLTAGE (pin 21).

The names of these three inputs are abbreviated as: $\overline{CE}$, $\overline{OE}$, and VPP. See the table below.

2716 STATES

STATE	$\overline{CE}$(18)	$\overline{OE}$(20)	Vpp	Vcc
READ	LOW	LOW	+5	+5
STANDBY	HIGH	Don't Care	+5	+5
PROGRAM	Pulsed Low to HIGH	HIGH	+25	+5
PROGRAM VERIFY	LOW	LOW	+25	+5
PROGRAM INHIBIT	LOW	HIGH	+25	+5

To program a location in the 2716 it is necessary to set chip enable to logic low, output enable to logic high, and to apply programming voltage of +25 volts to the VPP input. In this state the data lines of the 2716 are automatically configured as inputs. An external device must latch the address of the location to be programmed, as well as the data intended for that location, and drive the address and data inputs of the 2716.

Programming may then be accomplished by pulsing the chip enable input from logic low to logic high for precisely 50 milliseconds.

Several things are therefore required to make a 2716 EPROM programmer. First we need 11 latched data outputs to hold the 2716 address lines stable eight bidirectional lines are required to control the 2716 data lines. It is necessary that we write data to the 2716 for programming purposes, and read back data to confirm that individual locations have been corrrectly programmed. Two additional bits are required for controlling the chip enable and output enable inputs of the 2716. Finally it is necessary to provide some means of applying and removing the 25V programming voltage. The 6522 provides much of the needed capability of the single chip. Most of the necessary inputs and outputs are available directly from the 6522. In addition to this, the 6522 contains an interval timer which greatly simplifies the task of generating the required 50 millisecond pulse.

As shown in the attached schematic, 6522 port A drives the 2716 data bits. Seven bits from 6522 port B drive the seven least significant address bits of the 2716. The seventh bit from 6522 port B drives the 2716 chip enable via a 74LS04 inverter. The output of the 6522's interval timer is available at pin 39 (CA2), and this output is coupled to the 2716 output enable. Having exhausted that 6522's supply inputs and outputs, the four most significant address bits for the 2716 are provided via the 74LS175 4-bit-latch. Programming voltage is controlled with a toggle switch.

It is important that the programming voltage stays between 24,5 and 25.5 volts. This may be accomplished bythe use of an external position supply, or by using several batteries as described below.

Connection of three 9-volt-batteries in series yield the voltage of 27 to 28 volts. This is in excess of the 25.5 volt maximum required by the EPROM programming board. The necessary 2 to 3 volt drop may be attained using several 1-end 914 diodes in series. Each diode will exhibit a voltage drop of approximately 0.6 volts. Connection of four diodes connected in series should adjust the programming voltage to the desired 25 volt level.

Operation of the EPROM programmer

Insert the EPROM programming board into slot 1 of the ELCOMP -1 expansion card. Make sure that the + 5 volt switch and the programming voltage switch are both in the "off"-position (down). Apply power to the ELCOMP-1 expansion board and to the Superboard. Load the EPROM programming software. Next, load the desired object code into computer memory beginning at location 1000 hex to location 17FF hex. (4000 to 47FF if you use the board with the APPLE II).Insert the blank EPROMinto the socket on the EPROM programming card. Before proceeding it is necessary to tell the EPROM programming software the amount of EPROMs to be programmed, and the area of RAM where the desired information is currently stored. The computer systems machine language monitor should be used to place this information in memory as shown in the table below.

Location Hex	Address	
0010	Starting Address EPROM	LSB
0011	Starting Address EPROM	MSB
0012	Ending Address EPROM	LSB
0013	Ending Address EPROM	MSB
0014	Starting Address in RAM	LSB
0015	Starting Address in RAM	MSB
0016	Ending Address in RAM	LSB
0017	Ending Address in RAM	MSB

Note that the EPROM address is not specified as an absolute machine address, but rather as the offset into the EPROM. Thus the EPROM address will be a number between 0 and 7FF hex. The address in RAM where the object code is stored must be entered as an absolute address and may be anywhere in available RAM memory.

The user need be concerned with the above address assignments only if he intends to program a portion of the EPROM, rather than the entire EPROM. If EPROM programming software is executed beginning with location 0800 hex , then the software assumes that the entire EPROM is to be programmed (or read or tested) and that the RAM image of the EPROM will be present at locations 1000 to 17FF hex for the Superboard (4000 to 47FF for the APPLE II).

If only one section of an EPROM is to be programmed, read, or tested, then the EPROM programming software should be executed beginning with location 803 hex. In this case the EPROM programming software will give its attention to that portion of the EPROM indicated by the contents of the locations mentioned in the table above. Specific details of programming EPROMs, reading EPROMs, and testing EPROMs are described below.

Assembly instructions of the EPROM-Burner

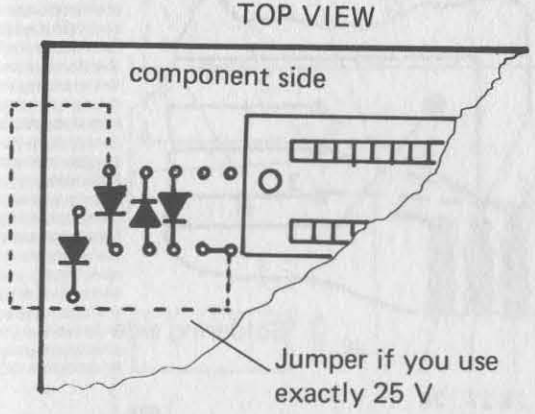
It is recommended to install all integrated circuits in DIL-sockets. To put the jumpers at the right place is a little bit tricky. There is a total of 7 jumperwires on the board.
First install the two on the componentside on the board (J1 + J2). Then install the five jumpers on the soldering side of the board. J3 connects pin 20 and pin 24 of the 6522 VIA chip. Place J4, J5, and J6 on the soldering side of the board. The matching wholes are provided on the right hand side of the connector.

Another connection has to be made to connect the clock to the 6522 chip. See jumper J7 in the assembly schematic and J1 on the soldering side of the board.
The power supply for the burner voltage is supplied via two soldering points near switch S1 (–+).

Installation of the diodes

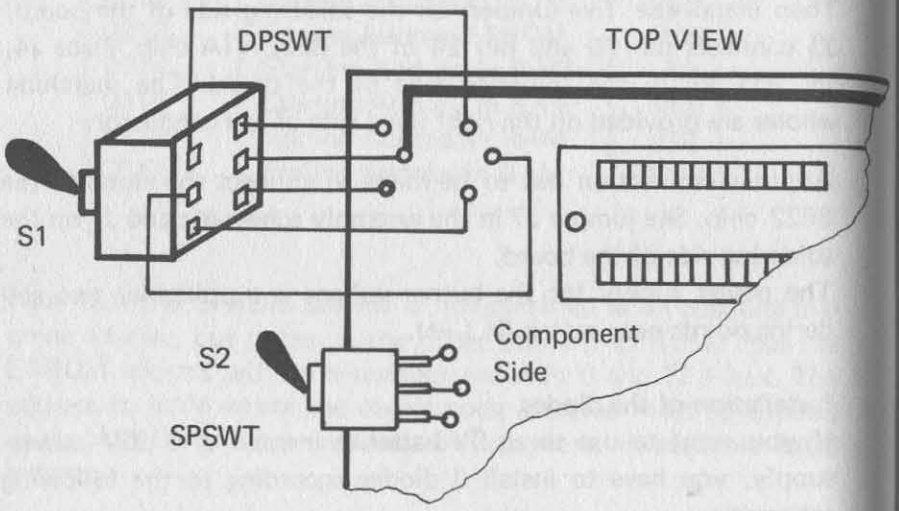
If you want to use three 9V-batteries instead of a 25V power-supply, you have to install 3 diodes according to the following schematic.

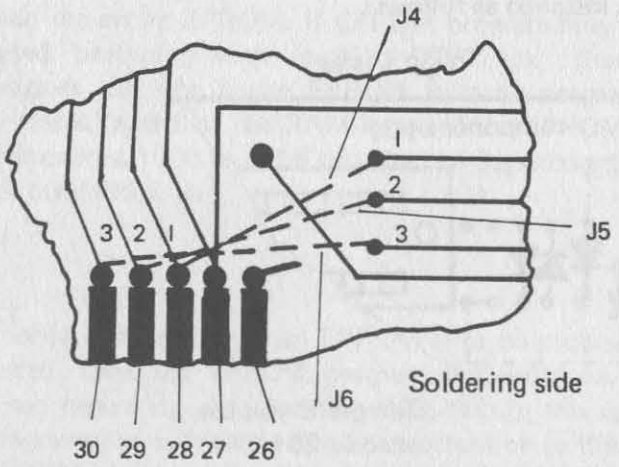
To reduce the programming voltage from 3 x 9V = 27 V to 25V, use three diodes installed as follows:

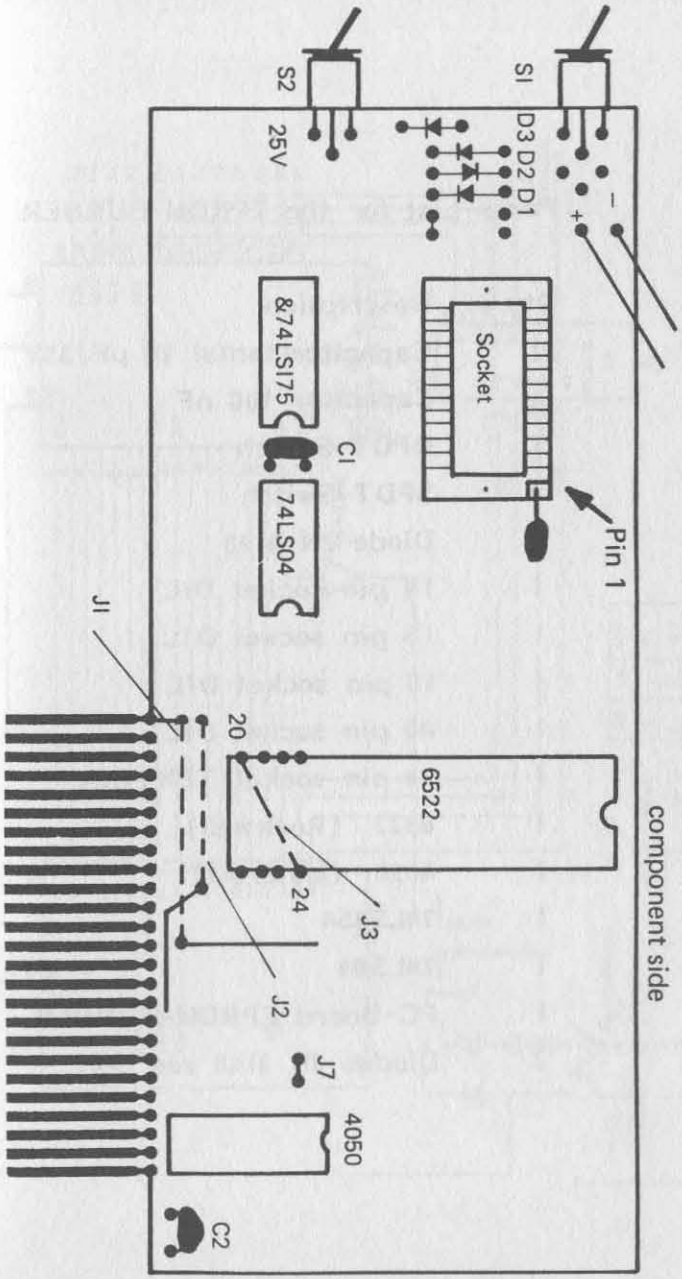


If you use a 25V power supply, wire a jumper from the upmost left soldering point to downmost right point.

The switches S1 and S2 must be mounted using a solid copperwire to give a fasten position. For details see the figure below.







View from the Component side

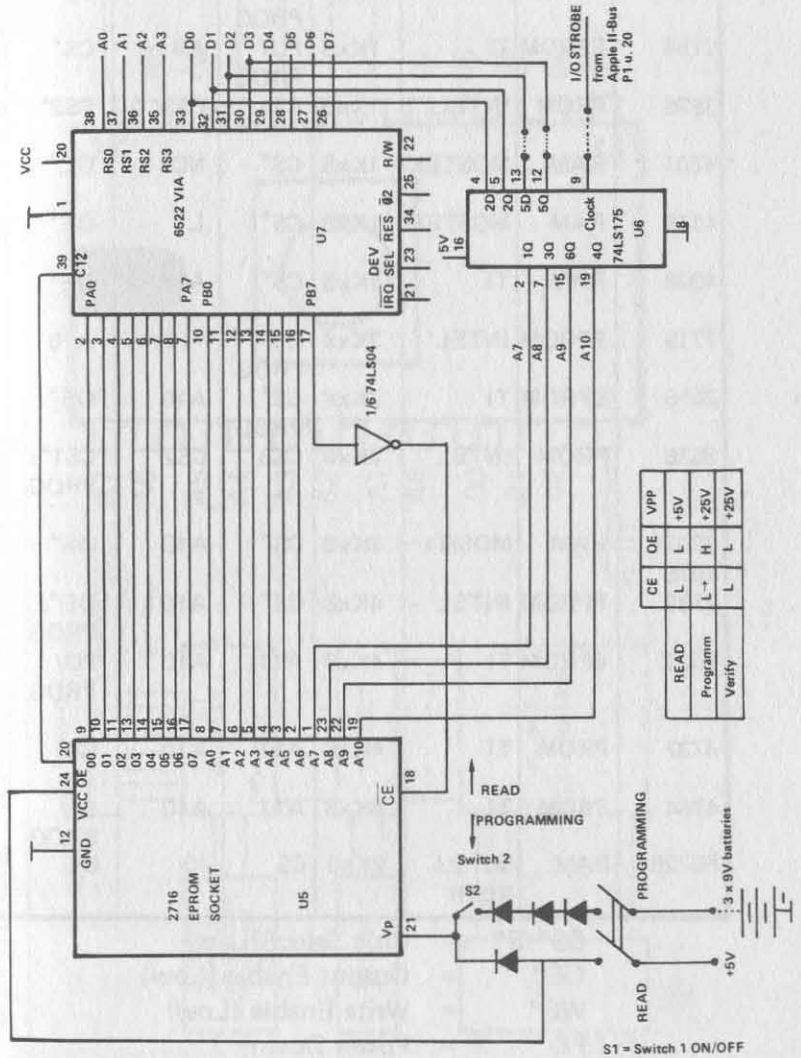
EPROM BURNER Parts Layout

component side

Parts-List for the EPROM-BURNER

Qty	Description
1	Capacitor tantal 10 µF/35V
1	Capacitor 100 nF
1	DPDT-Switch
1	SPDT-Switch
1	Diode 2N 4148
1	14 pin socket DIL
1	16 pin socket DIL
1	18 pin socket DIL
1	40 pin socket DIL
1	24 pin socket TEXTOOL
1	6522 (Rockwell)
1	4050 (Motorola)
1	74LS154
1	74LS04
1	PC-Board EPROM-BURNER
3	Diodes 2N 4148 see text

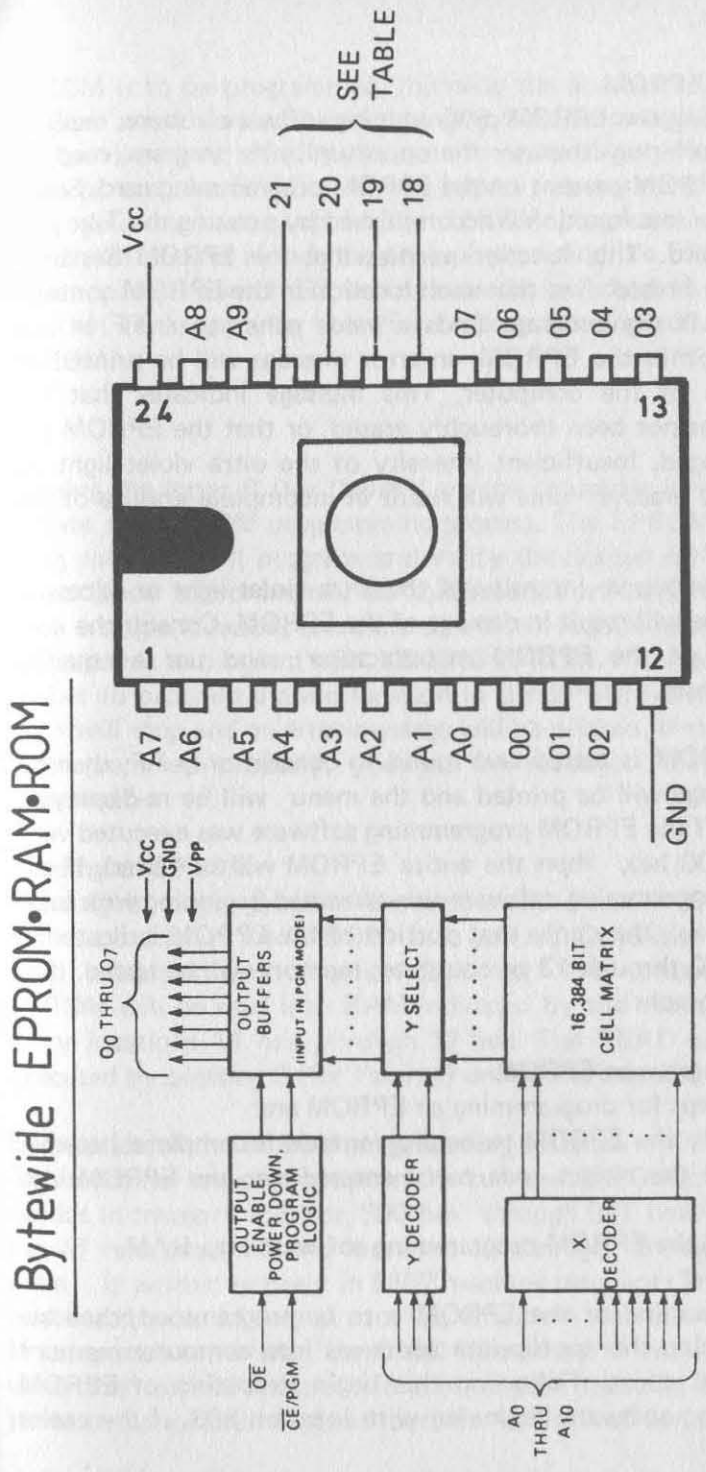
EPROM-BURNER BOARD



24 PIN MEMORY EPROMS & RAMS

DEVICE	TYPE	MANUF	SIZE	PIN 18	PIN 19	PIN 20	PIN 21
2708	EPROM	TI	1Kx8	PROG	+12V	CS*(PE)	–5V
2508	EPROM	TI	1Kx8	PC/ PROG	NC	CS*	Vpp
2758	EPROM	TI	1Kx8	PD/ PROG	AR	CS*	Vpp
3628	PROM	INTEL	1Kx8	CS4	CS3	CS2*	CS1*/ PROG
4801	RAM	MOSTEK	1Kx8	CS*	NC	OE*	WE*
4118	RAM	MOSTEK	1Kx8	CS*	L*	OE*	WE*
4008	RAM	TI	1Kx8	CS*	AR	OE*	WE*
2716	EPROM	INTEL	2Kx8	CS*/ PROG	+12V	A10	–5V
2516	EPROM	TI	2Kx8	CS*/ PROG	A10	OE*	Vpp
3636	PROM	INTEL	2Kx8	CS3	CS2	CS1*/ PROG	A10
4802/ 4016	RAM	MOS/TI	2Kx8	CS*	A10	OE*	WE*
2732	EPROM	INTEL	4Kx8	CS*	A10	OE*/ PROG	A11
2532	EPROM	TI	4Kx8	A11	A10	PD/ PROG	Vpp
4732	PROM	TI	4Kx8	A11	A10	CS	CS*/ PROG
4764	PROM	TI	8Kx8	A11	A10	S*/ PROG	A12
58725	RAM	MITSU-BISHI	2Kx8	CS	10	OE	WE

CS*/S* = Chip Select (Low)
OE* = Output Enable (Low)
WE* = Write Enable (Low)
PD = Power Down
PROG/(PE)= Program Enable
Vpp = 25V (Program Voltage)
L* = LATCH (LOW)
AR = Array: If True Output VI0
If False Output VI1



Testing an EPROM

After starting the EPROM programming software a menu will be displayed offering the user the opportunity to program, read, or test the EPROM present on the EPROM programming card. Selection of the test function is accomplished by pressing the T-key on the keyboard. This function verifies that the EPROM has been completely erased (i.e. that each location in the EPROM contains FF hex.). If the program finds a value other than FF in any location within the EPROM, an error message will be printed on the screen of the computer. This message indicates that the EPROM has not been thoroughly erased, or that the EPROM has been damaged. Insufficient intensity of the ultra violet light or insufficient erasure time will result in incomplete erasure of the EPROM.

