BUILD A SYNERGY CARD FOR YOUR PC

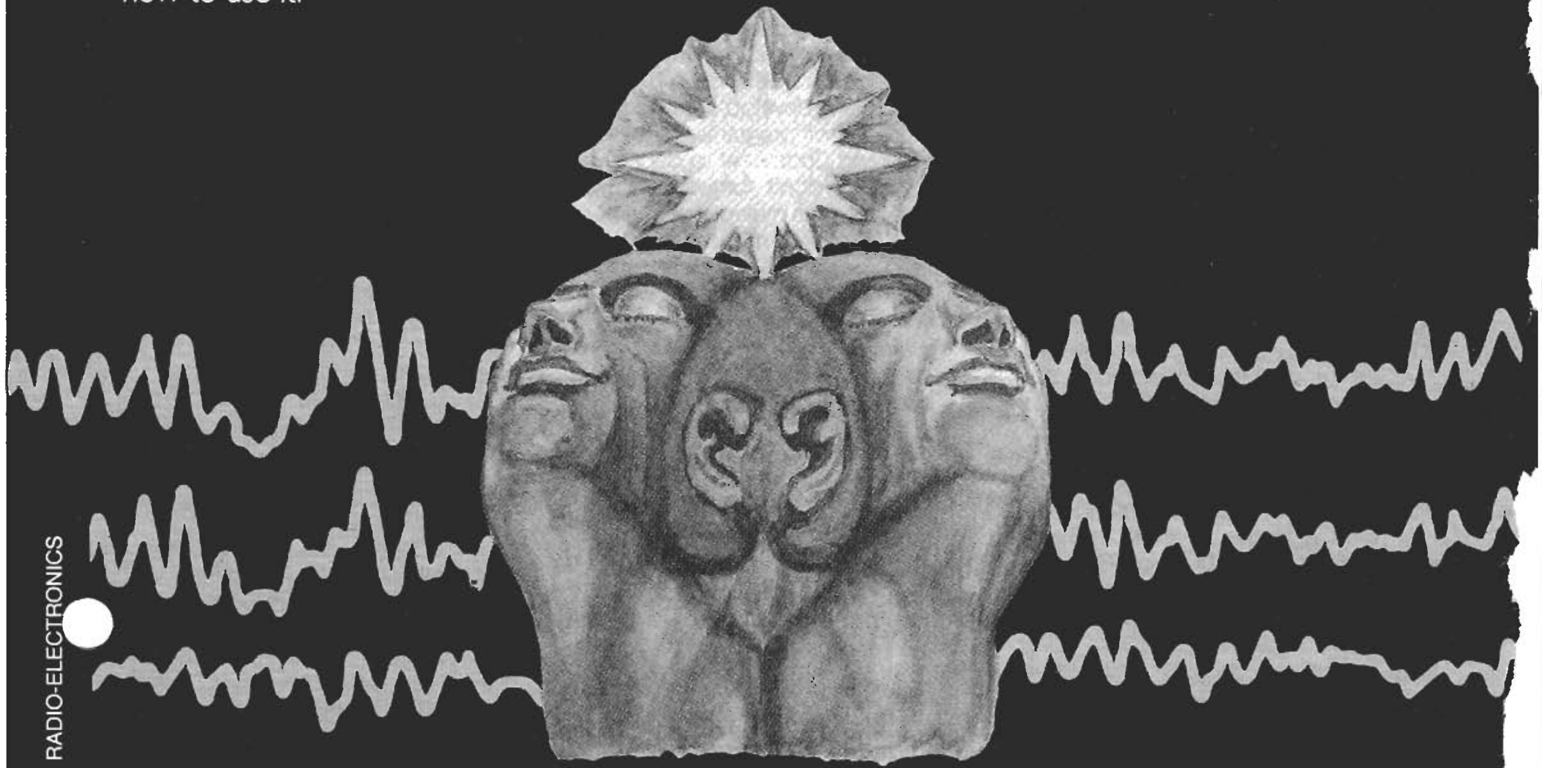
Tap your hidden potential!

Jeff Wiley & R. D. Warner

In our society, we tend to over-emphasize analytical, linear thinking, and under-emphasize intuitive, symbolic approaches. However, we short-change ourselves when we do. The best scientific achievements have something "artistic" about them, and the best artistic achievements have something "scientific" (organized or structured) about them. Systems design and analysis, for example, requires one to retain an image of the overall system while simultaneously being able to break that system down into its component parts.

To help you learn to maximize both analytical and intuitive types of skills, we're going to describe the construction of a device, called the Synergy Card, that plugs into any IBM PC or compatible with an eight-bit slot. The Synergy Card works by generating computer-controlled sound waves that put your brain into a state that is more conducive to both learning and creating. The Synergy Card can also be used to create various sound effects. The card can be built for under $100; assembled and tested versions are also available. See the sidebar for more information.

Because of the controversial nature of the technique, we'll spend most of this first article describing basic terms, background research, and philosophy. Next time we'll build the card itself and show you how to use it.



Background

Research has taught us much about how the human brain works. Using machines such as the *E*lectro-*E*ncephalo*G*raph (EEG) and the *S*uperconducting *QU*antum *I*nterference *D*evice (SQUID), researchers have discovered that there is a division of labor between the sides of the brain. The left hemisphere is predominantly analytical, linear, and verbal. By contrast, the right hemisphere is intuitive, creative, visual, and symbolic. Throughout the course of the day, we shift from one hemisphere to the other, depending on the task at hand. When speaking or balancing a checkbook, the highest brainwave amplitudes are in the left side. When painting or listening to music, the right side becomes most active.

Another discovery revealed that certain psychological and physiological states were associated with certain ranges of brain waves. That discovery led to the development of bio-feedback techniques. With the proper techniques, a person could learn to produce alpha or theta brainwaves, and thereby experience the physical and mental states associated with them.

A technique that, unlike biofeedback, requires no learning by the subject has been developed by Robert Monroe, presently of *T*he *M*onroe *I*nstitute (TMI) in Faber, Virginia. In the late 1950's, Monroe began investigating methods of accelerated learning, which led to remarkable discoveries about the nature of human consciousness. For example, Monroe found that regular sound waves at brainwave frequencies could induce the brain to produce brainwaves of a similar frequency. That phenomenon is called a *F*requency-*F*ollowing *R*esponse (FFR). The important thing about FFR is that, given the correct frequencies, one could enjoy the physical and mental states associated with those frequencies.