However, excessive intensity of the ultra violet light or excessive erasure time will result in damage of the EPROM. Consult the specifications of the EPROM manufacturer and use a quality EPROM eraser.

If the EPROM is tested and found to contain only FF, then no error message will be printed and the menu will be re-displayed. Note that if the EPROM programming software was executed with location 800 hex, then the entire EPROM will be tested. If the EPROM programming software was executed beginning with location 803 hex , then only that portion of the EPROM indicated in locations 10 through 13 of computer memory will be tested. (See above paragraphs).

How to program an EPROM

The first steps for programming an EPROM are:

1. Verify the EPROM to be programmed is completely erased.
2. Load the object code to be entered into the EPROM into RAM.
3. Load the EPROM programming software into RAM.

If only a portion of the EPROM is to be programmed, then begin by placing the appropriate addresses into computer memory as indicated above. Following this begin execution of EPROM programming software beginning with location 803. If the entire

EPROM is to be programmed, then skip the above step and begin program execution with location 800 hex. Once again a menu will be displayed offering the user the opportunity to program, read, or test the EPROM. If you intend to program the EPROM then you must now apply the EPROM programming voltage. Switch S1 should already be on (up) providing the EPROM with 5-volt power. Switch S2 should now be switched to the "up"-position (on), applying the 25-volt programming voltage to the EPROM.

Pressing the letter B (for "burn") on the computer keyboard will initiate the EPROM programming process. The EPROM programming software will program and verify the desired EPROM locations. Since approximately 50 milliseconds are required to program a single location, it is clear that it will require approximately 100 seconds to program a 2K X8 2716 EPROM. If the computer is unable to program a given location in the EPROM, then the program will stop and an error-message will be printed. If the EPROM is programmed properly, you get the message: "EPROM programmed".

Reading an EPROM

Once again begin program execution with location 803 hex if only a portion of the EPROM is to be read. Otherwise begin program execution with location 800 hex. The contents of the EPROM will be read into RAM indicated by the contents of memory locations 14 hex. through 17 hex. The READ operation is executed by pressing S (for "store") on the keyboard.

Description of the EPROM programming software

The machine language program to run the EPROM programmer resides in memory location 800 hex through 9FF hex. It requires use of zero page memory locations 10 through 1B hex. The program is written entirely in 6502 machine language. The program requires the use of several routines which are machine dependent. If computers other than the OHIO Scientific C1P or APPLE II are used, then several locations in the program must be changed to point at the location of these routines within your computer.

Machine dependant routine No. 1
Output an ASCII character from the accumulator. This routine is called by the EPROM programming software at location 083E hex. The location of this routine in several computers is noted below.

1. Output of an ASCII character from the accumulator onto the screen

APPLE	=	FDED
OHIO C1P	=	BF2D
ATARI	=	F6A4 (May be changed by ATARI later)
AIM	=	E97A
PET	=	FFD2

This routine is called at memory location 083E hex.

Machine dependent routine No. 2
Input an ASCII character from the keyboard into the accumulator. This routine is called by the EPROM programming software at location 080B hex. The location of this routine in several popular computers is noted below.

2. INPUT an ASCII character from keyboard into the accumulator

APPLE	=	FD35
OHIO C1P (Superboard)	=	FD00
ATARI	=	F6DD (may be changed by ATARI)
AIM	=	E93C
PET	=	FF4E

This routine is called at memory location 080B hex.

Machine dependent routine No. 3
Jump back to the monitor. This routine is called by the EPROM programming software at several points. These points are locations 081E, 0828, and 0938 hex. The location of this routine in several popular computers is noted below.

3. Jump back to the monitor

APPLE II	=	FF59 hex
OHIO Superboard C1P	=	0000 hex (Restart monitor FE00)
ATARI	=	difficult
AIM	=	E07B
PET	=	0400 TIM-Monitor

A call to this routine is made in locations
081E, 0828 and 0938

Source-Listing for EPROM-Burner

```
0800              1          DCM "PR#1"
C0C0              2          ORG $C0C0
C0C0              3  TORB    EQU *
C0C0              4  TORA    EQU *+!1
C0C0              5  DDRB    EQU *+!2
C0C0              6  DDRA    EQU *+!3
C0C0              7  T1CL    EQU *+!4
C0C0              8  T1CH    EQU *+!5
C0C0              9  ACR     EQU *+!11
C0C0             10  PCR     EQU *+!12
C0C0             11  IFR     EQU *+!13
C0C0             12  ;
C0C0             13  ;
C0C0             14  STR     EQU $C800
C0C0             15  COUT    EQU $BF2D
C0C0             16  RDCHAR  EQU $FD00
C0C0             17  MONITO  EQU $FE00
C0C0             18  ;
C0C0             19  SAEPL   EPZ $10
C0C0             20  SAEPH   EPZ SAEPL+!1
C0C0             21  EAEPL   EPZ SAEPL+!2
C0C0             22  EAEPH   EPZ SAEPL+!3
C0C0             23  SAPL    EPZ SAEPL+!4
C0C0             24  SAPH    EPZ SAEPL+!5
C0C0             25  EAPL    EPZ SAEPL+!6
C0C0             26  EAPH    EPZ SAEPL+!7
```

```
C0C0            27  LAL     EPZ SAEPL+!8
C0C0            28  LAH     EPZ SAEPL+!9
C0C0            29  LACL    EPZ SAEPL+!10
C0C0            30  LACH    EPZ SAEPL+!11
C0C0            31  HFZ     EPZ SAEPL+!12
C0C0            32  ;
C0C0            33  ;
0800            34          ORG $800
0800 20C208     35  CSTART  JSR DEFAU
0803 201509     36  WSTART  JSR INIT
0806 A246       37          LDX #70
0808 203B08     38          JSR TXTOUT
080B 2000FD     39          JSR RDCHAR
080E 8D4D08     40          STA SAVEC
0811 202DBF     41          JSR COUT
0814 AD4D08     42          LDA SAVEC
0817 C9D2       43          CMP #"R"
0819 D006       44          BNE L1
081B 205809     45          JSR LESEN
081E 4C00FE     46          JMP MONITO
0821 C9C2       47  L1      CMP #"B"
0823 D006       48          BNE L2
0825 20C809     49          JSR PROGRA
0828 4C00FE     50          JMP MONITO
082B C9D4       51  L2      CMP #"T"
082D D0D4       52          BNE WSTART
082F 207409     53          JSR PRUEFE
0832 A265       54          LDX #101
0834 203B08     55          JSR TXTOUT
0837 4C00FE     56          JMP MONITO
083A 00         57          BRK
083B            58  ;
083B            59  ;
083B 8D4B08     60  TXTOUT  STA SAVEA
083E BD4E08     61  TXT1    LDA TEXT,X
0841 F007       62          BEQ FIN
0843 202DBF     63          JSR COUT
0846 E8         64          INX
0847 18         65          CLC
0848 90F4       66          BCC TXT1
084A 60         67  FIN     RTS
084B 0000       68  SAVEA   HEX 0000
084D 00         69  SAVEC   HEX 00
```



```
084E            70  ;
084E            71  ;
084E 8D8D       72  TEXT    HEX 8D8D
0850 C5D0D2     73          ASC "EPROM NOT EREASED    "
0853 CFCDA0
0856 CECFD4
0859 A0C5D2
085C C5C1D3
085F C5C4A0
0862 A0A0A0
0865 008D       74          HEX 008D
0867 C5D0D2     75          ASC "EPROM NOT PROGRAMMED    "
086A CFCDA0
086D CECFD4
0870 A0D0D2
0873 CFC7D2
0876 C1CDCD
0879 C5C4A0
087C A0A0A0
087F 008D       76          HEX 008D
0881 C5D0D2     77          ASC "EPROM PROGRAMMED  "
0884 CFCDA0
0887 D0D2CF
088A C7D2C1
088D CDCDC5
0890 C4A0A0
0893 008D       78          HEX 008D
0895 C2A9D5     79          ASC "B)URNING T)ESTING R)EADING   "
0898 D2CEC9
089B CEC7A0
089E D4A9C5
08A1 D3D4C9
08A4 CEC7A0
08A7 D2A9C5
08AA C1C4C9
08AD CEC7A0
08B0 A0A0
08B2 00         80          HEX 00
08B3 8D         81          HEX 8D
08B4 C5D0D2     82          ASC "EPROM EREASED"
08B7 CFCDA0
08BA C5D2C5
08BD C1D3C5
08C0 C4
08C1 00         83          HEX 00
08C2            84  ;
08C2            85  ;
08C2            86  ;
08C2 A900       87  DEFAU   LDA #$00
08C4 8510       88          STA SAEPL
08C6 8511       89          STA SAEPH
08C8 8514       90          STA SAPL
08CA A9FF       91          LDA #$FF
08CC 8516       92          STA EAPL
08CE 8512       93          STA EAEPL
08D0 A907       94          LDA #$07
08D2 8513       95          STA EAEPH
08D4 A940       96          LDA #$40
08D6 8515       97          STA SAPH
```

```
08D8 A947     98          LDA #$47
08DA 8517     99          STA EAPH
08DC 60      100          RTS
08DD         101  ;
08DD         102  ;
08DD A518    103  EOUT    LDA LAL
08DF 8DC0C0  104          STA TORB
08E2 851A    105          STA LACL
08E4 A519    106          LDA LAH
08E6 851B    107          STA LACH
08E8 261A    108          ROL LACL
08EA 261B    109          ROL LACH
08EC A51B    110          LDA LACH
08EE 8D00C8  111          STA STR
08F1 18      112          CLC
08F2 60      113          RTS
08F3         114  ;
08F3         115  ;
08F3 E618    116  NEXT    INC LAL
08F5 D002    117          BNE N1
08F7 E619    118          INC LAH
08F9 E610    119  N1      INC SAEPL
08FB D002    120          BNE N2
08FD E611    121          INC SAEPH
08FF A511    122  N2      LDA SAEPH
0901 C513    123          CMP EAEPH
0903 900C    124          BCC N3
0905 F002    125          BEQ N4
0907 B00B    126          BCS N5
0909 A510    127  N4      LDA SAEPL
090B C512    128          CMP EAEPL
090D F002    129          BEQ N3
090F B003    130          BCS N5
0911 20DD08  131  N3      JSR EOUT
0914 60      132  N5      RTS
0915         133  ;
0915         134  ;
0915 A900    135  INIT    LDA #$00
0917 8DC3C0  136          STA DDRA
091A AA      137          TAX
091B A8      138          TAY
091C A97F    139          LDA #$7F
091E 8DC2C0  140          STA DDRB
```



```
0921 A980    141          LDA #$80
0923 8DCBC0  142          STA ACR     ;PB7 MONOFLOP
0926 60      143          RTS
0927         144  ;
0927         145  ;
0927 A510    146  START   LDA SAEPL
0929 8518    147          STA LAL
092B 851A    148          STA LACL
092D 8DC0C0  149          STA TORB
0930 A511    150          LDA SAEPH
0932 8519    151          STA LAH
0934 851B    152          STA LACH
0936 261A    153          ROL LACL
0938 261B    154          ROL LACH
093A A51B    155          LDA LACH
093C 8D00C8  156          STA STR
093F 60      157          RTS
0940         158  ;
0940         159  ;
0940 C901    160  ERROR   CMP #$01
0942 D008    161          BNE E1
0944 A200    162          LDX #$00
0946 203B08  163          JSR TXTOUT
0949 4C00FE  164          JMP MONITO
094C C902    165  E1      CMP #$02
094E D005    166          BNE E2
0950 A218    167          LDX #24
0952 203B08  168          JSR TXTOUT
0955 4C00FE  169  E2      JMP MONITO
0958         170  ;
0958 201509  171  LESEN   JSR INIT
095B A90C    172          LDA #$0C
095D 8DCCC0  173          STA PCR     ;OE=L READ
0960 202709  174          JSR START
0963 ADC1C0  175  LES1    LDA TORA
0966 9114    176          STA (SAPL),Y
0968 E614    177          INC SAPL
096A D002    178          BNE LES2
096C E615    179          INC SAPH
096E 20F308  180  LES2    JSR NEXT
0971 90F0    181          BCC LES1
0973 60      182          RTS
0974         183  ;
```

```
0974 201509   184  PRUEFE JSR INIT
0977 A90C     185         LDA #$0C
0979 8DCCC0   186         STA PCR   ;OE=L  READ
097C 202709   187         JSR START
097F ADC1C0   188  P1      LDA TORA
0982 C9FF     189         CMP #$FF
0984 F005     190         BEQ P2
0986 A901     191         LDA #$01
0988 4C4009   192         JMP ERROR
098B 20F308   193  P2      JSR NEXT
098E 90EF     194         BCC P1
0990 60       195         RTS
0991          196  ;
0991 A90E     197  MONOFL  LDA #$0E
0993 8DCCC0   198         STA PCR   ;OE=H PROGRA
0996 A950     199         LDA #$50
0998 8DC4C0   200         STA T1CL
099B A9C3     201         LDA #$C3
099D 8DC5C0   202         STA T1CH
09A0 ADCDC0   203  M01     LDA IFR
09A3 2940     204         AND #$40
09A5 F0F9     205         BEQ M01
09A7 A90C     206         LDA #$0C
09A9 8DCCC0   207         STA PCR   ;OE=L
09AC 60       208         RTS
09AD          209  ;
09AD          210  ;
09AD A200     211  CHANGE  LDX #$00
09AF A004     212         LDY #$04
09B1 B510     213  CA1     LDA $0010,X
09B3 851C     214         STA HFZ
09B5 B91000   215         LDA $0010,Y
09B8 9510     216         STA $0010,X
09BA A51C     217         LDA HFZ
09BC 991000   218         STA $0010,Y
09BF C8       219         INY
09C0 E8       220         INX
09C1 E004     221         CPX #$04
09C3 D0EC     222         BNE CA1
09C5 A000     223         LDY #$00
09C7 60       224         RTS
09C8          225  ;
09C8          226  ;
```



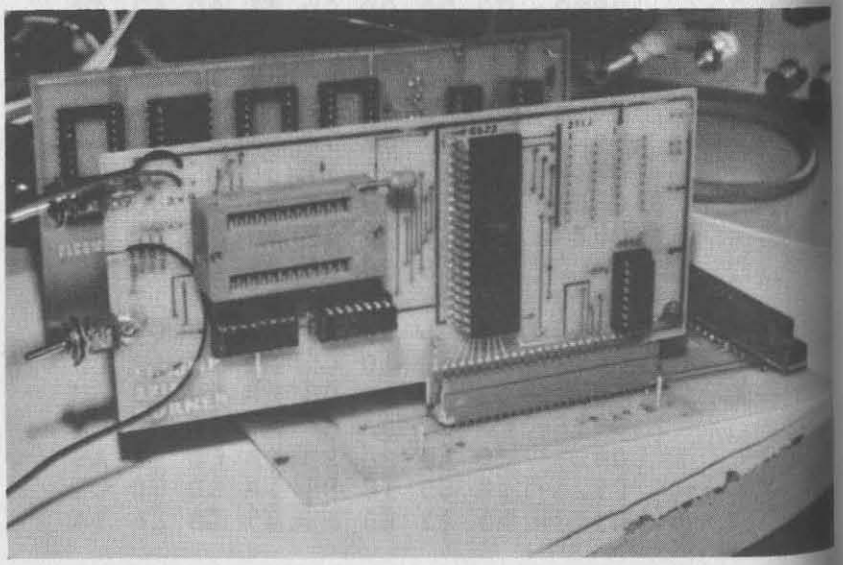
```
09C8 202709   227  PROGRA JSR START
09CB 20AD09   228         JSR CHANGE
09CE A9FF     229  PR1     LDA #$FF
09D0 8DC3C0   230         STA DDRA
09D3 B110     231         LDA (SAEPL),Y
09D5 8DC1C0   232         STA TORA
09D8 AA       233         TAX
09D9 209109   234         JSR MONOFL
09DC A900     235         LDA #$00
09DE 8DC3C0   236         STA DDRA
09E1 8A       237         TXA
09E2 CDC1C0   238         CMP TORA
09E5 F00B     239         BEQ PR3
09E7 A902     240         LDA #$02
09E9 4C4009   241         JMP ERROR
09EC E610     242  PR2     INC SAEPL
09EE D002     243         BNE PR3
09F0 E611     244         INC SAEPH
09F2 20F308   245  PR3     JSR NEXT
09F5 90D7     246         BCC PR1
09F7 A232     247         LDX #50
09F9 4C3B08   248         JMP TXTOUT
09FC 60       249         RTS
09FD          250  ;
              251         END

***** END OF ASSEMBLY
```

```
          **************************
          *                        *
          * SYMBOL TABLE -- V 1.5  *
          *                        *
          **************************

LABEL. LOC.  LABEL. LOC.  LABEL. LOC.

** ZERO PAGE VARIABLES:

SAEPL  0010  SAEPH  0011  EAEPL  0012  EAEPH  0013  SAPL   0014  SAPH   0015
EAPL   0016  EAPH   0017  LAL    0018  LAH    0019  LACL   001A  LACH   001B
HFZ    001C

** ABSOLUTE VARABLES/LABELS

TORB   C0C0  TORA   C0C1  DDRB   C0C2  DDRA   C0C3  T1CL   C0C4
T1CH   C0C5  ACR    C0CB  PCR    C0CC  IFR    C0CD  STR    C800  COUT   BF2D
RDCHAR FD00  MONITO FE00  CSTART 0800  WSTART 0803  L1     0821  L2     082B
TXTOUT 083B  TXT1   083E  FIN    084A  SAVEA  084B  SAVEC  084D  TEXT   084E
DEFAU  08C2  EOUT   08DD  NEXT   08F3  N1     08F9  N2     08FF  N4     0909
N3     0911  N5     0914  INIT   0915  START  0927  ERROR  0940  E1     094C
E2     0955  LESEN  0958  LES1   0963  LES2   096E  PRUEFE 0974  P1     097F
P2     098B  MONOFL 0991  MO1    09A0  CHANGE 09AD  CA1    09B1  PROGRA 09C8
PR1    09CE  PR2    09EC  PR3    09F2

SYMBOL TABLE STARTING ADDRESS:6000
SYMBOL TABLE LENGTH:020A
```


800.9FC

```
0800- 20 C2 08 20 15 09 A2 46
0808- 20 3B 08 20 00 FD 8D 4D
0810- 08 20 2D BF AD 4D 08 C9
0818- 52 D0 06 20 58 09 4C 00
0820- FE C9 42 D0 06 20 C8 09
0828- 4C 00 FE C9 54 D0 D4 20
0830- 74 09 A2 65 20 3B 08 4C
0838- 00 FE 00 8D 4B 08 BD 4E
0840- 08 F0 07 20 2D BF E8 18
0848- 90 F4 60 00 00 00 0D 0D
0850- 45 50 52 4F 4D 20 4E 4F
0858- 54 20 45 52 45 41 53 45
0860- 44 20 20 20 20 0D 45 0D
0868- 50 52 4F 4D 20 4E 4F 54
0870- 20 50 52 4F 47 52 41 4D
0878- 4D 45 44 20 20 20 00 00
0880- 0D 45 50 52 4F 4D 20 50
0888- 52 4F 47 52 41 4D 45 45
0890- 44 20 20 00 0D 42 29 55
0898- 52 4E 49 4E 47 20 54 29
08A0- 45 53 54 49 4E 47 20 52
08A8- 29 45 41 44 49 4E 47 20
08B0- 20 20 00 0D 45 50 52 4F
08B8- 4D 20 45 52 45 41 53 45
08C0- 44 00 A9 00 85 10 85 11
08C8- 85 14 A9 FF 85 16 85 12
08D0- A9 07 85 13 A9 40 85 15
08D8- A9 47 85 17 60 A5 18 8D
08E0- C0 C0 85 1A A5 19 85 1B
08E8- 26 1A 26 1B A5 1B 8D 00
08F0- C8 18 60 E6 18 D0 02 E6
08F8- 19 E6 10 D0 02 E6 11 A5
0900- 11 C5 13 90 0C F0 02 B0
0908- 0B A5 10 C5 12 F0 02 B0
0910- 03 20 DD 08 60 A9 00 8D
0918- C3 C0 AA A8 A9 7F 8D C2
0920- C0 A9 80 8D CB C0 60 A5
0928- 10 85 18 85 1A 8D C0 C0
0930- A5 11 85 19 85 1B 26 1A
```

```
0938- 26 1B A5 1B 8D 00 C8 60
0940- C9 01 D0 08 A2 00 20 3B
0948- 08 4C 00 FE C9 02 D0 05
0950- A2 18 20 3B 08 4C 00 FE
0958- 20 15 09 A9 0C 8D CC C0
0960- 20 27 09 AD C1 C0 91 14
0968- E6 14 D0 02 E6 15 20 F3
0970- 08 90 F0 60 20 15 09 A9
0978- 0C 8D CC C0 20 27 09 AD
0980- C1 C0 C9 FF F0 05 A9 01
0988- 4C 40 09 20 F3 08 90 EF
0990- 60 A9 0E 8D CC C0 A9 50
0998- 8D C4 C0 A9 C3 8D C5 C0
09A0- AD CD C0 29 40 F0 F9 A9
09A8- 0C 8D CC C0 60 A2 00 A0
09B0- 04 B5 10 85 1C B9 10 00
09B8- 95 10 A5 1C 99 10 00 C8
09C0- E8 E0 04 D0 EC A0 00 60
09C8- 20 27 09 20 AD 09 A9 FF
09D0- 8D C3 C0 B1 10 8D C1 C0
09D8- AA 20 91 09 A9 00 8D C3
09E0- C0 8A CD C1 C0 F0 0B A9
09E8- 02 4C 40 09 E6 10 D0 02
09F0- E6 11 20 F3 08 90 D7 A2
09F8- 32 4C 3B 08 60
*
```



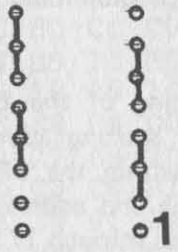
Summary of operating instruction

1. Insert the EPROM programming card to slot 1 of the ELCOMP-1 mother board with power "off".
2. Apply power to your computer.
3. Connect programming voltage to the EPROM programming card.
4. Make sure that both switches on the EPROM programming card are down ("off").
5. Insert the blank EPROM into the socket.
6. Read the object program into RAM (the program you wish to place in the EPROM).
7. Load the EPROM programming software from cassette tape.
8. Switch S1 up ("on").
9. If partial programming of the EPROM is desired then you must insert the starting address within the EPROM, the ending address within the EPROM, the start address within RAM, and the end address within RAM, into memory locations 10 hex through 17 hex.
10. For partial EPROM programming, begin execution of the EPROM programming software beginning with location 0803 hex., otherwise begin program execution with locations 0800 hex.
11. For saving or testing the EPROM, leave switch S2 in down-position and select the save or test function by pressing key S or key T respectively.
12. To program the EPROM move switch S2 to the up-position ("on"). Then press B to burn the EPROM.
13. After programming is complete return switch S2 to the down ("off") position.
14. Return switch S1 to the down ("off") position and remove EPROM from the socket.

Using the 2716 EPROM Burner without the motherboard

If you want to use the burner with your C1P or Superboard without the motherboard you have to do your own decoding for the burner board. Therefore you can connect a decoding circuit (built on a small prototyping board) to the 40pin expansion connector of your Superboard. The other end of the decoding circuit you can connect with a 50pin APPLE II® bus connector.

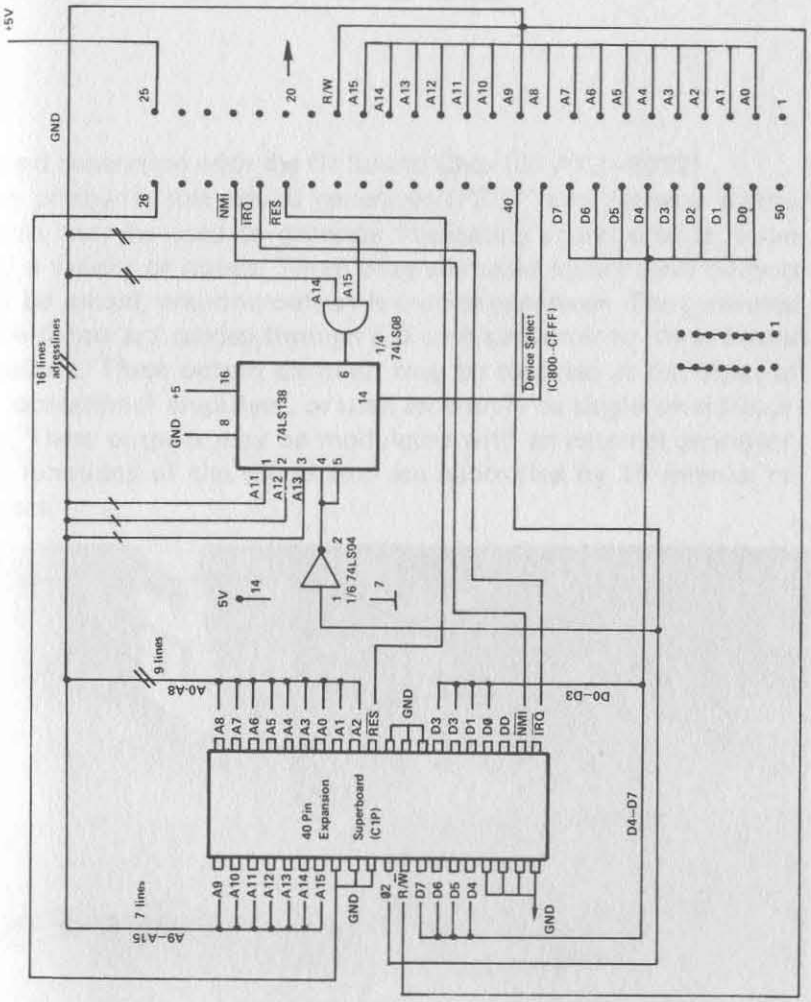
Of course you need a +5V powersupply for the board and 9 x 3 V to burn the EPROMs.
You also have to remove the 8T28 chips (U6 + U7) and span the corresponding data lines.



The $\overline{\text{RES}}$ signal has to be brought to pin 11 of the 40 pin Superboard expansion connector. For details see the following schematic.

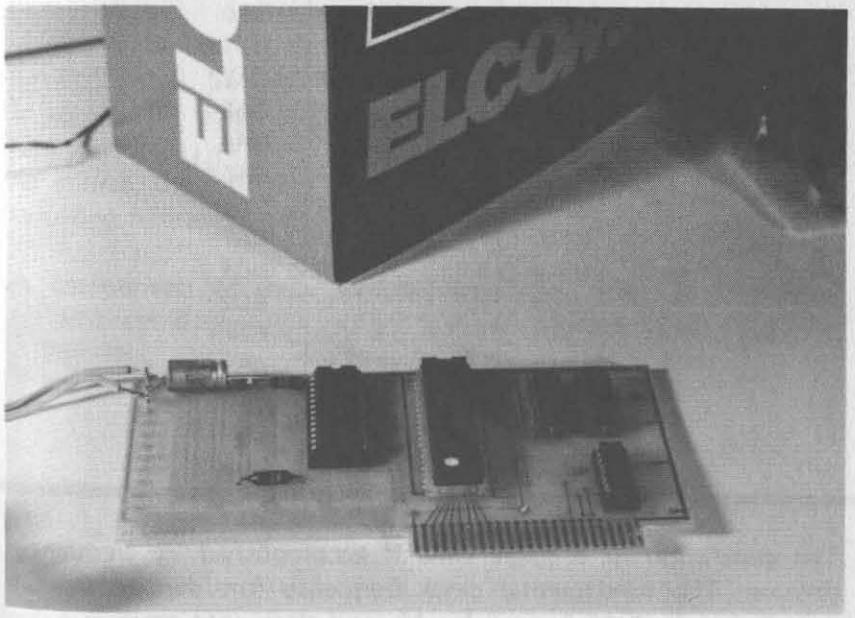
*APPLE is a trademark of APPLE Computer Inc., Cupertino CA

EPROM-BURNER for Superboard without the expansion



Sound generation with AY – 3 – 8912

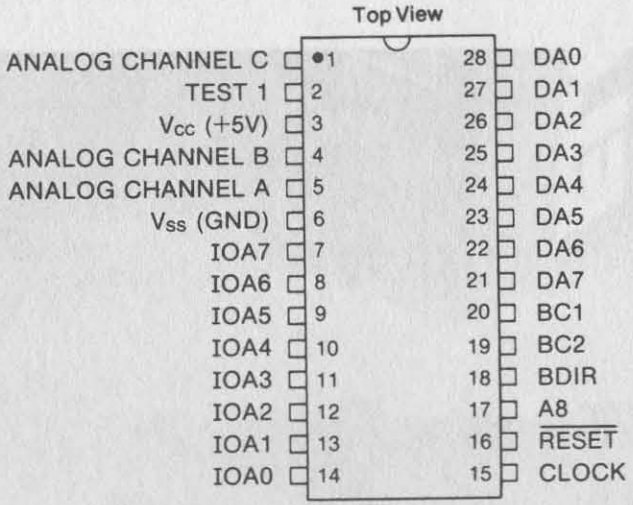
Sound generation with the GI Sound Chip (GI AY3–8912)
The programmable sound generator (PSG) from General Instruments may be used to generate fascinating sound effects, tones and a variety of noises. Three programmable square wave outputs can be mixed, and one output is a noise generator. The generated wave forms are carried through a D to A converter to three output channels. These output channels may be summed at the input of an operational amplifyer, or used separately as single sound sources. These outputs may be modulated with an external generator. All functions of the sound chip are controlled by 16 internal registers.



REGISTER		BIT B7	B6	B5	B4	B3	B2	B1	B0
R0	Channel A Tone Period	8-BIT Fine Tune A							
R1		4-BIT Coarse Tune A							
R2	Channel B Tone Period	8-BIT Fine Tune B							
R3		4-BIT Coarse Tune B							
R4	Channel C Tone Period	8-BIT Fine Tune C							
R5		4-BIT Coarse Tune C							
R6	Noise Period	5-BIT Period Control							
R7	Enable	IN/OUT		Noise			Tone		
		IOB	IOA	C	B	A	C	B	A
R10	Channel A Amplitude				M	L3	L2	L1	L0
R11	Channel B Amplitude				M	L3	L2	L1	L0
R12	Channel C Amplitude				M	L3	L2	L1	L0
R13	Envelope Period	8-BIT Fine Tune E							
R14		8-BIT Coarse Tune E							
R15	Envelope Shape/Cycle					CONT	ATT	ALT	HOLD
R16	I/O Port A Data Store	8-BIT PARALLEL I/O on Port A							
R17	I/O Port B Data Store	8-BIT PARALLEL I/O Port B							

The 16 internal registers

AY-3-8912 PIN ASSIGNMENTS



The generation of a single tone is accomplished by frequency division. The fundamental clock frequency furnished to the GI sound chip is divided once by 16, and then once again in a divide by 12 counter.

This 12 bit word now will be put into register 0 (8 lower bits) and the remaining four bits for channel A will be put into register 1. For a given clock frequency you can calculate the tone period as follows:

$$tp = \frac{fclock}{t*16}$$

F = the desired frequency
fclock = applied clock frequency for the chip
Both values are used in Hz.

Example: f = 440 Hz
fclock = 1 MHz

$$tp = \frac{10^6}{440*16} = 142.04$$

If you connect 142 decimal in a 12-bit binary number you will get 08E (hex). So with a content of 8E in register R0 and a zero in register R1 you will get a signal with a frequency of 440 Hz on one channel.
Of course, the rounding of 142.02 gives you an error, so that the resulting frequency will be 440,14 MHz.

A comparison of the calculated frequency and real frequency (at different clocks) you will get is shown in the following table:

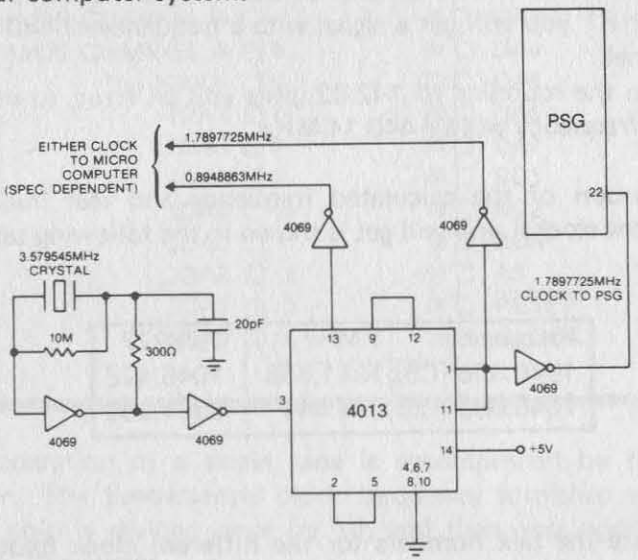
Frequency	1 MHz	1.78977
1046.496 (C6)	1041.666	1045.428
7040.000 (C8)	694.444	6991.299

To calculate the hex numbers for the different clock frequency you can use the following table:

```
100  INPUT "F= ";F
110 FC = 1000000
112  IF F < 0 THEN 180
115 FF = 16
120 TP = FC / (FF * F)
130 MSD =  INT (TP / 256)
140 TP = TR - MSD * 256
150 NSD =  INT (TP / 16)
160 LSD =  INT (TP - NSD * 16 + 0.5)
165 FI = FC / ((MSD * 256 + NSD * 16 + LSD) * FF)
170  GOSUB 200
175  GOTO 100
180  END
200  IF MSD > 9 THEN MSD = MSD + 7
210 MSD = MSD + 48:A$ =  CHR$ (MSD)
220  IF NSD > 9 THEN NSD = NSD + 7
230 NSD = NSD + 48:B$ =  CHR$ (NSD)
240  IF LSD > 9 THEN LSD = LSD + 7
250 LSD = LSD + 48:C$ =  CHR$ (LSD)
260  PRINT F;"  ";A$;B$;C$;"   ";FI
270  RETURN
```

This program calculates the contents of the registers for the PSG AY-3-8912. Clock frequency (FC) = 1 MHz.
12 bit value output in hex.
Output of actual and the must-be frequency.

The next figure shows you, how to generate a clock frequency with a 3.579545 MHz crystal.
Then the signal is divided using the CMOS chip 4013. In many applications it will be more than sufficient to use the 1 MHz clock of your computer system.



How the internal registers work
The registers R0–R5 are used to program the tone periods for the three channels A, B, and C.

The register R6 is used to program the noise generator. Therefore you only have to use the five lower bits of this registers. The lowest noise frequency can be achieved by placing 1F hex into the lower five bits (all five bits are 1). The highest possible noise frequency needs a 1 in that part of the register.