Because most brainwave frequencies are quite low, "beat" frequencies are used to generate them. The way they're generated is unusual, though, because the actual beat frequencies are synthesized by the brain instead of being heard normally. To understand how it works, imagine listening through headphones to two distinct signals. A 200-Hz signal is fed to one side, and a 205-Hz signal to the other. Those signals are not being mixed electrically, nor are they mixing in the air. Nevertheless, you will hear a distinct 5-Hz beat frequency, which is created in your brain. The way it works is that the two brain hemispheres synchronize to synthesize the beat signal. That process was dubbed *Hemi-Sync* (for hemisphere synchroniztion) by Monroe. We'll call it HS.

In that example, if you were being monitored with an EEG, it would trace a 5-Hz brainwave, and the amplitudes of the signals from both hemispheres would gradually become equal.

Which signals produce desirable physiological and psychological responses? TMI was founded in 1971 to answer just that question. Thousands of experiments with volunteers led to the development of certain combinations to produce specific responses. For example, a frequency of about 4 Hz can produce theta waves, which are associated with deep relaxation and high receptivity. By itself, theta waves may produce drowsiness, but when combined with beta waves (which are associated with concentration and alert problem solving), you have an excellent mental state for learning, retention, and problem solving.

ORDERING INFORMATION

The following are available from Perceptual Research Ventures, P.O. Box 201516, Missoula, MT 59801: Etched, drilled, tin-plated, and silk-screened PC board (PR-10), $36.00; assembled, tested, and conformal-coated card (PR-48), $319.95 (for research organizations only); custom cabling (PR-8), $28.95; Sleep Lab software, compiled, runs card as a background task, leaving CPU free for other work, (PR-100), $29.95. Add $5 for postage and handling.

Case studies

A philosophy professor at the Tacoma Community College in Washington has been using HS signals combined with music in his classes since 1978. The professor spends less time on housekeeping and department tasks; students are more relaxed, cooperative, and interested in learning; and student performance is enhanced.

In another study, HS (delta, theta, and beta) was mixed with music and played in 24 second- through fifth-grade classrooms. Teachers reported that students had more highly focused attention, were more interested in learning, and were able to learn more material in a given period of time. The general classroom atmosphere was one of relaxed attentiveness.

Dream research

HS also looks useful for dream research. In fact, HS was developed mainly as a way to maintain a state conductive to lucid dreaming, so that dreaming could be studied in a more consistent manner. That fascinating phenomenon has potential as a psychological and creativity tool, and as entertainment. In fact, HS has been used to develop techniques that can teach you how to increase dream awareness, and how to dream with finesse.

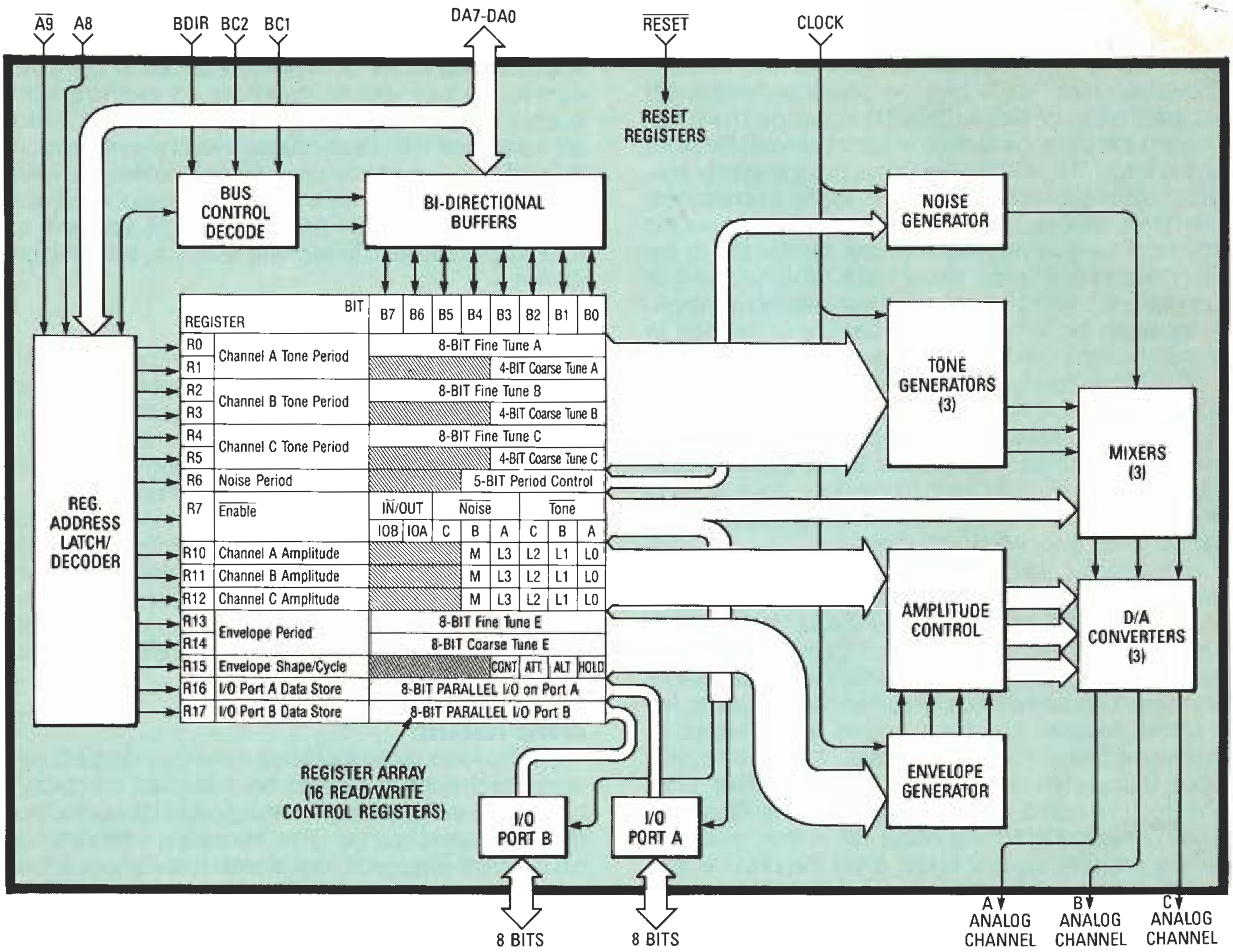
Sleep research has shown that in REM sleep (the period when most dreams occur) the right hemisphere is much more active than the left. That's understandable based on what we said before, because dreams are usually pictorial, emotional, and illogical, and the right side of the brain is the home of such states.