The clock frequency is now divided first by 16 and then by the 5-bit word. The noise period can be calculated with the following equation:

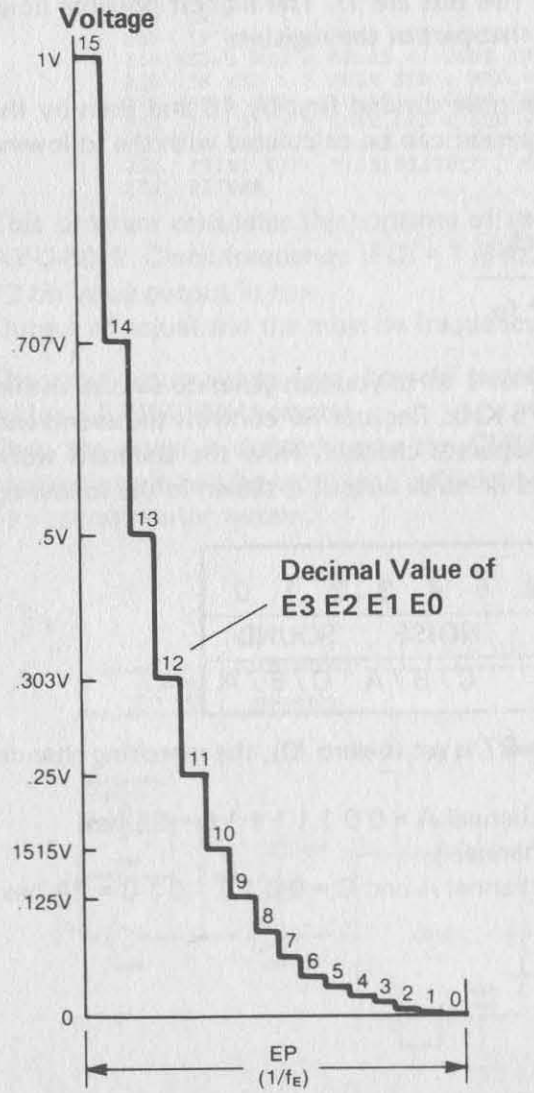
$$NP = \frac{fclock}{16 * fu}$$

With a clock frequency of 1 MHz you can generate a noise within a range from 2KHz – 75 KHz. Register R7 controls the sound and noise output of each separate channel. How the channels work together with the sound or noise output is shown in the following chart:

Bit	7	6	5	4	3	2	1	0
	I/O		NOISE			SOUND		
			C	B	A	C	B	A

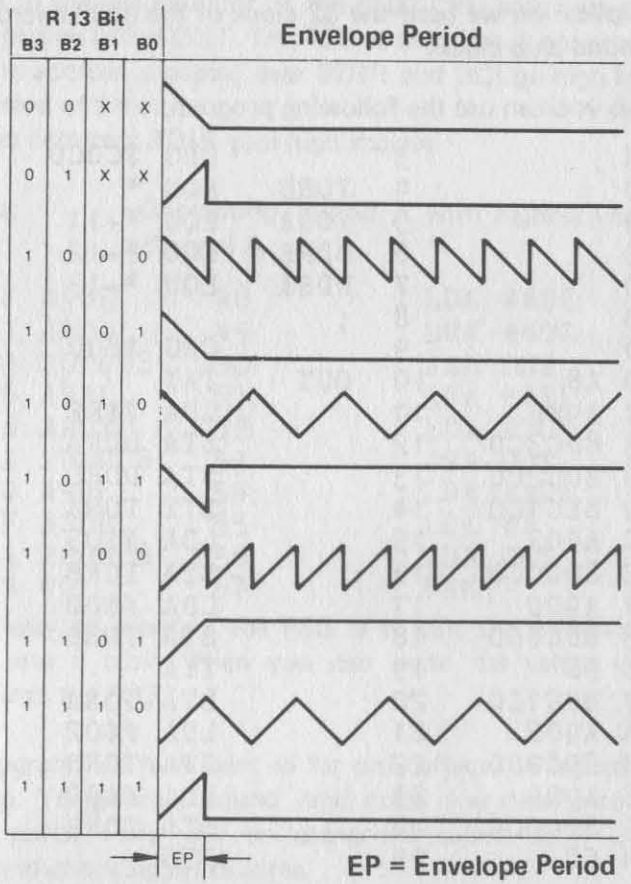
When one bit of register R7 is set to zero (0), the matching channel is open.
Example: Sound on channel A = 0 0 1 1 1 1 1 0 = 3E hex
Noise on channel B
Sound on channel A and C = 0 0 1 0 1 0 1 0 = 2A hex

The two most significant bits are used for the data transfer via the I/O port of the PSG. You don' t need them for sound generation. Registers R8, R9 und R10 are responsable for the volume of the sound outputs of each channel. In the same order they are dedicated to the channels A, B, and C.
The first four bits set the volume in 16 different levels for each channel. This setting is not linear, it is logarithmic.



If in one of these registers bit no. 5 is set to a logical 1, the amplitude of that channel is controlled by the envelope generator, which can be programed via the registers R11, R12 and R13. R11 and R10 form a 16-bit counter to generate the length of the period of the envelope.
The clockfrequency is divided by 256 and then by the content of the registers R11,and R12.R12 therefore is the least significant bit.
A 1 MHz clockfrequency allows you envelope periods of 0.06 Hz to 4000 Hz. To calculate the period you can use:

$$EP = \frac{fclock}{256 * fe} \qquad fe = fenvelope$$



The 16 bit binary value for EP is written into register R11 and R12. For that calculation you can use the program above, if you change line 111 into FF = 256.
The least significant bit of the register R13 determine the shape of the envelope (configuration).
The second waveform, with R13 = 04 generates an increasing tone with the period of EP. The volume then suddenly decreases.

How to program the GI-Soundchip
The control lines BDIR and BC2 are used to select a register. The third control line is not used; it is connected to +5V. The databus and the control lines can be controlled via a 6522 VIA (Versatile Interface Adapter). Additionally the reset and a clock is needed.

In our application we used the $\phi 2$ clock of the 6502 microprocessor for our sound chip clock.

To do this you can use the following program:

```
C0C0            3          ORG  $C0C0
C0C0            4  TORB     EQU  *
C0C0            5  TORA     EQU  *+!1
C0C0            6  DDRB     EQU  *+!2
C0C0            7  DDRA     EQU  *+!3
C0C0            8  ;
0800            9          ORG  $800
0800 A8        10  OUT      TAY
0801 A9FF      11          LDA  #$FF
0803 8DC3C0    12          STA  DDRA
0806 8DC2C0    13          STA  DDRB
0809 8EC1C0    14          STX  TORA
080C A903      15          LDA  #$03
080E 8DC0C0    16          STA  TORB
0811 A900      17          LDA  #$00
0813 8DC0C0    18          STA  TORB
0816 98        19          TYA
0817 8DC1C0    20          STA  TORA
081A A902      21          LDA  #$02
081C 8DC0C0    22          STA  TORB
081F A900      23          LDA  #$00
0821 8DC0C0    24          STA  TORB
0824 60        25          RTS
```


The data lines DA0–DA7 are connected to port A of the 6522. The control lines BCI and BDIR are hooked to PB0 and PB1. To find the data into the matching registers you first have to issue the address and then the data over the data lines.
It is controlled via the data lines BDIR and BC1.

BDIR	BCI	PS to
0	0	chip disabled
0	1	Read one register
1	0	Write into one register
1	1	Select an address

The number of the matching register is stored in the X-register and the data in the accumulator of the 6502 CPU and then passed to the subroutine called OUT. The PSC at this time is not enabled.
When the address is passed over BDIR and BCI go high for a very short period of time.
When the data pass BDIR goes high shortly.

Example: Generating sound A with highest volume on channel A

```
083F A98E      48          LDA  #$8E
0841 A200      49          LDX  #$00
0843 200008    50          JSR  OUT
0846 A93E      51          LDA  #$3E
0848 A207      52          LDX  #7
084A 200008    53          JSR  OUT
084D A90F      54          LDA  #$0F
084F A208      55          LDX  #8
0851 200008    56          JSR  OUT
0854 00        57          BRK
```

Another way to program the PSG is to put the contents of the registers into a table. Then you can write the values via your program into the PSG.

These programs we have seen so far only affect the register of the soundchip. To generate sound and noise you need a few more program parts. They will be comprised substantially of delay routines and checking procedures.

Program: SIREN

Via channel A for approximately 1 s a 440 Hz tone is outputted, after that a tone with 187 Hz is generated for 1s. (Clockfrequency 1 MHz).

```
0855              58  ;
0855              59  ;
0855              60  ;
0855 A93E         61  SIRENE LDA #$3E
0857 A207         62         LDX #7
0859 200008       63         JSR OUT
085C A90F         64         LDA #$0F
085E A208         65         LDX #8
0860 200008       66         JSR OUT
0863 A98E         67  S      LDA #$8E
0865 A200         68         LDX #$00
0867 200008       69         JSR OUT
086A A900         70         LDA #$00
086C A201         71         LDX #01
086E 200008       72         JSR OUT
0871 A9FF         73         LDA #$FF
0873 203308       74         JSR WAIT
0876 A901         75         LDA #$01
0878 A201         76         LDX #$01
087A 200008       77         JSR OUT
087D A94E         78         LDA #$4E
087F A200         79         LDX #$00
0881 200008       80         JSR OUT
0884 A9FF         81         LDA #$FF
0886 203308       82         JSR WAIT
0889 18           83         CLC
088A 90D7         84         BCC S
0833              36  ;
0833 38           37  WAIT   SEC
0834 48           38  W2     PHA
0835 E901         39  W3     SBC #$01
0837 D0FC         40         BNE W3
0839 68           41         PLA
083A E901         42         SBC #$01
083C D0F6         43         BNE W2
083E 60           44         RTS
083F              45  ;
083F              46  ;
083F              47  ;
```

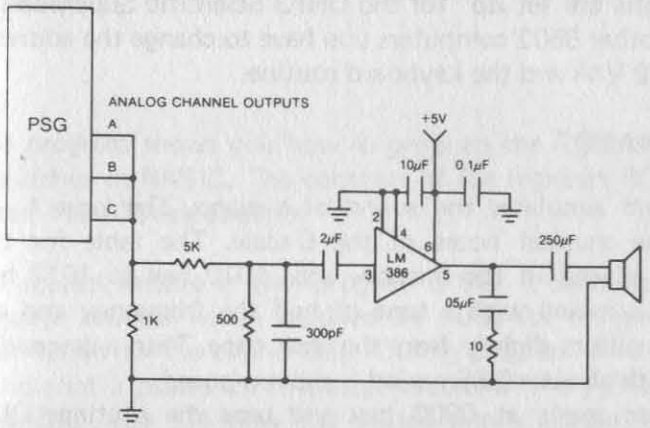

Programing example gunshot

To simulate a gunshot you only need the noise generator and the generator for the envelopes. We set up a table in memory and if a button is pushed, the content of the table is brought into the PSG.

If you change the content of location 1006 (noisefrequency) to 00 (highest noise-period) and location 100C to 40 hex (envelope apr. 2 seconds) you can simulate an explosion.

```
088C              87  TASTE  EQU $FD00
088C              88  ;
088C 202508       89  SCHUSS JSR LADE
088F 2000FD       90         JSR TASTE
0892 18           91         CLC
0893 90F7         92         BCC SCHUSS
0895              93  ;
0895              94  ;
1000              95         ORG $1000
1000 000000       96         HEX 000000000000
1003 000000
1006 0F           97         HEX 0F
1007 07           98         HEX 07
1008 101010       99         HEX 101010
100B 0010        100         HEX 0010
100D 00          101         HEX 00
```

AUDIO OUTPUT INTERFACE



```
0800- A8 A9 FF 8D C3 C0 8D C2
0808- C0 8E C1 C0 A9 03 8D C0
0810- C0 A9 00 8D C0 C0 98 8D
0818- C1 C0 A9 02 8D C0 C0 A9
0820- 00 8D C0 C0 60 A2 00 BD
0828- 00 10 20 00 08 E8 E0 10
0830- D0 F5 60 38 48 E9 01 D0
0838- FC 68 E9 01 D0 F6 60 A9
0840- 8E A2 00 20 00 08 A9 3E
0848- A2 07 20 00 08 A9 0F A2
0850- 08 20 00 08 00 A9 3E A2
0858- 07 20 00 08 A9 0F A2 08
0860- 20 00 08 A9 8E A2 00 20
0868- 00 08 A9 00 A2 01 20 00
0870- 08 A9 FF 20 33 08 A9 01
0878- A2 01 20 00 08 A9 4E A2
0880- 00 20 00 08 A9 FF 20 33
0888- 08 18 90 D7 20 25 08 20
0890- 00 FD 18 90 F7 90
*
```

The program above gives you a hexdump of all the DEMO-programs with the following addresses:

083F – SOUND
0855 – SIREN
088C – GUNSHOT

The programs are set up for the OHIO Scientific Superboard II or III. For other 6502 computers you have to change the addresses for the 6522 VIA and the keyboard routine.

Program "PIANO"

This program simulates the sound of a piano. The keys 1 – 8 refer to the musical notes of the C-scale. The table for that program is placed in the memory area 1010 hex to 1017 hex. Each tone is mixed with a tone of half the frequency and one tone which differs slightly from the basic tone. Then a descending envelope with about a 0.85 period is superimposed.
The program starts at 0900 hex and uses the routines OUT, LOAD and KEYB


```
0900            104          ORG $900
0900            105  ;
0900            106  FTAB    EQU $1010
0900            107  ;
0900 2000FD     108  PIANO   JSR TASTE
0903 290F       109          AND #$0F
0905 AA         110          TAX
0906 CA         111          DEX
0907 BD1010     112          LDA FTAB,X
090A 8D0010     113          STA TAB
090D AA         114          TAX
090E CA         115          DEX
090F 8E0210     116          STX TAB+2
0912 4A         117          LSR
0913 8D0410     118          STA TAB+4
0916 202508     119          JSR LADE
0919 4C0009     120          JMP PIANO
091C            121  ;
091C            122  ;
1000            123          ORG $1000
1000 000000     124          HEX 000000000000
1003 000000
1006 0038       125          HEX 0038
1008 101010     126          HEX 101010
100B 000A00     127          HEX 000A00
100E 0000       128          HEX 0000
1010 EFD5BE     129          HEX EFD5BEB39F8E7F75
1013 B39F8E
1016 7F75
```

This program shows you how to program the registers in the GI-Soundchip in BASIC. The contents of the registers R0....R13 are placed in DATA-statements.

The special feature of that program is that it contains a machine language routine which supplies the pulse for bringing over the information to the sound chip. During program development we found that a pulse, which was generated with a POKE-command in BASIC, was too slow and caused unpredictable functions of the AY-3-8912 chip.

Line 1–5 POKE USR(X) routine into memory
" 7 Starting address of USR(X)
Lower byte → dec (0230 hex)
Higher byte → 12 dec
Line 10 Switch all lines of port A and B of 6522 to outputs.
Line 210 Stop the present sound while disabling all channels of the sound chip
Line 230 See line 210
Line 1000– 1040 Writing a value DCA into a register with the address...

Program: Sound-Demo for the Challenger Superboard

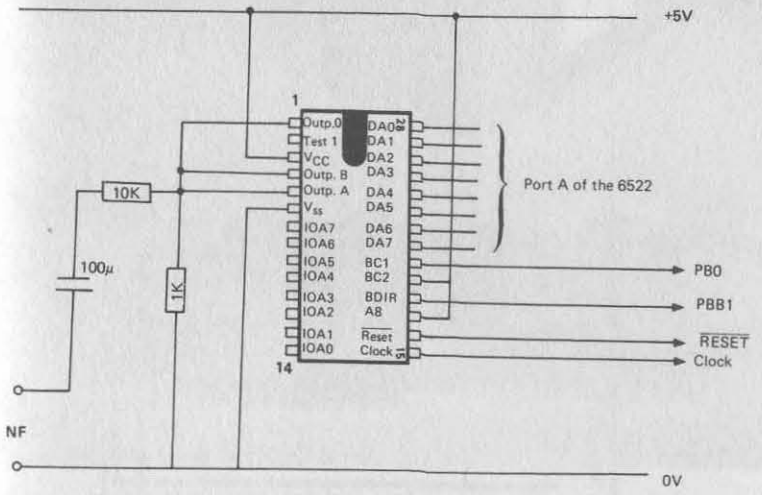
```
1 POKE656,169:POKE657,3
2 POKE658,141:POKE659,192:POKE660,192
3 POKE661,169:POKE662,0
4 POKE663,141:POKE664,192:POKE665,192
5 POKE666,96
7 POKE11,144:POKE12,2
10 POKE49346,255:POKE49347,255
20 DIMD(14)
100 FORX=1TO30:PRINT:NEXT
110 PRINTTAB(6);"GERAEUSCHDEMO"
120 FORX=1TO10:PRINT:NEXT
130 FORX=1TO3000:NEXT
140 FORX=1TO20:PRINT:NEXT
150 READG$
160 PRINTTAB(8);G$
170 FORX=1TO10:PRINT:NEXT
180 GOSUB500
190 FORX=1TO5000:NEXT
195 IFG$="MEER"THENFORX=1TO10000:NEXT
200 Y=Y+1:IFY<5THEN230
210 A=7:D(A)=255:GOSUB1000
220 Y=0:RESTORE:GOTO100
230 A=7:D(A)=255:GOSUB1000
240 GOTO140
500 FORA=0TO13
510 READD(A)
520 GOSUB1000
530 NEXTA
540 RETURN
```



```
1000 POKE49344,0:POKE49359,A
1010 POKE657,3:X=USR(X)
1020 POKE49344,0:POKE49359,D(A)
1030 POKE657,2:X=USR(X)
1040 RETURN

 5000 DATA"PIANO",200,0,201,0,100,0,0,248,16,16,16,0,20,8
 5010 DATA"EXPLOSION",0,0,0,0,0,0,31,7,16,16,16,0,20,0
 5020 DATA"SCHUSS",0,0,0,0,0,0,15,7,16,16,16,0,16,0
 5030 DATA"LOKOMOTIVE",0,0,0,0,0,0,15,199,16,16,16,180,2,12
 5040 DATA"MEER",0,0,0,0,0,0,31,199,16,16,16,255,60,14
OK
```

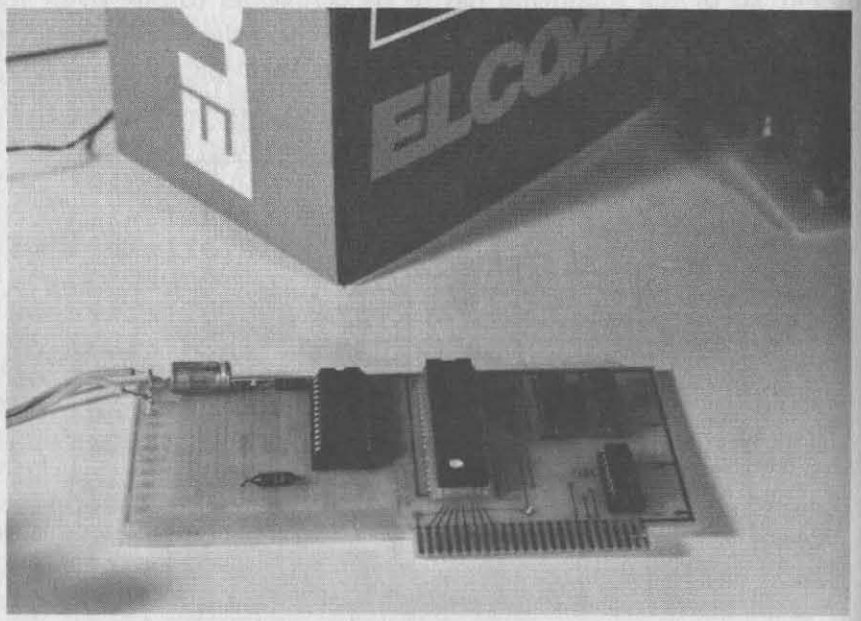
Soundgenerator with AY-3-8912

To construct your sound generator board you first have to assemble the 6522 VIA board, as described earlier. Then you use the prototyping area on the left hand side of the board to assemble the sound circuitry. Place the AY-3-8912 soundchip so that the input lines DA0–DA7 match with the outputs PA0–PA7 of the 6522 VIA (see schematic). Now you have to cut the lines which connect the soundchip to the pins PB0 – PB3 (four lines). Pin 20 of the soundchip has to be connected to pin 10 of the 6522, pin 19 to + 5V, pin 18 to pin 11 and the 6522, pin 17 to +5V, pin 16 to pin 34 of the 6522, pin 15 to pin 25 of the 6522 pin 6 to ground and pin 3 to +5V.