In order to become lucid in a dream, it is necessary to increase activity in the left hemisphere. The effect of hemisphere synchronization seems to be that it allows one to carry more left-brain critical-analysis functions into dreaming. Much of what we've said about HS and dreaming is speculative, but the authors can personally attest that HS helps induce lucid dreaming.

Hardware introduction

The Synergy Card is built around two 8255A parallel interfaces, three AY-3-8910 *P*rogrammable *S*ound *G*enerators (PSG's), and a stereo amplifier suitable for driving headphones or the line-level inputs of your stereo system.

Because sound generation is done with hardware rather than software, the control software is easy to write. The hardware approach also provides much greater control over the sound envelope generated by the Synergy Card. In addition, it also frees the CPU for other tasks. For example, the CPU can give the card the necessary parameters, and then go off and monitor biofeedback equipment, altering the signal mix based on the information provided by that equipment.



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	$\overline{\text{Enable}}$	$\overline{\text{IN}}$/OUT		$\overline{\text{Noise}}$			$\overline{\text{Tone}}$		
		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							

FIG. 1—THE PROGRAMMABLE SOUND GENERATOR has sixteen registers that control the operation of a noise generator, tone generators, and the frequency and amplitude of the resulting waveforms.

Figure 1 shows a block diagram of the PSG. As you can see, each PSG has three analog outputs and 16 digital I/O lines. Each of the three analog outputs has its own tone generator; the output of a single noise generator (if enabled) may be mixed with one or more outputs. The amplitude of each output may be fixed or variable, depending on the state of the envelope generator. The output of each mixer is fed through its own four-bit (sixteen-level) D/A convertor. The analog outputs of the IC have maximum amplitudes of 1 volt pk-pk.

All operational parameters are specified by a set of sixteen registers. Output frequency, for example, is determined by the contents of registers R13 and R14, which are the fine and coarse envelope period control registers. Here is the formula for determining output frequency:

$$F_O = F_C/[16 \times (256 \times C_T + F_T)]$$

F_O is the desired frequency, F_C is the clock frequency, C_T is the contents of the coarse-tune register, and F_T is the contents of the fine-tune register.

Control software must be able to calculate C_T and F_T, given a desired output frequency (F_O). (Clock frequency F_C is set at 1.8432 MHz.) For example, to obtain a 200-Hz tone, plug in the known values and re-arrange the equation as follows:

$$200 \text{ Hz} = 1843200 \text{ Hz}/[16 \times (256 \times C_T + F_T)]$$

$$200 \text{ Hz} = 1843200 \text{ Hz}/(4096C_T + 16 F_T)$$

$$4096C_T + 16F_T = 1843200 \text{ Hz}/200 \text{ Hz}$$

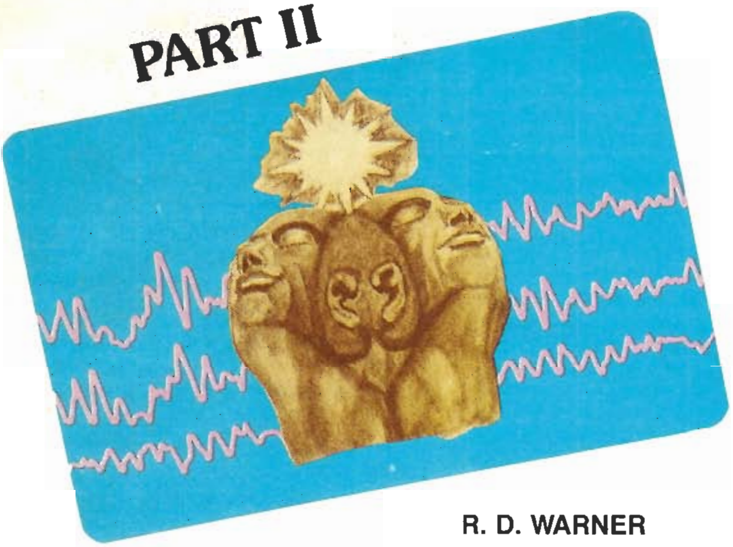
$$4096C_T + 16F_T = 9216$$

$$C_T = 9216/4096 = 2$$

$$F_T = 1024/16 = 64$$

C_T is calculated first, and then F_T is determined from the remainder. The reason is that the PSG views the eight-bit fine-tune register and the four-bit coarse-tune register as representing one twelve-bit number. So you really aren't solving the equation for two unknowns, but one that is broken down into most-significant and least-significant parts that fit into the two registers.

Next time we'll present the complete circuit, discuss how it works and how to build it.



BUILD A SYNERGY CARD FOR YOUR PC

This month let's build the Synergy Card.

R. D. WARNER

Last month we discussed the concepts of hemisphere synchronization (HS) and frequency-following responses (FFR) in the brain. Those terms refer to a technology that uses audio signals to influence brain function. This time we'll show you how to build an expansion card for IBM PC's and compatibles. The Synergy Card can generate sound affects and musical notes, as well as HS signals. In addition, the card has a number of digital I/O lines that can interface your PC to bio-monitoring equipment. With those capabilities, the Synergy Card may be used for research into the affects of sound waves on psychological states.

Circuit overview

First let's look at the block diagram shown in Fig. 1. The address decoder (IC1–IC3) determines where in the microprocessor's address space the Synergy Card will respond. The card is built around two standard parallel ports; the ports consist of two 8255's (IC5 and IC6), each of which has three bi-directional 8-bit ports. The 8255 is commonly used in PC's and compatibles.

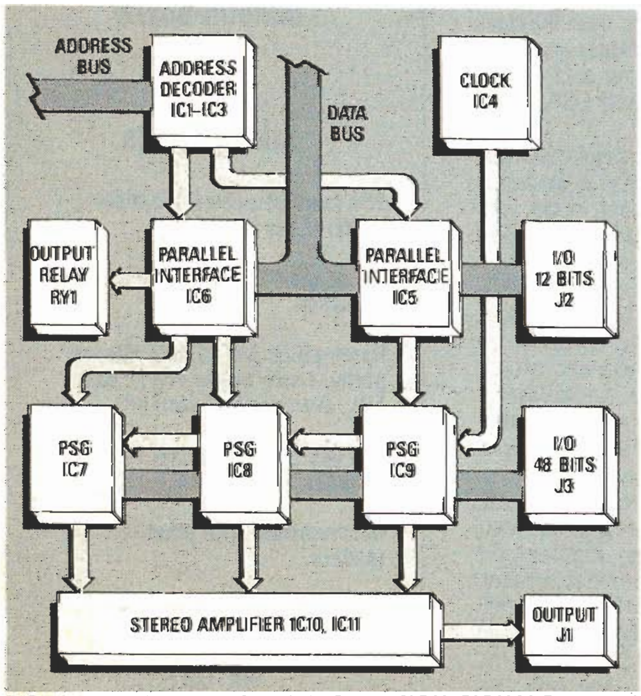


FIG. 1—HEMISPHERE-SYNCHRONIZATION SIGNALS, sound effects, and multi-part music may be synthesized with the inexpensive circuit outlined here.

TABLE 1: Memory MAP

Address	Device	Port
D000:FFF8	PPI 0	Port A - PSG 0
D000:FFF9	PPI 0	Port B - PSG 1
D000:FFFA	PPI 0	Port C - PSG 0,1 Cntl
D000:FFFB	PPI 0	Control Register
D000:FFFC	PPI 1	Port A -IO
D000:FFFD	PPI 1	Port B - PSG 2
D000:FFFE	PPI 1	Port C - IO, PSG 2 Cntl
D000:FFFF	PPI 1	Control Register

The 8255's drive three Programmable Sound Generators (PSG's, IC7–IC9), which we introduced last time. One bit of IC6 also drives an output relay, which may be used to control the motor of an external cassette recorder. The remaining I/O lines of IC5 are available at J2 for external use. The analog outputs of the PSG's drive a stereo amplifier, and the 48 digital outputs available from the three IC's are not used at all by the Synergy Card. Those outputs aren't brought to a connector on the PC board, but instead, for experimental purposes, we soldered a ribbon cable to the appropriate pins on the foil side of the PC board.

Unusually, the Synergy Card's I/O ports are not decoded in the CPU's I/O address space, but in main memory where programs run and data is stored. This is known as memory-mapped I/O, a technique usually reserved for "6" family processors (68000, 6502, etc.) Memory-mapped I/O is used because Intel supplies few machine-language instructions for manipulating I/O ports, but many for manipulating main memory. The memory instructions could be simulated using the simple I/O instructions, but speed requirements preclude such simulation.

As shown in Table 1, the Synergy Card's I/O ports are mapped to the eight memory addresses ranging from

D000:FFF8 to D000:FFFF. Address lines 5–7 are not decoded, so the circuit actually responds to addresses in the range D000:FF18 through D000:FFFF. In binary this appears:
1101 1111 1111 XXX1 1000 to
1101 1111 1111 XXX1 1111.
Because A5–A7 are not decoded, each register appears every eight locations (18, 20, 28, etc.). However, our software always addresses the F8–FF offsets. Also, because the Synergy Card is memory-mapped, make sure that no other card (especially EMS cards and network adapters) installed in your PC uses memory in the D000 segment. Otherwise the resulting address contention may well crash your machine.

Each of the Programmable Peripheral Interfaces (PPI's, IC5 and IC6) has 4 registers: one each for the three ports (A, B, C), and a control register for setting operating mode, so that's why eight addresses are required.

The complete schematic is shown in Fig. 2; let's discuss the address decoder first. As shown, IC1 and IC2 enable IC3 when an address in the range specified above is accessed. Assuming IC3's B and C (A3 and A4) inputs are high, the $\overline{Y6}$ or $\overline{Y7}$ output goes low, depending on the state of the A (A2) input. When A2 is high, IC5 is selected; when A2 is low, IC6 is selected. The low-order address lines (A0 and A1) then address the desired register in the selected 8255.

The 8255 has many modes of operation; the desired mode is set by placing highs on A0 and A1, and then writing a value to the control register. (Consult Intel's *Microsystem Components Handbook*, Volume II, for more information on the 8255.) In our software, the PPI's are programmed in Mode 0, so all pins function as outputs. To use any of the I/O pins available at J2 as inputs, you'll have to program a different mode. We'll discuss programming in detail next month.

In Mode 0, a value written to one of the 8255's ports (A, B, or C) will be latched on the corresponding output pins, where an attached PSG can then read the signal at its more leisurely pace. Port C of each PPI may be divided into two independent four-bit ports, and that is done here, for purposes of driving the control lines of the PSG's.

PARTS LIST

All resistors are ¼-watt, 5% unless otherwise noted.
R1–R3—1000 Ohms
R4, R5—500 Ohms
R6, R7—68,000 Ohms
R8, R9—15,000 Ohms
R10, R11—10 Ohms
R12, R13—10,000 Ohms Trimmer Potentiometer
R14, R15—27,000 Ohms
Capacitors
C1, C2—330 pF, mica
C3, C4—2.2 µF, 15 volts, tantalum
C5, C6—0.01 µF, ceramic disc
C7, C8—10 µF, 15 volts, tantalum
C9, C10—0.1 µF, 15 volts, tantalum
C11, C12—0.047 µF, ceramic disc
C13—22 µF, 15 volts, tantalum
C14–C18—0.1 µF, 15 volts, tantalum
C19, C20—220 µF, axial, 15 volts, electrolytic
Semiconductors
IC1—74LS30N, 8-input NAND gate
IC2—74LS10P, triple 3-input NAND gate
IC3—74LS138, 3-to-8 decoder
IC4—1.8432-MHz, clock
IC5, IC6—8255A-5, programmable peripheral interface
IC7–IC9—AY-3-8910A, programmable sound generator
IC10, IC11—LM386N-1, audio amplifier
Q1—2N3904, NPN transistor
Other components
J1—9-pin "D" connector, PC mount
J2—15-pin "D" connector, PC mount
RY1—5-volt SPST reed relay (Radio Shack)
Miscellaneous
Metalized hood for 9- and 15-pin connectors.
Shielded plugs and cables for stereo hookup.
3/32" plug for cassette remote jack.
Note: The following are available from Perceptual Research Ventures, P.O. Box 20151, Missoula, MT 59801: Etched, drilled, tin-plated, and silk-screened PC board (PR-10), $36.00; assembled, tested, and coated card (PR-48), $319.95; custom cabling (PR-8), $28.95; Sleep Lab software, compiled, runs card as a background task, leaving CPU free for other work, (PR-100), $25. Unfortunately, due to FCC regulations, the assembled and tested unit may be sold only to qualified research institutions. All orders add $5 for postage and handling.