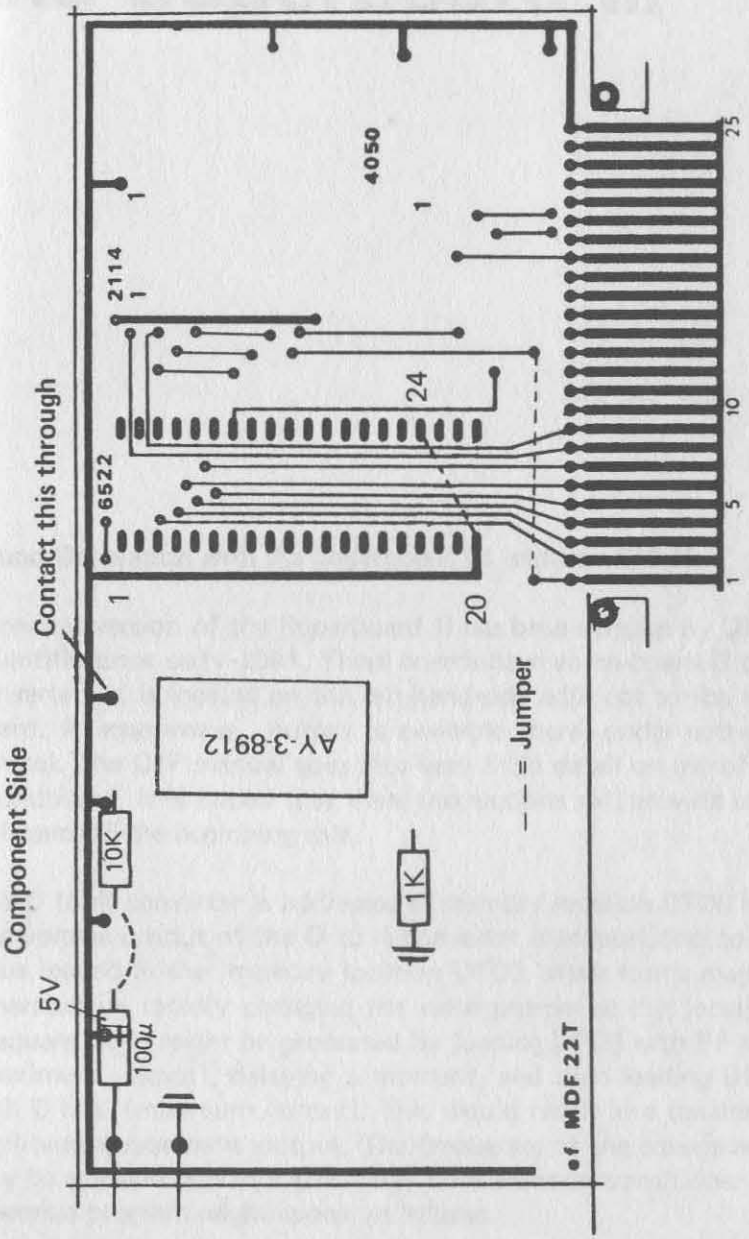
Pins 1, 4, 5 are the common output of the AY-3-8912

You can hook them to the next convenient trace on the printed circuit (P.C.) board. From this foil connect a 1K register to ground.
Then connect a 10 KΩ resistor with a 100μF capacitor to the output which goes to your audio amplifier. At the 6522 VIA chip connect pin 24 with pin 20. On the component side of the pc-board you need a small jumper (see schematic) and you must also bring to the component side the +5V supply voltage over from the soldering side.



The AY-3-8912 Soundboard

Parts Layout and construction of the Sound Board



Sound generation with the Superboard III

Sound Generation with the Superboard III with the C1P II

A revised version of the Superboard II has been shipped by OHIO Scientific since early 1981. These boards have an on-board D to A converter. It is located on the left-hand-side adjacent to the keyboard. A squarewave output is available there, under software control. The C1P manual goes into very little detail on use of the Soundboard. It is hoped that these instructions will provide some assistance to the beginning user.

The D to A converter is addressed at memory location DF00 hex. The voltage output of the D to A converter is proportional to the value loaded in the memory location DF00. Wave forms may be generated by rapidly changing the value present at this location. A square wave might be generated by loading DF00 with FF hex. (maximum output), delaying a moment, and then loading DF00 with 0 hex. (minimum output). This would result in a maximum amplitude square wave output. The frequency of the square wave may be adjusted by varying the delay time between transitions.
A sample program might appear as follows.

```
START  LDA # $FF        FF = maximum volume
       STA $DF00        Output to port
       JSR DELAY
       LDA # $00        00 = minimum volume
       STA $DF00        output to port
       JSR DELAY
       JMP START
```

The delay subroutine might appear as follows:

```
       LDX $..          Load index register
                        with contents of a zero page
                        location
┌─►DEX                  count down
└─BNE M     to zero
       RTS              jump back
```

Clearly, if you use a fix delay value you will obtain a fix frequency square wave. If you intend to vary the frequency of the output it is necessary that you vary the amount of delay time. Following is a sample program to generate the sound of an explosion. This is accomplished with a short BASIC program and a machine language subroutine in a protected RAM area of the Superboard beginning at location 0230 hex. Such an approach provides an interesting extension to arcade games at minimum hardware costs.

The BASIC routine to call the machine language program might appear as follows:

```
10  POKE11,48:POKE12,2
20  X=USR(X)
```

```
                CLD
                SEC
                LDA$13
                ADC $16
                ADC $17
                STA $12
                LDX # 04
Generating a random number ┌►LDA $12,X
                STA $13,X
                DEX
                └─BPL
                LDA $12
                STA $....
                RTS
```


The machine language program called by this routine may be entered using the machine language monitor, or may be entered using POKE and data statements from a BASIC program. A sample machine language routine follows.

```
LDA# $10
STA $D800
0230 A9FF    LDA #$FF
0232 8D8002  STA $0280
0235 851A    STA $1A
0237 A910    LDA #$10
0239 8D00D8  STA $D800
023C 206602  JSR $0266
023F A51A    LDA $1A
0241 8D00DF  STA $DF00
0244 205F02  JSR $025F
0247 A900    LDA #$00
0249 8D00DF  STA $DF00
024C 205F02  JSR $025F
024F CE8002  DEC $0280
0252 D006    BNE $025A
0254 A900    LDA #$00
0256 8D00D8  STA $D800
0259 60      RTS
025A C61A    DEC $1A
025C 4C3C02  JMP $023C
025F AE7F02  LDX $027F
0262 CA      DEX
0263 D0FD    BNE $0262
0265 60      RTS
0266 D8      CLD
0267 38      SEC
0268 A513    LDA $13
026A 6516    ADC $16
026C 6517    ADC $17
026E 8512    STA $12
0270 A204    LDX #$04
0272 B512    LDA $12,X
0274 9513    STA $13,X
0276 CA      DEX
0277 10F9    BPL $0272
```

```
0279 A512    LDA $12
027B 8D7F02  STA $027F
027E 60      RTS
```

```
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0230  A9 FF 8D 80 02 85 1A A9 10 8D 00 D8 20 66 02 A5
0240  1A 8D 00 DF 20 5F 02 A9 00 8D 00 DF 20 5F 02 CE
0250  80 02 D0 06 A9 00 8D 00 D8 60 C6 1A 4C 3C 02 AE
0260  7F 02 CA D0 FD 60 D8 38 A5 13 65 16 65 17 85 12
0270  A2 04 B5 12 95 13 CA 10 F9 A5 12 8D 7F 02 60 16
:
```

Hexdump of the previous program

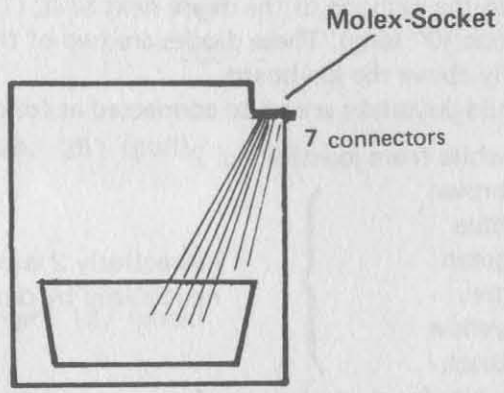
Superboard with Joystick

Superboard with Joystick

How to connect a joystick to the Superboard C1P

The Superboard from Ohio Scientific is very well suited for the experimenters interested in modifying and extending the hardware, thereby widening their professional and leisure experience.

In this chapter we will show how to attach a joystick to the Superboard and how to program exciting games.
The old Superboard has a plug to the left of the keyboard. The newer Superboard III instead has a plug for sound output. If you own a Superboard III and want to attach joysticks to it, you must search the connection spots under the keyboard and wire them. We have attached a Molex-plug to the back of the board and wired it to the keyboard connections.



In our example for two joysticks we used the following signals:
R6, R7, C3, C4, C5, C6, C7

The connection of the joysticks happens at plug J4 or at the self installed plug on the back of the board.
The pin-out appears as follows:

1	– R1
2	– R7
3	– C1
4	– C2
5	– C3
6	– C4
7	– C5
8	– C6
9	– C7
10	– R6
11	– GND
12	– NC

In this case R6 and R7 are the inputs for the two joysticks and C1–C7 supply the bitcombinations depending on the particular position of the joystick. To use two joysticks simultaneously you must neutralize them with diodes. But first you must cut the traces leading to R7 and R6 (about 1/2 " from pin 2 resp. pin 10). Pin 10 is to be connected to the cathode of the outermost diode and pin 2 to the cathode of the diode next to it. (The wire for that will be about 10" long). These diodes are two of the eight you can find directly above the keyboard.
The Fairchild-Joysticks are to be connected as follows:

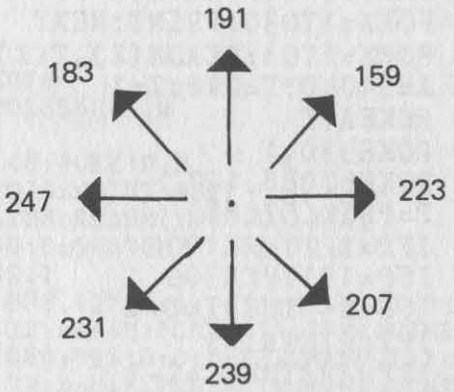
R7 white from joystick no. 2
C2 brown
C3 blue
C4 green
C5 grey
C6 yellow
C7 Black
} respectively 2 wires, neutralized by diodes
R6 white from joystick no. 1

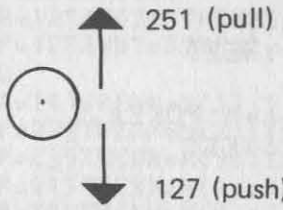
Joystick 1 is initialized by POKE 57088,191, joystick 2 by POKE 57088,127 (57088 is the address of port J4)
The following program shows you, which value is at address 57088 dependent on the joysticks position. You need these values when you want to develop software for the joysticks.

```
100 FORX = 1TO30:?:NEXT
110 POKE 530,1
120 POKE 57088,127
130 ?PEEK (57088)
140 GOTO 130
```

Change line 120 to POKE 57088,191 for the other joystick

You should get the following values if you run the above program:





Sample program 1:
1 Tank, Sound (Superboard III)

Line	Description
50–70	pokeing in the USR(X)-routine (shot) beginning 0230 hex
80	start-address of USR(X)-routine
105	read in the directions of movements and matching graphics-symbols from line 10000
120	disable CTRL-C
130–220	turn of tank, depending on joystick position
300–310	forward movement of the tank
1000–1030	shot (optical and accoustical)

```
50 FORX=0TO78
60 READW:POKE560+X,W
70 NEXTX
80 POKE11,48:POKE12,2
100 FORX=1TO30:PRINT:NEXT
105 FORX=1TO8:READM(X),T(X):NEXT
110 A=54040:T=248:I=1
115 POKEA,T
120 POKE530,1
130 POKE57088,127
140 P=PEEK(57088)
150 IFP=127THEN1000
160 IFP=191THEN300
180 IFP=247THENI=I+1
182 IFI=9THENI=1
190 IFP=223THENI=I-1
192 IFI=0THENI=8
200 POKEA,T(I)
210 M=M(I):T=T(I)
215 FORX=1TO200:NEXT
220 GOTO140
300 POKEA,32:A=A+M:POKEA,T
305 FORX=1TO100:NEXT
310 GOTO140
```


Sample program 2:
2 Tanks, Sound (Superboard III)

Line	Description
50–70	pokeing in the USR(X)-routine
80	start address f USR(X)-routine (lower byte → 11, higher byte → 12)
100	read in the directions of movements and matching graphics-symbols from line 10 000
120	disable CTRL-C
80–300	definition of direction of movement and of graphics-symbol depending on joystick-position
1000–1010	short, control whether a bit
2000–2070	explosion and score
3000–3030	output of final score

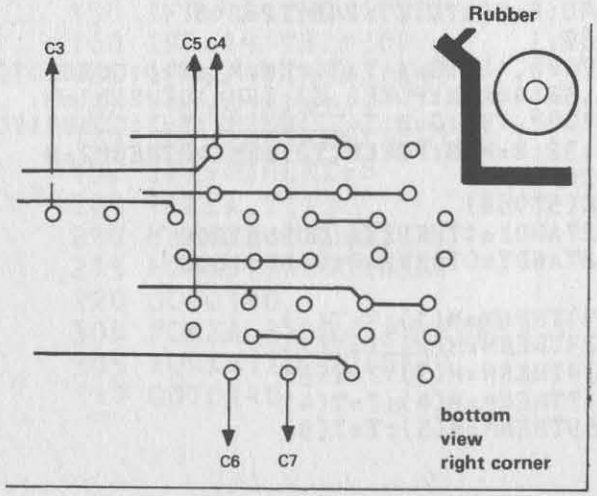
```
50 FORX=0TO78
60 READW:POKE560+X,W
70 NEXTX
80 POKE11,48:POKE12,2
90 FORX=1TO30:PRINT:NEXT
100 FORX=1TO8:READM(X),T(X):NEXT
110 A=54040:B=54070:T1=248:T2=248
120 POKE530,1
130 POKE57088,127:G=A:T=T1:GM=M1:Y=0:GOSUB170:T1=T
135 POKEA,32:A=A+M:POKEA,T1:IFM<>0THENM1=M
140 POKE57088,191:G=B:T=T2:GM=M2:Y=1:GOSUB170:T2=T
150 POKEB,32:B=M+B:POKEB,T2:IFM<>0THENM2=M
160 GOTO130
170 P=PEEK(57088)
190 IFP=127ANDY=1THENZ=A:GOSUB1000
195 IFP=127ANDY=0THENZ=B:GOSUB1000
200 M=0
210 IFP=191THENM=M(1):T=T(1)
220 IFP=223THENM=M(2):T=T(2)
230 IFP=239THENM=M(3):T=T(3)
240 IFP=247THENM=M(4):T=T(4)
250 IFP=159THENM=M(5):T=T(5)
```

```
260 IFP=207THENM=M(6):T=T(6)
270 IFP=231THENM=M(7):T=T(7)
280 IFP=183THENM=M(8):T=T(8)
300 RETURN
1000 FORX=1TO10:POKEG+X*GM,171
1005 IFPEEK(Z)=171THEN2000
1006 POKEG+X*GM,32
1007 NEXT
1008 X=USR(X)
1010 RETURN
2000 E=42
2010 FORX=1TO5:POKEZ+X,E:POKEZ-X,E:POKEZ-32,E:POKEZ+32,E
2020 POKEZ-32*X,E:POKEZ+32*X,E:POKEZ-32*X,E:NEXTX
2030 IFE=42THENE=32:GOTO2010
2035 X=USR(X)
2040 FORX=1TO25:PRINT:NEXT
2042 A$="B"
2045 IFY=0THENA$="A"
2050 PRINT"  S C O R E    "
2055 PRINT
```

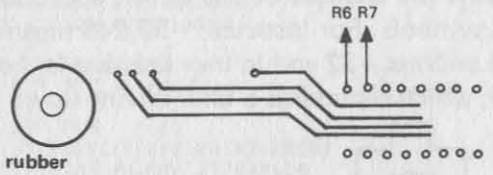
Installation of MOLEX-plug at Superboard III

Connect the points C3, C4, C5, C6, C7, R6, R7 with the matching pins at the MOLEX plug at the backside of the Superboard III.
picture a)

picture A



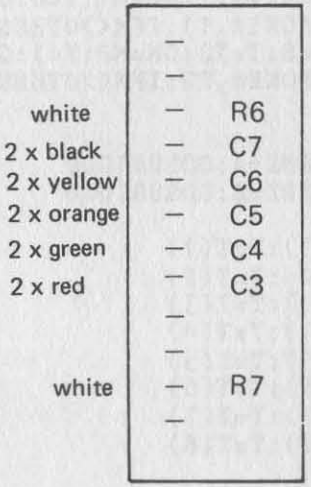
picture B



bottom view, left corner under the keyboard

Picture a) shows the Superboard III under the keys (soldering side)
Picture b) shows the connections of R6 and R7, how to wire them to the cathodes of the two diodes.

picture c)
12 pin plug on the backside of the board



If your connections match the picture shown above you can control your tank by turning the joystick. If you want to use only one joystick, connect C3 – C7 and R6. If you want to use two sticks C3–C7 have to be connected, neutralized by diodes, the whire from stick 2 is to be connected to R7.