TABLE 2: PSG Control Registers

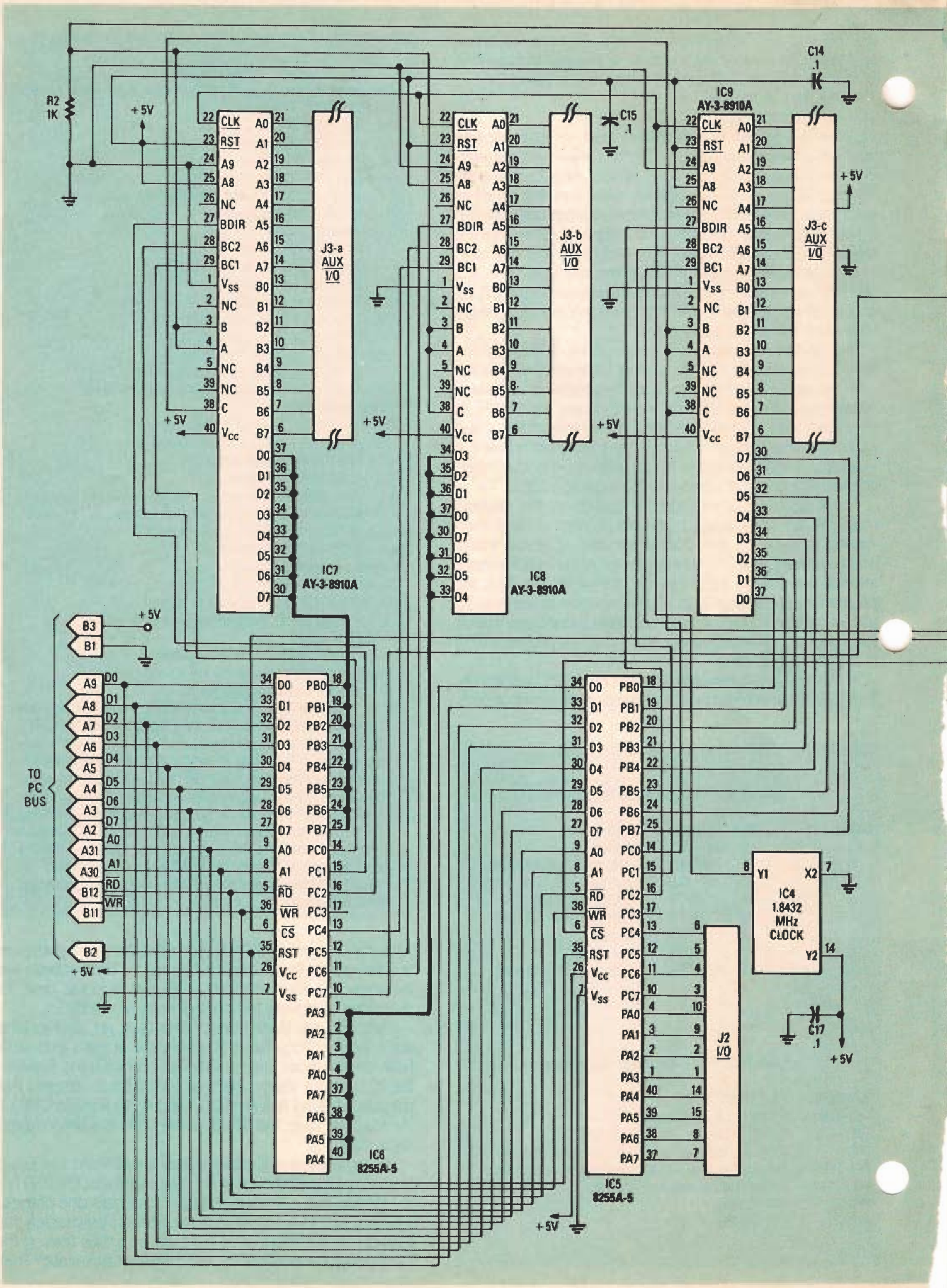
BDIR	BC2	BC1	Name	Description
0	0	0	NACT	Inactive
0	0	1	ADAR	Latch register address
0	1	0	IAB	Inactive, data lines high-Z
0	1	1	DTB	Read from PSG
1	0	0	BAR	Latch register address
1	0	1	DW	Inactive
1	1	0	DWS	Write to PSG
1	1	1	INTAK	Latch register address

TABLE 3: PSG Register functions

Registers	Function
R0–R5	Program tone periods
R6	Program noise period
R7	Enable noise and/or tone on selected channel
R8–R10	Select fixed or envelope-variable amplitude
R11–R13	Envelope period and shape
R14	I/O Port A
R15	I/O Port B

Each PSG has three control lines (BDIR, BC1, and BC2) that tell it what to do. Table 2 shows the actions that result from driving those lines in various combinations. Basically, the IC has four states: Inactive (000), Latch Register Address (111), Read Register (011), and Write Register (110). In the inactive state, the IC's data lines are in a high-impedance state.