Sample program 3:
Line 10 000 contains the changes of the screen addresses and the matching graphics symbols. For instance: –32,248 means that the new address is old address –32 and in that location is the graphics symbol 248 poked, which represents a tank driving upwards;

–32 accounts for the line length (–32 equals 1 line up)
–32 equals –32 + / which is a movement to the upper right. If you use a computer with another line length you have to change those numbers.

```
90 FORX=1TO30:PRINT:NEXT
100 FORX=1TO8:READM(X),T(X):NEXT
110 A=54040:B=54010:T1=248:T2=248
120 POKE530,1
130 POKE57088,127:G=A:T=T1:GM=M1:Y=0:GOSUB170:T1=T
135 POKEA,32:A=A+M:POKEA,T1:IFM<>0THENM1=M
140 POKE57088,191:G=B:T=T2:GM=M2:Y=1:GOSUB170:T2=T
150 POKEB,32:B=M+B:POKEB,T2:IFM<>0THENM2=M
160 GOTO130
170 P=PEEK(57088)
190 IFP=127ANDY=1THENZ=A:GOSUB1000
195 IFP=127ANDY=0THENZ=B:GOSUB1000
200 M=0
210 IFP=191THENM=M(1):T=T(1)
220 IFP=223THENM=M(2):T=T(2)
230 IFP=239THENM=M(3):T=T(3)
240 IFP=247THENM=M(4):T=T(4)
250 IFP=159THENM=M(5):T=T(5)
260 IFP=207THENM=M(6):T=T(6)
270 IFP=231THENM=M(7):T=T(7)
280 IFP=183THENM=M(8):T=T(8)
300 RETURN
1000 FORX=1TO10:POKEG+X*GM,171
1005 IFPEEK(Z)=171THEN2000
1006 POKEG+X*GM,32
1007 NEXT
1010 RETURN
2000 E=42
```



```
2010 FORX=1TO5:POKEZ+X,E:POKEZ-X,E:POKEZ-32,E:POKEZ+32,E
2020 POKEZ-32*X,E:POKEZ+32*X,E:POKEZ-32*X,E:NEXTX
2030 IFE=42THENE=32:GOTO2010
2040 FORX=1TO25:PRINT:NEXT
2042 A$="B"
2045 IFY=0THENA$="A"
2050 PRINT"  S C O R E "
2055 PRINT
2060 PRINT" FOR   ";A$
2063 PRINT:PRINT:PRINT:PRINT
2065 INPUT"PLAY AGAIN (Y/N)";Q$
2067 IFQ$="N"THENEND
2070 RESTORE:GOTO90
10000 DATA -32,248,1,250,32,252,-1,254,-31,249,33,251,31,253,-33,255
OK
```

EPROM/RAM board (Bytewide)

EPROM/RAM-Board for 6502 computers

ELCOMP has developed an EPROM/RAM-board which can be connected to all 6502 computers through the expansion board 606. We have used the board with the following computers:

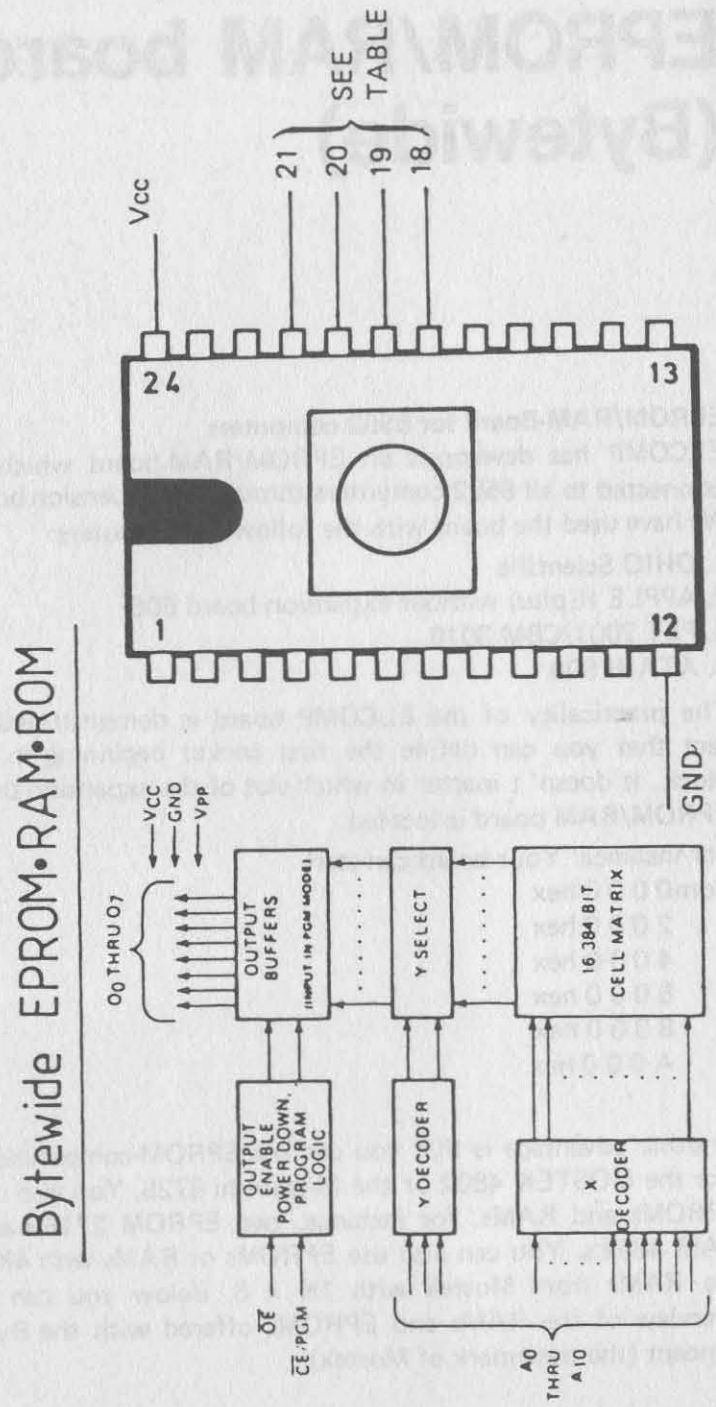
1. OHIO Scientific
2. APPLE II(plus) without expansion board 606
3. PET 2001/CBM 3016
4. ATARI 800

The practicality of the ELCOMP board is demonstrated by the fact that you can define the first socket beginning in any 8K block. It doesn' t matter in which slot of the expansion board the EPROM/RAM board is located.

For instance: Your board can start
from0 0 0 0 hex
2 0 0 0 hex
4 0 0 0 hex
6 0 0 0 hex
8 0 0 0 hex
A 0 0 0 hex
:

Another advantage is that you can use EPROM-compatible RAMs like the MOSTEK 4802 or the Mitsubishi 8725. You also can mix EPROMs and RAMs: for instance, two EPROM 2716's and two RAM 4802's. You can also use EPROMs or RAMs with 4K x 8 or the RAMs from Mostek with 1K x 8. Below you can see an overview of the RAMs and EPROMs offered with the Bytewide concept (the trademark of Mostek).

Bytewide EPROM•RAM•ROM



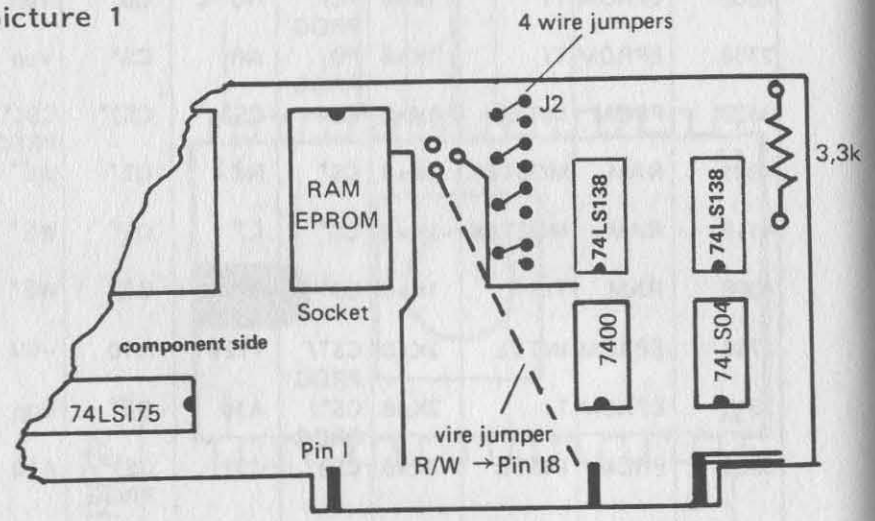
24 PIN MEMORY EPROMS & RAMS

DEVICE	TYPE	MANUF	SIZE	PIN 18	PIN 19	PIN 20	PIN 21
2708	EPROM	TI	1Kx8	PROG	+12V	CS*(PE)	–5V
2508	EPROM	TI	1Kx8	PC/ PROG	NC	CS*	Vpp
2758	EPROM	TI	1Kx8	PD/ PROG	AR	CS*	Vpp
3628	PROM	INTEL	1Kx8	CS4	CS3	CS2*	CS1*/ PROG
4801	RAM	MOSTEK	1Kx8	CS*	NC	OE*	WE*
4118	RAM	MOSTEK	1Kx8	CS*	L*	OE*	WE*
4008	RAM	TI	1Kx8	CS*	AR	OE*	WE*
2716	EPROM	INTEL	2Kx8	CS*/ PROG	+12V	A10	–5V
2516	EPROM	TI	2Kx8	CS*/ PROG	A10	OE*	Vpp
3636	PROM	INTEL	2Kx8	CS3	CS2	CS1*/ PROG	A10
4802/ 4016	RAM	MOS/TI	2Kx8	CS*	A10	OE*	WE*
2732	EPROM	INTEL	4Kx8	CS*	A10	OE*/ PROG	A11
2532	EPROM	TI	4Kx8	A11	A10	PD/ PROG	Vpp
4732	PROM	TI	4Kx8	A11	A10	CS	CS*/ PROG
4764	PROM	TI	8Kx8	A11	A10	S*/ PROG	A12
58725	RAM	MITSU-BISHI	2Kx8	CS	10	OE	WE

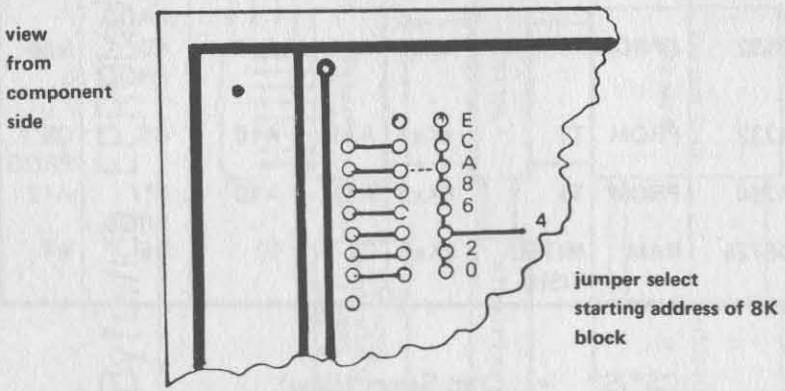
CS*/S* = Chip Select (Low)
OE* = Output Enable (Low)
WE* = Write Enable (Low)
PD = Power Down
PROG/(PE)= Program Enable

Vpp	=	25V (Program Voltage)
L*	=	LATCH (LOW)
AR	=	Array: If True Output VI0 If False Output VI1

picture 1



picture 2



For the use of EPROM-compatible RAMs you have to connect the R/$\overline{W}$-signal to the common $\overline{W}$/E (write enable) on the EPROM board (order no. 609). The R/$\overline{W}$-signal is on pin 18 of the bus on the ELCOMP-expansion board.

The equipping of the board is easy (see picture). Use a socket for each IC.
The definition of the start address of the first socket happens by making connections on the soldering side of the board in the upper left corner. Make a connection where you want to have your start address. If you make a connection as in picture 2, your start address on the first socket will be $A000.

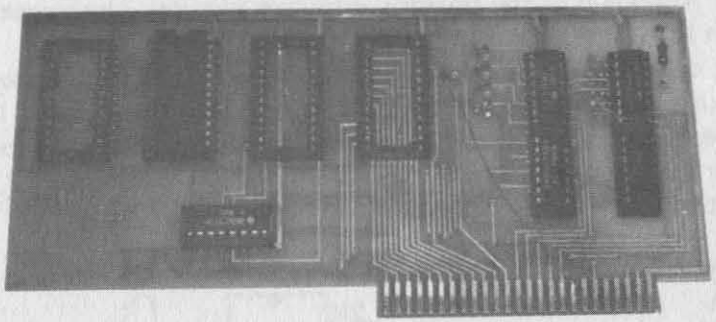
1. socket	A000 –	2K
	A7FF	
2. socket	A800 –	2K
	AFFF	
3. socket	B000 –	2K
	B7FF	
4. socket	B800 –	2K
	BFFF	

If you define the startaddress to C000, it looks similar:
C000–C7FF, C800–CFFF, D000–D7FF, D800–DFFF.

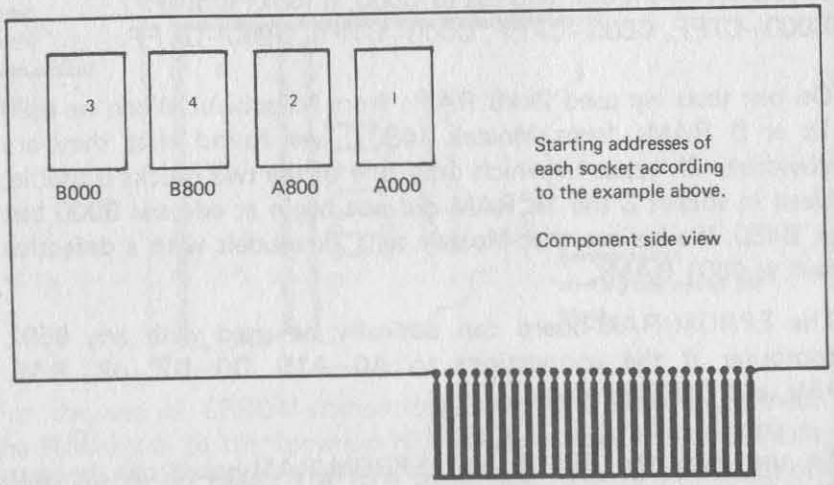
On our tests we used 2kx8 RAMs from Mitsubishi. When we used 1k x 8 RAMs from Mostek (4801) we found that they are obviously 2k types in which only one of the two blocks is usable. Used in socket 3 the 1k RAM did not begin at address B000 but at B400. We believe that Mostek sells 2k models with a defective half as 4801 RAMs.

The EPROM/RAM-board can basically be used with any 6502 computer if the connections to A0–A15, D0–D7, ϕ2, R/W, +5V, and GND are made.

To use with the APPLE, the EPROM/RAM-board can be put directly to an empty slot.



Here you can see the EPROM/RAM-board (component side). In socket no. 3 there is a 2k x 8 static RAM.
The advantage of the EPROM/RAM-board is that you can test programs in RAM at the right place in memory and replace it by an EPROM in the same place if the program is acceptable.
For instance:
If you want to relocate a monitor program from A000 to B000, put the monitor into the # 1 socket and a RAM 8725 into the # 3 socket. Relocate the program to the RAM beginning at B000. Then you can get the program to the area beginning $1000 with a block-move and burn a new EPROM.
The Superboard 8K is free for RAM/EPROM-expansion from $2000–9FFF (BASIC starts at A000).



Schematics of the EPROM/RAM board (Bytewide)

EPROM-board 4 x 2716

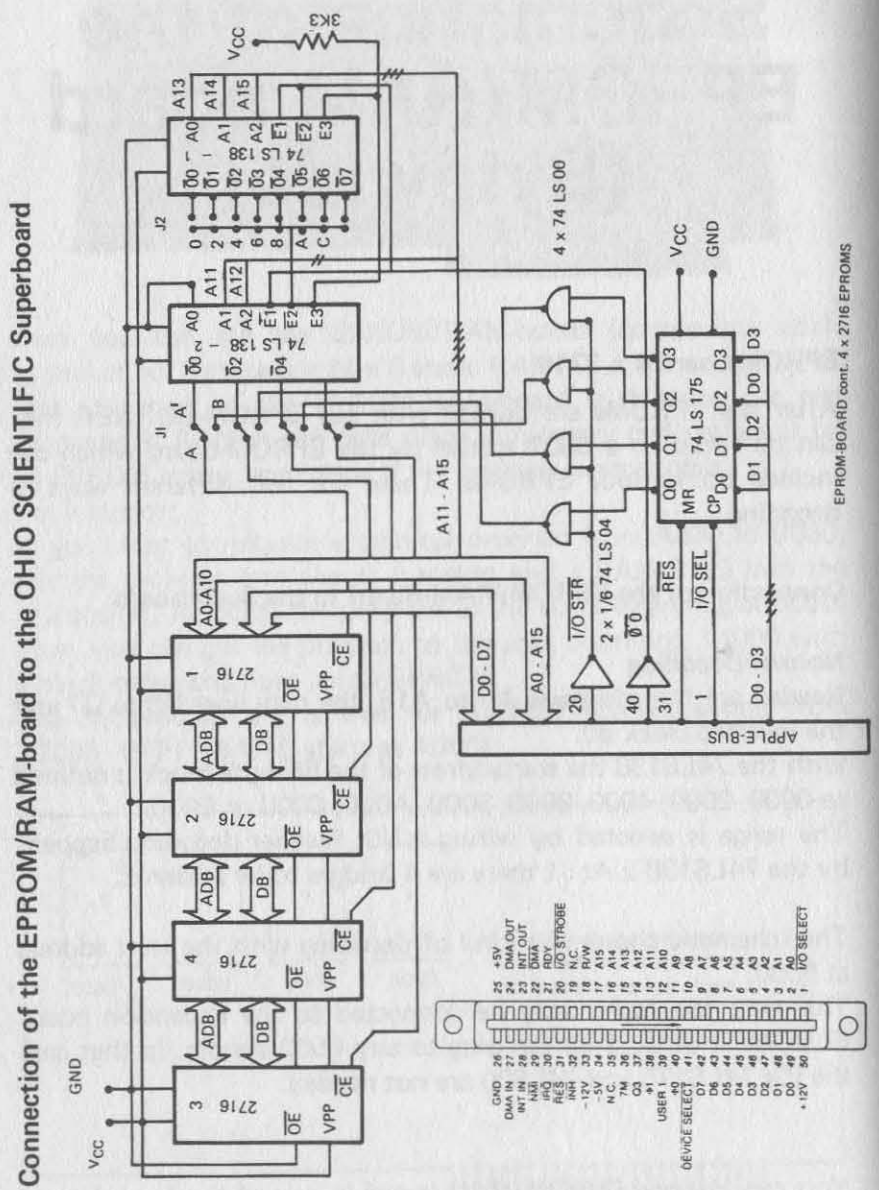
After the EPROMs are burned with our EPROM-BURNER they can be wired to a 6502 system by the EPROM-board which can include up to four EPROMs. There are two different ways of decoding.

Connection of the EPROM/RAM-Board to the Superboard

Normal Decoding
Needed are the addresses A0 to A15, the data lines D0 to D7 and the inverted clock ϕ0.
With the 74LS138 the startaddress of the 8K byte block is defined to 0000, 2000, 4000, 6000, 8000, A000, C000 or E000.
The range is selected by wiring at J0. Further decoding happens by the 74LS138-2.At J1 there are 4 bridges to be soldered.

The schematic shows this kind of decoding with the start address at 8000.
This way the board may be connected to the expansion board ELCOMP-1 or by a 50 pin plug to any 6502 system. in that case the ICs 74LS175 and 74LS00 are not needed.

You can put the EPROM/RAM-board in any of the four slots of the expansion board 606. It is important that you by-pass the driver on the Superboard. The best way to do this is to use 16-pin sockets and make the connections as shown in the figure below.



How to assemble the board

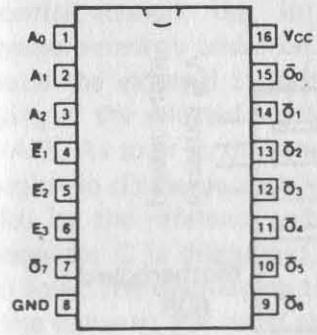
Parts list:

- 4 x 24 pin DIL-sockets
- 1 x 74LS175
- 1 x 7400
- 1 x 74LS04
- 1 x EPROM/RAM board
- 2 x 14pin DIL sockets
- 1 x 16pin DIL-sockets

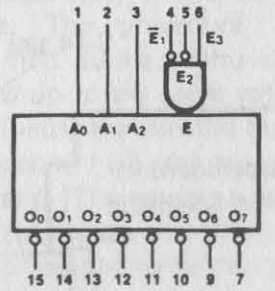
First solder all the sockets for the integrated circuits. Then wire the necessary jumpers.