To understand the other states, recall from the block diagram of the IC shown in Fig. 1 last time that the PSG has 16 registers that control operation. To access one of those registers, first its address must be latched by placing that address on the PSG's D0–D7 inputs, and then forcing the three control lines high (i.e., entering the Latch state). Then


R2
1K
+5V
C14
.1
C15
.1
C17
.1
IC7
AY-3-8910A
IC8
AY-3-8910A
IC9
AY-3-8910A
CLK
RST
A9
A8
NC
BDIR
BC2
BC1
Vss
B
A
C
Vcc
A0
A1
A2
A3
A4
A5
A6
A7
B0
B1
B2
B3
B4
B5
B6
B7
D0
D1
D2
D3
D4
D5
D6
D7
J3-a
AUX
I/O
J3-b
AUX
I/O
J3-c
AUX
I/O
TO
PC
BUS
B3
B1
A9
A8
A7
A6
A5
A4
A3
A2
A31
A30
B12
B11
B2
RD
WR
CS
RST
PB0
PB1
PB2
PB3
PB4
PB5
PB6
PB7
PC0
PC1
PC2
PC3
PC4
PC5
PC6
PC7
PA0
PA1
PA2
PA3
PA4
PA5
PA6
PA7
IC6
8255A-5
IC5
8255A-5
J2
I/O
Y1
X2
Y2
IC4
1.8432
MHz
CLOCK

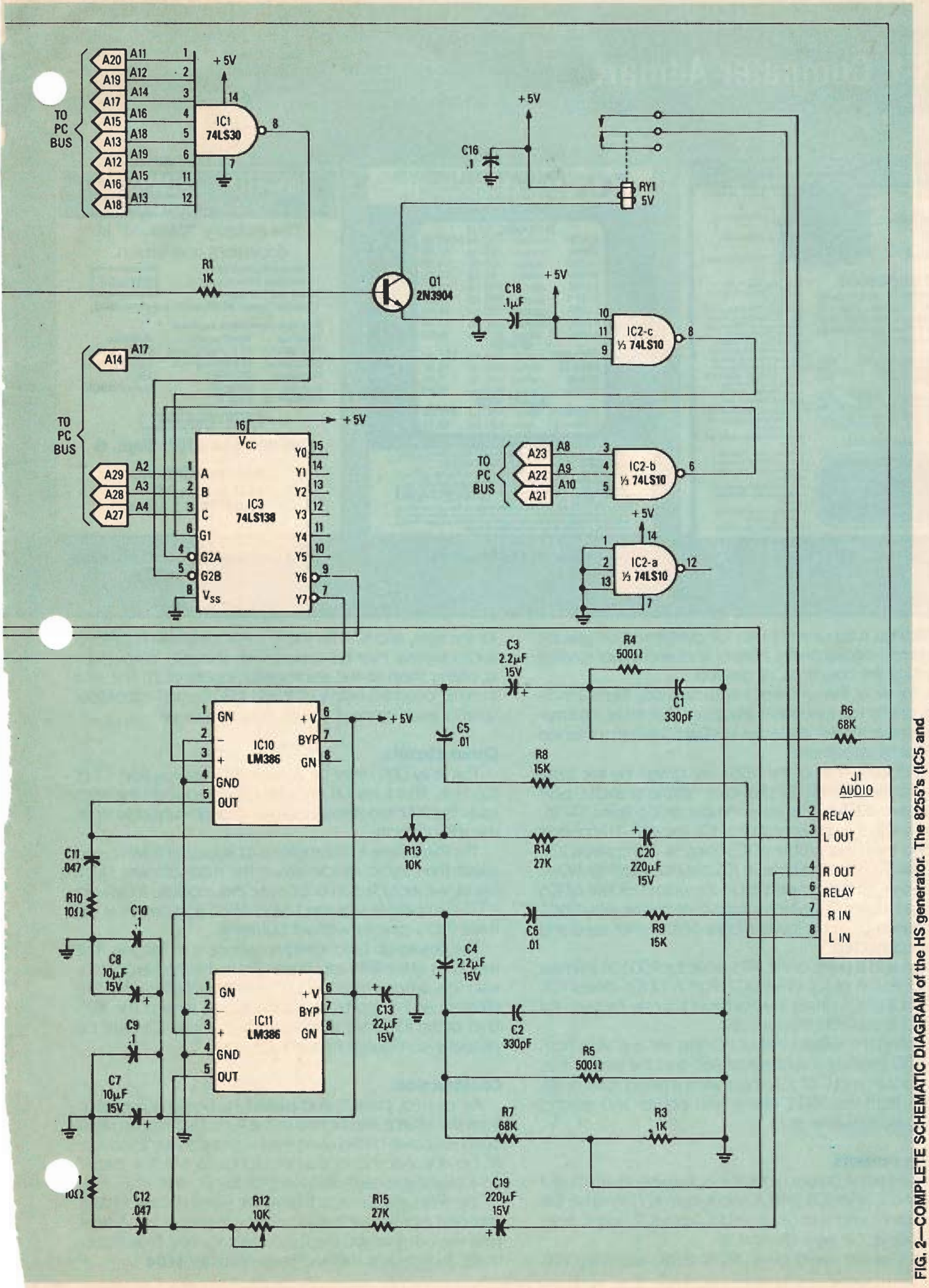


FIG. 2—COMPLETE SCHEMATIC DIAGRAM of the HS generator. The 8255's (IC5 and IC6) are memory-mapped, not I/O-mapped; they control the three programmable sound generators (IC7–IC9).

the desired read or write may be performed by placing the correct signals on the control lines and either reading or writing the data lines, as desired.

As for what the registers themselves do, Table 3 indicates briefly the functional groupings. For more information, consult the *Microelectronics Data Catalog* published by General Instrument.

The control lines of the PSG's are driven by the three least-significant bits (LSB's) of each nibble of each C port of the two 8255's. The lower nibble of IC6 (pins 14–16) controls IC7, the upper nibble of IC6 (pins 13–11) controls IC8, and the lower nibble of IC5 (pins 14–16) controls IC9. The "extra" (upper) nibble in IC5 may be used for I/O; it appears at J2. The unused bit in the upper nibble of IC6 drives transistor Q1, which in turn controls the relay (RY1). The unused bit in the lower nibble of IC6 is not used and is not connected.

The A and B ports of the PPI's drive the PSG's in a similar manner. Port B of IC6 drives IC7; Port A of IC6 drives IC8, and Port B of IC5 drives IC9. Port A of IC5 may be used for I/O, and is available through J2.

Counting the sixteen bits of I/O that are available from each PSG (making a subtotal of 48), and the twelve bits from Ports A and C of IC5, that makes a grand total of 60. All bits from the PSG's, along with power and ground signals, are available at J3.

Analog outputs

Looking at the output of the PSG's, five voices are mixed (A, B, and C from IC8, and A and B from IC7) to drive the left channel, and four (A, B, and C from IC9, and C from IC7) to drive the right channel.

Each channel has its own LM386 audio amplifier; IC11 for the right, and IC10 for the left. Also, external high-level audio signals may be mixed with the PSG outputs by applying them to the appropriate inputs of J1. The two trimmer potentiometers (R12 and R13) control the output level of each channel. Set each to mid-range.

Other circuits

The relay (RY1) may be activated by driving Port C7 of IC6 high. That turns Q1 on, which then activates the reed relay. The PPI cannot source or sink enough current to drive the relay directly.

The PSG's have a maximum clock speed of 2-MHz, and rather than try to divide down the motherboard clock signal, we included an onboard clock module. It delivers a TTL-compatible signal at 1.8432-MHz, and can drive the three PSG's directly without buffering.

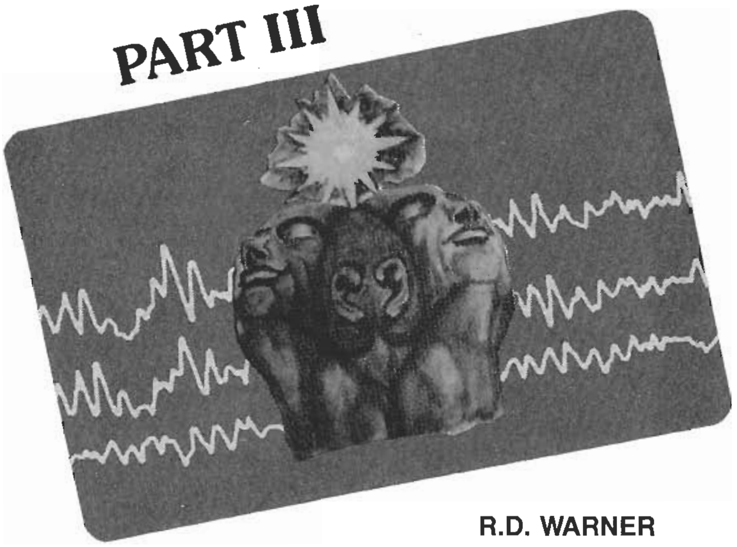
The power-up (and reset) sequence is as follows. The RESET lines of the PPI's are connected to the motherboard's RESET line. When a reset occurs, the PPI control registers are cleared, and all ports are configured as inputs. The $\overline{\text{RESET}}$ lines of the PSG's are simply tied high; those IC's must be properly configured through software.

Construction

An etched, drilled, and plated PC board is available from the source mentioned in the Parts List. You can also build your own board using the foil patterns we'll show in PC Service. Another option would be to wire the circuit on a prototype card designed for the PC bus.

The only problem is, is that you're going to have to wait until next month to actually build the Synergy card. At that time we will give you the foil pattern and the Parts Placement, and discuss the necessary software.

See "Digital Telephone Lock" PCB for Layout



BUILD A SYNERGY CARD FOR YOUR PC

Wrapping up the Synergy Card

R.D. WARNER

This is the last of three articles discussing the theory, construction, and use of a Synergy Card, a device used for generating sound effects and influencing brain function. In Part 1 we discussed the theory of a technique that uses audio signals to influence brain function. The goal of that technique is to achieve *H*emisphere *S*ynchronization (HS) by means of *F*requency-*F*ollowing *R*esponse (FFR). It is theorized that hemisphere synchronization creates a mental atmosphere that allows for better concentration and more creativity.

In the second part we described the circuitry of a card that plugs into any IBM PC compatible, and allows you to experiment with HS, FFR, and to generate sound effects and multi-voice music. The card also has a number of digital I/O lines that can interface your PC to bio-monitoring or to other equipment.

This time, we'll show you how to build, test and operate the Synergy Card. Included are software listings in BASIC, assembler, and DEBUG scripts. Those programs may be used as-is for testing purposes, or may be used as models and expanded for more-complex usage. All listings are also available on RE-BBS (516-293-2283, 300/1200 baud, 8 data bits, no parity, 1 stop bit); just download the file HEMISYNC.ARC.

If you build your own card, you'll have to deal with the absence of plated-through holes. That means that you'll have to solder many components on both sides of the board, install feed-throughs, etc. If you choose that method, be very careful, and check your work several times.

Use the parts-placement diagram shown in Fig. 3 to mount all components. Note that the IC's are installed in various orientations, so double-check to make sure that none is installed backward.

Regardless of construction method, when you're ready to stuff the board, first install all decoupling capacitors (C13–C18, with C16 mounted on the solder side), and C19 and C20. When you install C19, note its proximity to trimmer resistor R12. Now connect a DMM across V_{CC} and ground to check for shorts.

Now do the audio section, mounting R14 and R15 on the solder side of the board. Some of the discrete components are very close together, so be careful and check your work several times.

Next, install J1, J2, and the mounting bracket. The mounting tabs of the bracket go on the component side of the board. There should be enough play in the mounting holes so the bracket lines up with the connectors; if

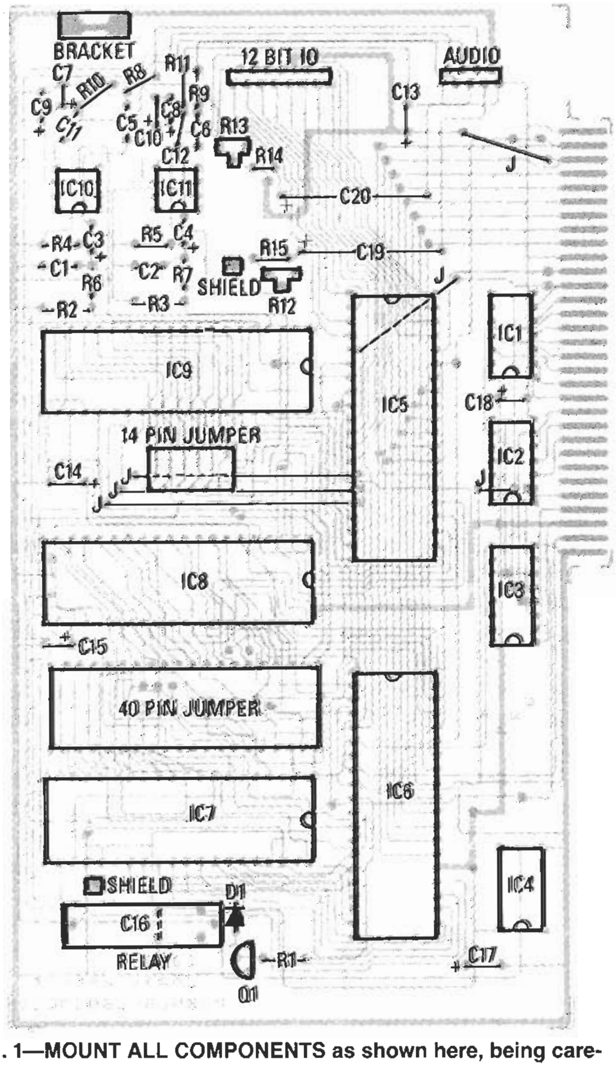


FIG. 1—MOUNT ALL COMPONENTS as shown here, being careful in the audio section, where close component mounting could lead to shorts.