74S138
74LS138

CONNECTION DIAGRAM
PINOUT A



LOGIC SYMBOL

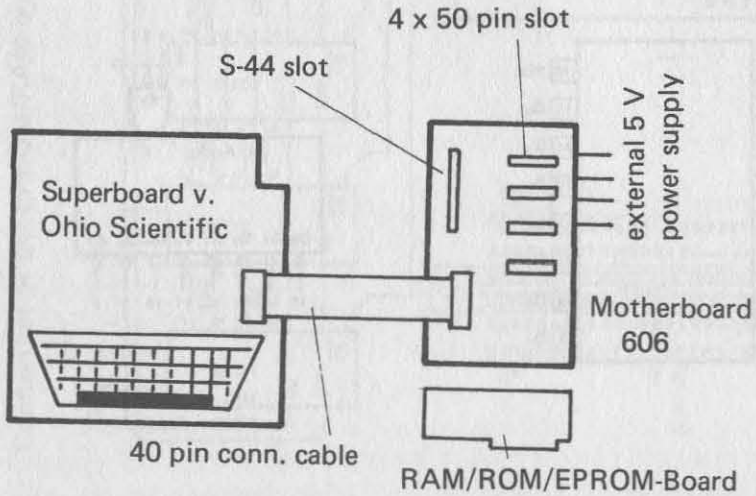


VCC = Pin 16
GND = Pin 8

TRUTH TABLE

INPUTS						OUTPUTS							
$\overline{E_1}$	$\overline{E_2}$	E_3	A_0	A_1	A_2	$\overline{O_0}$	$\overline{O_1}$	$\overline{O_2}$	$\overline{O_3}$	$\overline{O_4}$	$\overline{O_5}$	$\overline{O_6}$	$\overline{O_7}$
H	X	X	X	X	X	H	H	H	H	H	H	H	H
X	H	X	X	X	X	H	H	H	H	H	H	H	H
X	X	L	X	X	X	H	H	H	H	H	H	H	H
L	L	H	L	L	L	L	H	H	H	H	H	H	H
L	L	H	H	L	L	H	L	H	H	H	H	H	H
L	L	H	L	H	L	H	H	L	H	H	H	H	H
L	L	H	H	H	L	H	H	H	L	H	H	H	H
L	L	H	L	L	H	H	H	H	H	L	H	H	H
L	L	H	H	L	H	H	H	H	H	H	L	H	H
L	L	H	L	H	H	H	H	H	H	H	H	L	H
L	L	H	H	H	H	H	H	H	H	H	H	H	L

H = HIGH Voltage Level
L = LOW Voltage Level
X = Immaterial



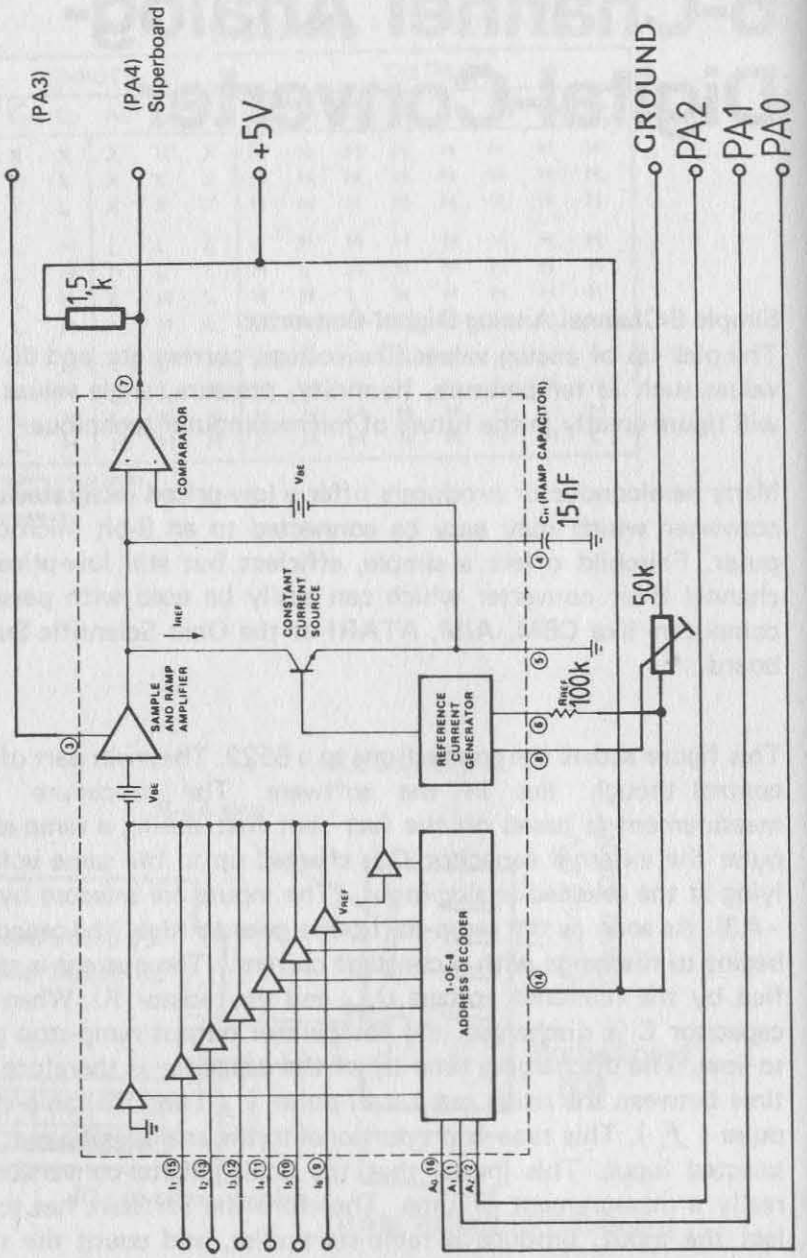
6-Channel Analog-Digital-Converter

Simple 6-Channel Analog-Digital-Converter

The pick-up of analog values like voltage, current etc. and divided values such as temperature, humidity, pressure, angle values etc. will figure greatly in the future of microcomputer technique.

Many semiconductor producers offer a low-priced integrated A/D converter which may easy be connected to an 8-bit microcomputer. Fairchild offers a simple, efficient but still low-prices 6-channel 8-bit converter which can easily be used with personal computers like CBM, AIM, ATARI or the Ohio Scientific Superboard.

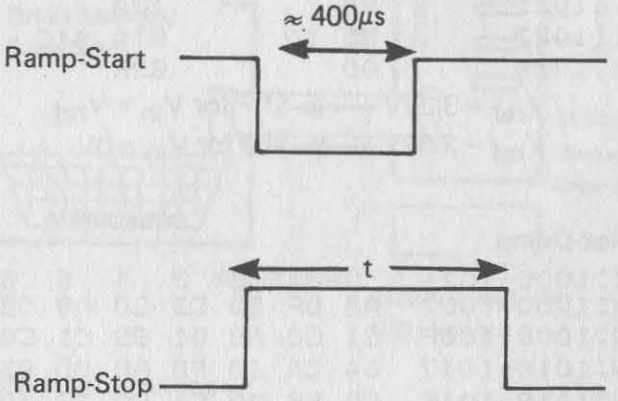
This figure shows the connections to a 6522. The main part of the control though lies in the software. The procedure of measurement is based on the fact that first during a ramp-start-pulse the external capacitor C is charged up to the same voltage lying at the selected analog-input. (The inputs are selected by A0 –A3). As soon as the ramp-start-pulse goes to high, the capacitor begins to discharge with a constant current. (The current is specified by the reference voltage U_{ref} and the resistor R). When the capacitor C is discharged, the comparator output ramp-stop goes to low. The discharging time (t) of the capacitor is therefore the time between the rising ramp-start-pulse (⌐) and the ramp-stop-pulse (¬). This time is proportional to the analog-voltage at the selected input. This means that the analog-digital-conversion is really a measurement of time. Therefore the software has to select the input, produce a ramp-start-pulse, and count the time until the ramp-stop goes to low. Since it is a real-time-problem the software has to be written in machine language.



Description of measurement:
During the ramp-start-pulse the capacitor C is charged with the voltage (Vin) lying at the selected analog-input. As soon as the ramp-start-pulse ends, the capacitor starts to discharge with a constant current (defined by V_{ref} and R_{ref}).

As soon as the capacitor is completely discharged, the ramp-stop-pulse goes to low. This means that the time between end of ramp-start and end of ramp-stop has to be measured by software. This time is directly proportional to the applied voltage.

Lapse of time of the signals:



The following program shows, how the time can be taken. The ADCµA9708 was then connected to PORT A of a 6522. A0 to PA0, A1 to PA1, A2 to PA2, ramp-start to PA3 and ramp-stop to PA4. In this example there is analog input 1 selected. If you want to select another input, for instance, no. 6 you have to change 2 lines:

line 5 from LDA # $01 to LDA # $06 (A9 06)
line 10from LDA # $09 to LDA # $0E (A9 0E)

```
D:1000-1024    1  2  3    MNC-CODE
I:1000         A9 0F       LDA =$0F
I:1002         8D C3 C0    STA $C0C3
I:1005         A9 08       LDA =$08
I:1007         8D C1 C0    STA $C0C1
I:100A         A9 01       LDA =$01
I:100C         8D C1 C0    STA $C0C1
I:100F         A2 64       LDX =$64
I:1011         CA          DEX
I:1012         10 FD       BPL $1011
I:1014         A9 09       LDA =$09
I:1016         8D C1 C0    STA $C0C1
I:1019         E8          INX
I:101A         AD C1 C0    LDA $C0C1
I:101D         29 10       AND =$10
I:101F         D0 F8       BNE $1019
I:1021         8A          TXA
I:1022         85 10       STA $10
I:1024         00          BRK
```

V_{ref} = 3.32V ⟶ $FF for $V_{in} = V_{ref}$

V_{ref} = 3.32V ⟶ $03 for V_{in} = 0V

Hex-Dump

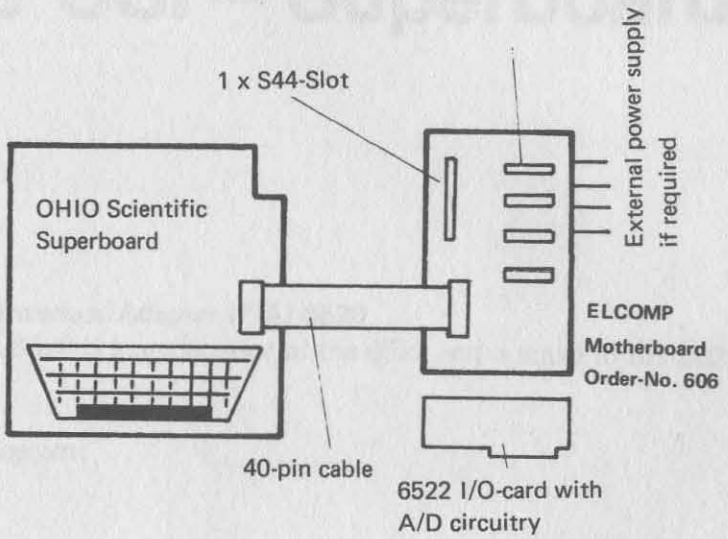
```
M:1000-1024    0  1  2  3  4  5  6  7
W:1000-1007   A9 0F 8D C3 C0 A9 08 8D
W:1008-100F   C1 C0 A9 01 8D C1 C0 A2
W:1010-1017   64 CA 10 FD A9 09 8D C1
W:1018-101F   C0 E8 AD C1 C0 29 10 D0
W:1020-1027   F8 8A 85 10 00 AA AA AA
```

The analog voltage V_{in} can be between 0 V and the reference voltage V_{ref}. For best resolution V_{ref} is set so that $V_{in} = V_{ref}$ causes $FF (or 255) in location $10. With V_{ref} = 3.32 V we got $FF, with $V_{in} = V_{ref}$ and $03 with V_{in} = 0V.

If you replace the break-command at the end of the machine-language routine by a RTS (=60) you can call the routine from BASIC and print the value on the screen:

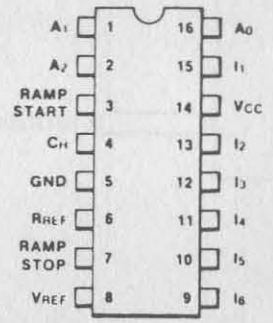

```
10 POKE 11,0
20 POKE 12,16
30 X = USR(X)
40 PRINT PEEK (16)
50 GOTO 30
```

Circuit with the Superboard



CONNECTION DIAGRAM
16-PIN DUAL IN-LINE

(TOP VIEW)
PACKAGE OUTLINES 7B 9B
PACKAGE CODES D P



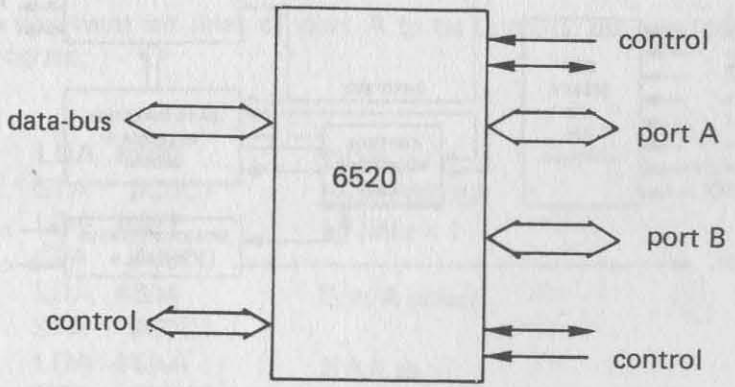
ORDER INFORMATION

TYPE	PART NO.
µA9708	µA9708DM
µA9708	µA9708DC
µA9708	µA9708PC

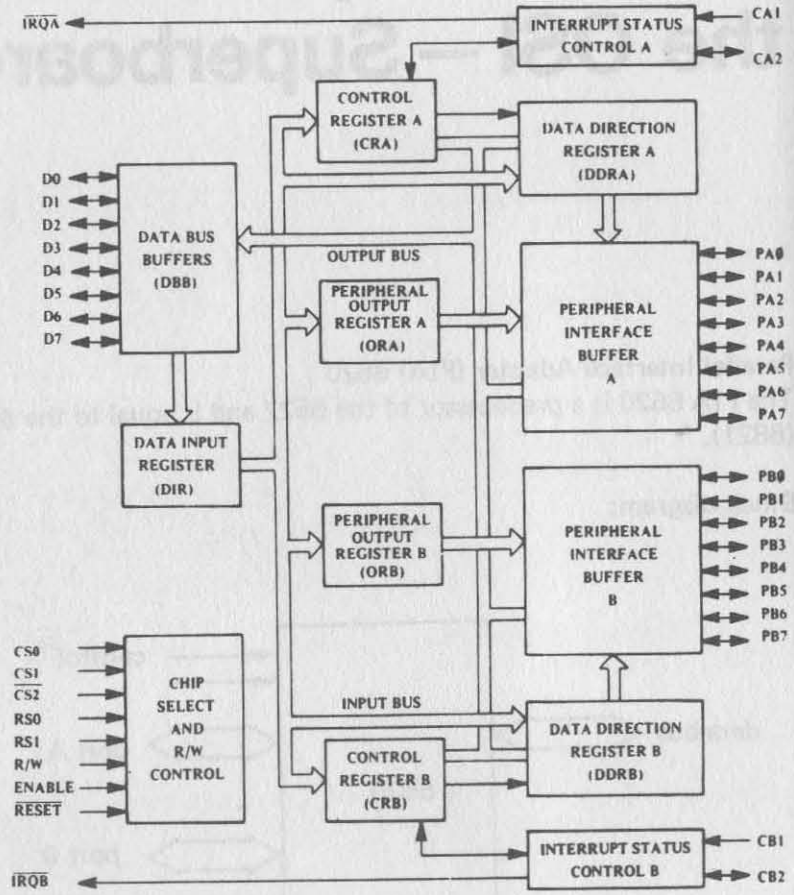
Parallel Interface Adapter (PIA) 6520 Memory expansion for the OSI – Superboard

Parallel Interface Adapter (PIA) 6520
The PIA 6520 is a predecessor of the 6522 and is equal to the 6820 (6821).

Block diagram:



Internal structure:
The block diagram shows that the 6520 has two ports, A and B and four control lines for communication with the outside world. To the processor side there is the 8-bit data bus and control lines. For details see figure below.



Internal structure of the 6520

Port A and B are controlled by the data direction registers D DRA and DDRB and the control registers CRA and CRB. So there are six registers but only two address lines R50 and R51 (RS=register select). The data direction registers and the port have the same address. One bit in the control register (CRA and CRB) define which one of the two registers will be selected (see figure below).

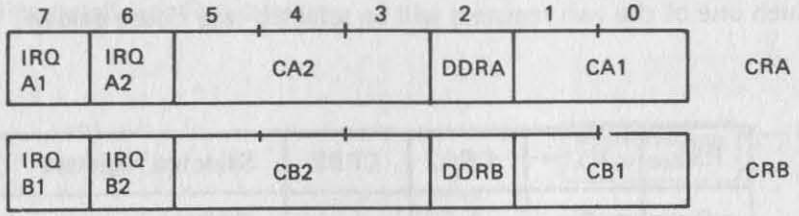
RS0	RS1	CRA2	CRB2	Selected registers
0	0	1	–	PORTA
0	0	0	–	DDRA
0	1	–	–	CRA
1	0	–	1	PORTB
1	0	–	0	DDRB
1	1	–	–	CRB

Selection of the registers

If you want all lines of port A to be outputs, use the following program:

```
LDA  #$00     ;   DDR
STA   $C0C1   ;   Select DDRA
LDA  #$FF     ;   all lines = 1
STA   $C0C0   ;
LDA  #$04     ;   Port A select
STA   $C0C1   ;
LDA  #$AA     ;   $AA to
STA   $C0C0   ;   port A
```

It is assumed here that the start address of the 6520 is at $C0C0. Port B may be programed the same way, using control register CRB. The meaning of the bits in the control registers shows the figure below:



Meaning of bits in control registers

Bits 0 and 1 control the inputs CA1 and also CA2. Bit 2 selects the data direction registers. Bits 3, 4, 5 control the bidirectional lines CA2 and also CB2. Bits 6 and 7 mark interrupts. They are usually one. If they are set to zero by external circumstances, this will cause a zero in IRQA and also IRQB and by that an interrupt. The two lines are open-collector-lines and can be connected via "wired or".

CRA (CRB)

Bit 1	Bit 8	Active Transition on CA1(CB1)	$\overline{IRQA}$, ($\overline{IRQB}$)
0	0	↓	Disable – remain high
0	1	↓	Bit 7 = 0 Interrupt
1	0	↑	no interrupt
1	1	↑	Bit 7 = 0 Interrupt

Interrupt Control CA1(CB1)

The above figure shows that an interrupt is triggered if, for instance, CA1 goes low, if bit 0 = 1. These two bits have the same meaning in both control registers. The figure below shows the effect of a pulse on line CA2. If bits 3, 4, 5, are zero, neither a negative nor a positive going edge will cause an interrupt. These three bits also have the same meaning in both registers.

CRA (CRB)

Bit 5	Bit 4	Bit 3	Active Transition on CA2(CB2)	$\overline{IRQA}$ ($\overline{IRQB}$)
0	0	0	↓	,) interrupt
0	0	1	↓	Bit 6 = 0 Interrupt
0	1	0	↑	no interrupt
0	1	1	↑	Bit 6 = 0 Interrupt

Interrupt Control CA2 (CB2)

If bit 5 is set to one, the control lines will act differently. The control register CRA performs the handshake for port A.

With bits 3 and 4 = 0 and bit 5 = 1, CA2 will be an output line. An interrupt on CAI sets line CA2 to 1. This line only becomes 0 again, if a "LOAD FROM PORT A" command had happened. With the next bit-record after a "LOAD FROM PORT A"-command, a negative pulse will be issued to CA2 for a period of one clock-cycle. With the next two bit-records CA2 can be set to 0 or 1.

CRA

Bit 5	Bit 4	Bit 3	Mode	Description
1	0	0	Handshake on read	CA2=1 if interrupt on CA1 CA2=0 if Load-command
1	0	1	pulse output	CA2 = ⊓⊔, after LOAD-command
1	1	0		CA2 = 0
1	1	1		CA2 = 1

Mode handshake read

Control-register CRB controls mode handshake write to port A. With bit 5 = 1 and bits 3 and 4 = 0 CB2 will be set to zero with a "STORE TO PORT B"-command. It will be put back with an active interrupt at CB1. With the bitcombination 101 a negativ pulse will be send to CB2.

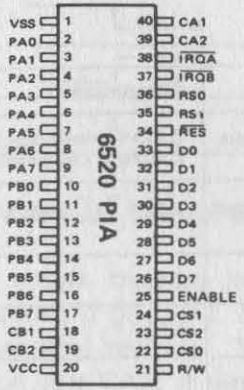
The length of that pulse is also one clock-cycle here. With the remaining two bitcombinations CB2 can be set to zero or one.

CRB

Bit 5	Bit 4	Bit 2	Betriebsart	Beschreibung
1	0	0	Handshake Schreiben	CB2 = 0 bei STORE-Befehl, CB2 = 1 bei Interrupt auf CB1
1	0	1	Ausgangs-impuls	CB2 = ⎍, nach STORE-Befehl
1	1	0		CB2 = 0
1	1	1		CB2 = 1

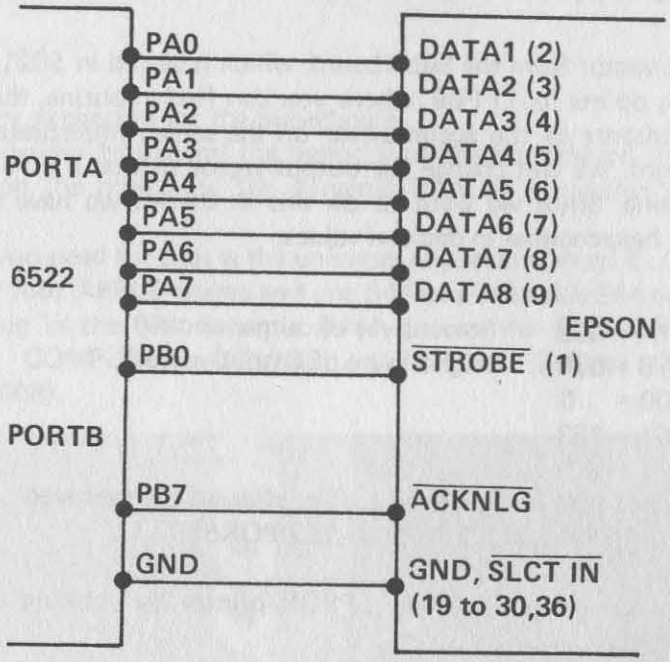
Mode handshake write

As you can see the 6520 is very suitable for data transfer between the microprocessor and external devices.
The best way to build up a circuit with the 6520 is to use the prototyping card from ELCOMP (order no. 604).



Pin-Out of the 6520

Connection of a parallel printer to the Superboard
We have used our expansion board ELCOMP-1 together with our 6522 I/O board in slot 1, to connect an EPSON printer with parallel interface to the OHIO Scientific Superboard.
Port A of the 6522 is used for data-transfer, two lines of Port B for control:



How to connect a parallel printer to the Superboard

On the 6522 I/O board there are 256 bytes ofRAM (addresses $C100–C1FF, if board is in slot 1). Into these locations we write our output routine:

```
C100 8507    STA $07
C102 A9FF    LDA #$FF
C104 8D93C0  STA $C093
C107 A903    LDA #$03
C109 8D92C0  STA $C092
C10C A902    LDA #$02
C10E 8D90C0  STA $C090
C111 A507    LDA $07
```

```
C113 8D91C0 STA $C091
C116 4E90C0 LSR $C090
C119 0E90C0 ASL $C090
C11C AD90C0 LDA $C090
C11F 10FB   BPL $C11C
C121 A507   LDA $07
C123 4C69FF JMP $FF69
```

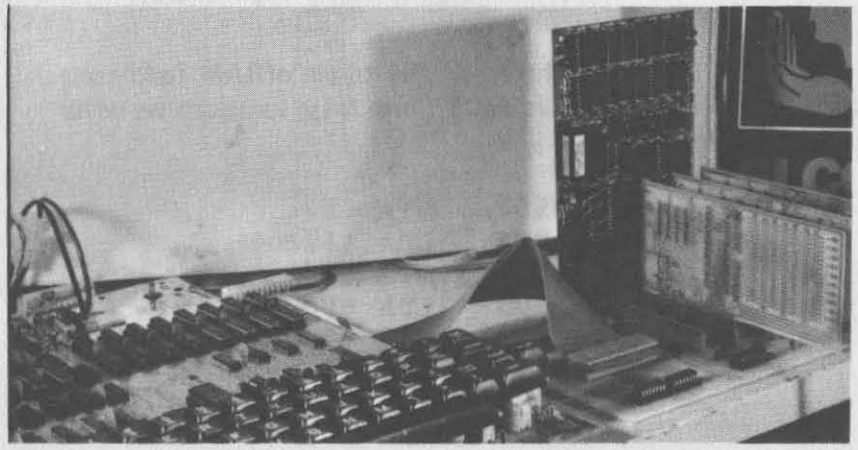
The output vector from the Superboard, which is stored in $021A and $021B, points to $FF69, where you can find a routine, that puts the contents of the accumulator on the screen (interpreted as ASCII sign). We will change the output-vector so that it points to our routine. Since we want to do this in BASIC we have to convert the hexadecimal to decimal values:

```
$021A = 538    (lower byte of output-vector)
$021B = 539    (higher byte of output-vector)
  $00 =   0
  $C1 = 193
```

and we have to turn on the save-flag. So, after we have entered

```
POKE 538,0:POKE539,193:POKE517,1
```

we can list our program on the EPSON printer by entering the command LIST.



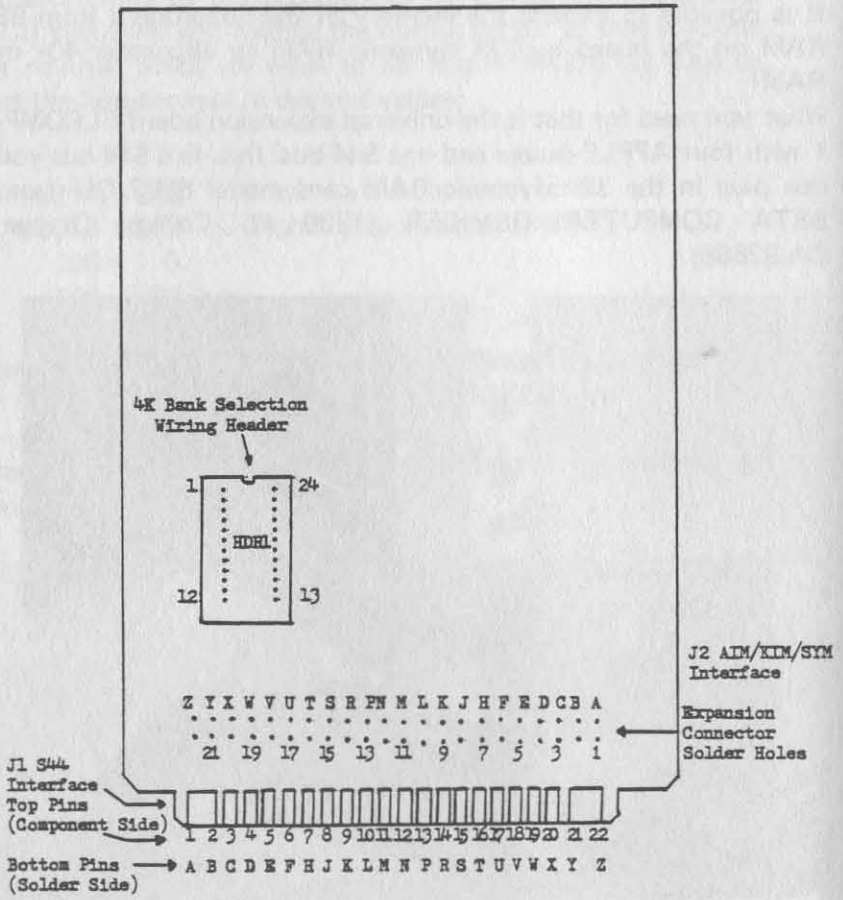
Memory expansion for the Superboard

It is possible to expand the memory of the Superboard from 8k RAM on the board by 32k dynamic RAM to altogether 40k of RAM!

What you need for that is the universal expansion board ELCOMP-1 with four APPLE-busses and one S44-bus. Into this S44-bus you can plug in the 32k dynamic RAM card model 6502 DM from BETA COMPUTER DEVICES (1230 W. Collins, Orange, CA 92668).

32 K dynamic RAM board from BETA Computer Devices

If you use revision B of this board it should work without any changes. If you use revision C or revision D of this RAM-card notice that there are four changes:
A12 must be at pin K (is at C), A13 must be at pin 14 (is at X), A14 must be at pin R (is at B), A15 must be at pin 9 (is at Y).
On the memory expansion card there is a 24pin bank selection wiring header. Here you can define the start addresses of the eight 4k banks.



Interface connection identification

The 8K of RAM on the Superboard itself have the addresses $ 0000 to $ IFFF. This means we have to set the start address of the first bank on the expansion board to $2000, the second bank to $3000 and so on up to $9000 for the eighth bank. The addresses of our expansion board are thus $2000 – $9FFF. The next address $A000 is already BASIC in ROM.
The eight connections on the bank selection leader are.
pin 17 to pin 14, 18 to 13, 19 to 12, 20 to 11, 21 to 10, 22 to 9, 23 to 8, 24 to 7.

Address Bank Selection Header Pin Identification

4K Bank Selection Header

Memory Bank Decode (A12–A15)

Pin	Bank
1	Fxxx
2	Exxx
3	Dxxx
4	Cxxx
5	Bxxx
6	Axxx
7	9xxx
8	8xxx
9	7xxx
10	6xxx
11	5xxx
12	4xxx
13	3xxx
14	2xxx
15	1xxx
16	0xxx

Address Selection Encode	
Pin	Bank
17	Bank 0
18	Bank 1
19	Bank 2
20	Bank 3
21	Bank 4
22	Bank 5
23	Bank 6
24	Bank 7

Address bank selection header pin identification

Other very important book of interest

The First Book of Ohio Scientific

Introduction to OSI computers. Diagrams, hardware and software information not previously available in one compact source.

TABLE OF CONTENTS

$7.95

The Second Book of OHIO-Scientific

Very valuable information about OSI microcomputer systems. Introduction to OS65D and OS 65U. Networking. Hard- and software hints and tips. Systems specification. Business applications.

TABLE OF CONTENTS

$7.95

The Fourth Book of OHIO Scientific

Over 40 programs, games, personal, math functions, hints, memory map.

TABLE of CONTENTS

$ 9.95

Superboard

expansion system

ELCOMP announces a new low cost expansion path for owners of 6502-based microcomputers. (APPLE, OSI, AIM KIM, PET etc.) . The basic element is the ELCOMP Expansion Board. This board provides:

1. Four APPLE-compatible fifty-pin slots
2. One S-44 card slot
3. Low cost, low parts count
4. Low power consumption
5. Extremely simple interfacing to existing systems
6. Provision for prototyping custom interfaces
7. On-board decoding enables use as motherboard expansion to APPLE systems.
8. Complete, accurate documentation

The ELCOMP-1 Expansion Board provides owners of various 6502-based systems with access to a growing variety of compact, low-cost APPLE expansion products (excluding products requiring dynamic RAM refresh, APPLE monitor, or language card features). To provide these features, ELCOMP offers this system as an extensively documented bare board.
Available for use with the ELCOMP-1 Expansion Chassis, ELCOMP offers the following cards:

1. Model 604 Prototyping card for custom circuit development
2. Model 605 6522 via I/O Experiment Card. Provides 6522, convenient 1K static RAM for user I/O routines and a large prototyping area.
3. Model 607 2716 EPROM Programmer. Includes complete software, schematics, descriptions, etc. Ideal for programming single supply 2716 type EPROMs.

4. Model 609 8K EPROM Card. Provides sockets for four 2K 2716 single supply EPROMs. Provision for software/hardware bank switching. Ideal for conservation of ROM address space, and for software-selectable multiple monitor ROMs in OSI machines.
5. Model 610 12-bit A/D Converter Board. Uses low cost 12-bit A/D converter and precision reference. Includes software.
6. Model 608 software, schematic, and description for use of General Instrument AY3-8912 Sound Effects chip with the ELCOMP Model 605 I/O Board.

Each of the cards is available as a bare board plus complete documentation. Software, where applicable, is provided for the APPLE-II and Ohio Scientific systems. Software for other 6502 systems is forthcoming.

APPLE OWNERS! The above cards may be used with the ELCOMP-1 Expansion or inserted directly into your APPLE.

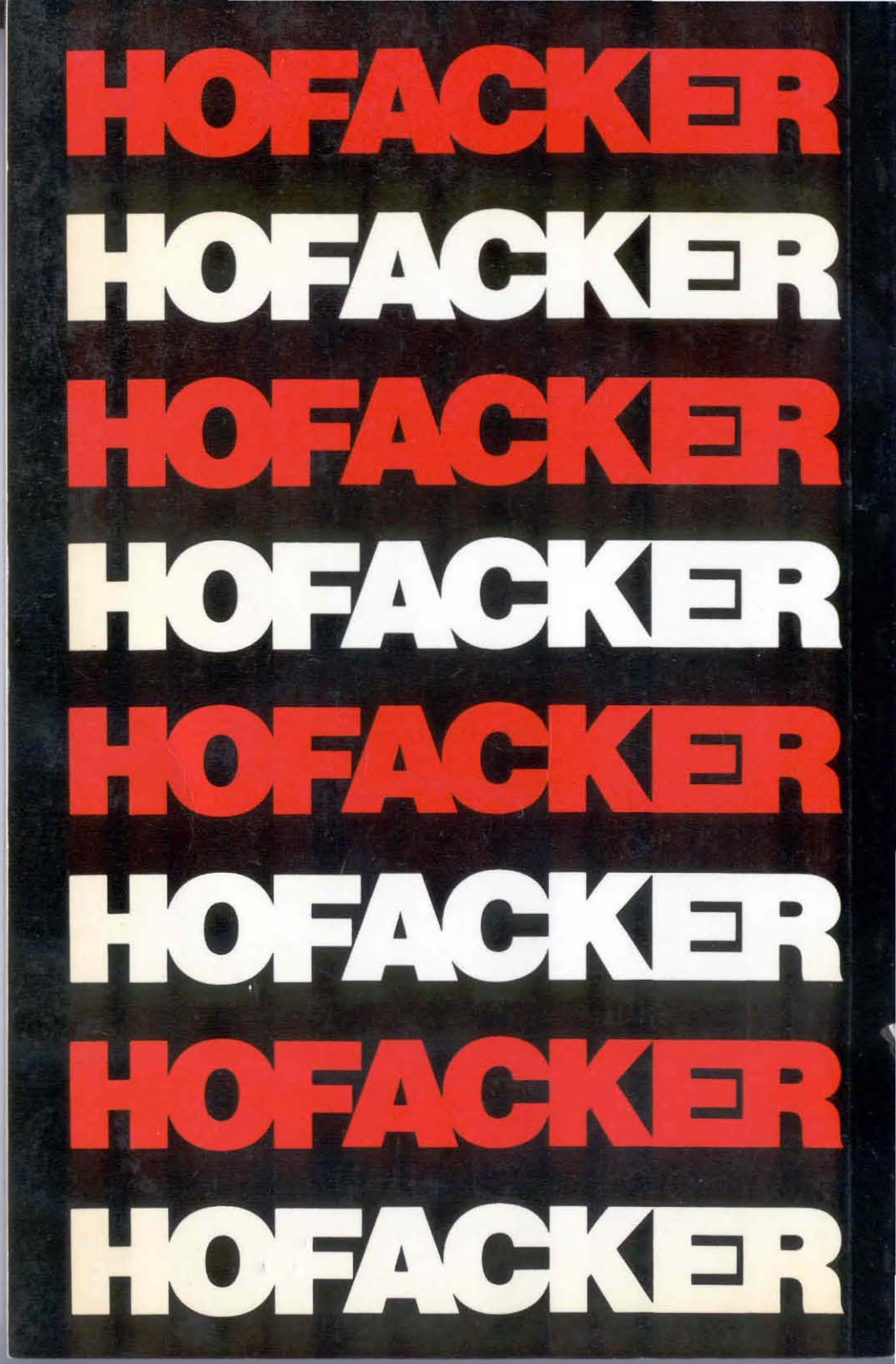
OSI OWNERS! ELCOMP-1 is ideal for Superboard expansion via 40 conductor ribbon. The S-44 slot is ideal for use with 32K dynamic RAM Board. (Available from Beta Computer Devices, Orange, California).

FUTURE PRODUCTS: S-44 8K Static RAM Disk Controller.

Pricing information:

Order-No.	Description	Price
Model 606	ELCOMP-1 Expansion Board	$49.00
Model 604	Prototyping Card	$29.00
Model 605	6522 VIA I/O Experiment Card	$39.00
Model 607	2716 EPROM Programmer	$49.00
Model 609	8K EPROM Card (2716)	$29.00
Model 610	12 Bit A/D Converter Board	$74.00
Model 608	Sound with the GI AY-3-8912	$39.00

Prices are subject to change without notice.


HOFACKER
HOFACKER
HOFACKER
HOFACKER
HOFACKER
HOFACKER
HOFACKER
HOFACKER
HOFACKER