TABLE 1—LOGIC LEVELS

	PSG		PPI	
Pin	Before	After	Before	After
1	LOW		VAR, HI-Z	LOW
2	N.C.		VAR, HI-Z	LOW
3	LOW		VAR, HI-Z	LOW
4	LOW		VAR, HI-Z	LOW
5	N.C.		PULSE	
6	HIGH		HIGH	
7	HIGH		LOW	
8	HIGH		PULSE	
9	HIGH		PULSE	
10	HIGH		LOW, HI-Z	LOW
11	HIGH		HI-Z	LOW
12	HIGH		HI-Z	LOW
13	HIGH		HI-Z	LOW
14	HIGH		HI-Z	LOW
15	HIGH		HI-Z	LOW
16	HIGH		HI-Z	LOW
17	HIGH		HI-Z	LOW
18	HIGH		VAR	LOW
19	HIGH		VAR	LOW
20	HIGH		VAR	LOW
21	HIGH		VAR	LOW
22	PULSE		VAR	LOW
23	HIGH		VAR	LOW
24	LOW		VAR	LOW
25	HIGH		VAR	LOW
26	VARIES		HIGH	
27	HI-Z	LOW	PULSE	
28	HI-Z	LOW	PULSE	
29	HI-Z	LOW	PULSE	
30	VARIES	LOW	PULSE	
31	VARIES	LOW	PULSE	
32	VARIES	LOW	PULSE	
33	VARIES	LOW	PULSE	
34	VARIES	LOW	PULSE	
35	VARIES	LOW	LOW	
36	VARIES	LOW	PULSE	
37	VARIES	LOW	VAR, HI-Z	LOW
38	LOW		VAR, HI-Z	LOW
39	VAR		VAR, HI-Z	LOW
40	HIGH		VAR, HI-Z	LOW

not, you can drill the mounting holes out slightly. Don't drill them too much, though, or you'll damage the traces. The card was designed using a bracket supplied by Vector; others may not align correctly. Install the 40- and 14-pin jumper header sockets, and then all wire jumpers.

Let's do a little more checking before continuing. Again, make sure that V_{CC} and ground are not shorted. The forward-biased resistance across a decoupling capacitor should be around 35 ohms. Also, make sure that none of the edge-card fingers is shorted to a neighbor. Correct any problems before proceeding.

Then wire up an audio cable that connects J2 to the appropriate inputs of your stereo system. The LM386's can drive headphones directly; doing so may be most convenient for testing.

With the computer turned off, insert the card in a slot, and connect the card to the Tape or Aux. jacks of your stereo. Make sure the card is aligned in the slot properly. Then turn your stereo on with the volume low. Now turn the computer on and apply an audio signal to the junction of R2 and R6. You should hear the tone through the left speaker. Then apply the the tone to the junction of R3 and R7: This time you should hear the tone over the right speaker.

Remove the card from your PC (after powering it down, of course), and correct any problems. Then solder in the clock (IC4), Q1, R1, and RY1. The clock has a dot on one end that indicates pin 1. A transistor with in-line pins (not pin-circle) should be used for Q1. Otherwise, the collector and emitter will be reversed. Now mount IC1, IC2, and IC3, followed by IC5 and IC6.

Testing

After resolving any problems, install the card in your PC and boot up. Then use a logic probe to check the pins of the *P*rogrammable *P*eripheral *I*nterfaces (PPI's) and the *P*rogrammable *S*ound *G*enerators (PSG's). Your readings should match those in Table 1. The "Before" and "After" columns in that table refer to the states of the respective pins before and after the initialization program is run.

Here are a few notes on why various readings are obtained. Before initialization, PPI I/O ports are in a high-impedance state. So, for example, Port A of PPI 1 (IC5, pins

LISTING 1

```
C>debug                                 ;Enter Debug
-A                                      ;Go into assemble mode

XXXX:0100 MOV     BX,D000               ;Set segment to D000,
XXXX:0103 MOV     DS,BX                 ; which is card location.
XXXX:0105 MOV     DI,FFF8
XXXX:0108 MOV     BH,80                 ;Set PPI ports to OUTPUT -
XXXX:010A MOV     [DI+03],BH            ; first PPI 0,
XXXX:010D MOV     [DI+07],BH            ; then PPI 1.
XXXX:0110 MOV     [DI+02],BL            ;Write zero to all ports, the
XXXX:0112 MOV     [DI+01],BL            ; PSG control reg's first so
XXXX:0115 MOV     [DI],BL               ; that PSG's are in INACT state.
XXXX:0118 MOV     [DI+06],BL
XXXX:011B MOV     [DI+05],BL
XXXX:012E MOV     [DI+04],BL
XXXX:0121 INT     20                    ;Return to DOS
XXXX:0123                               ;Hit return to quit assembly
-RBX
BX 0000                                 ;Make register BX set to zero
:0
-RCX
CX 0000                                 ;Set CX to program length
:23                                     ; in hex bytes
-N B:SCINIT.COM                         ;Give the program a COM name
-W                                      ;Write it to disk
Writing 0023 bytes
-Q                                      ;Quit

C>
```

LISTING 2

```
10 DEF SEG = &H5000                          'This loads the
procedure
20 FOR I = 256! TO 535!                      ' outside of BASIC -
30 READ IMBED%                               ' set 10 to where you
like
40 POKE I,IMBED%                             ' but leave offset
alone.
50 NEXT I
55 '--------------------------------------------------------------
60 DATA &H50,&H53,&H51,&H52,&H1E
70 DATA &H57,&H56,&HB8,&H00,&HD0,&H8E,&HD8,&HB7,&H80,&HBF,&HFB
75 DATA &HFF,&H88,&H3D,&HBF,&HFF,&HFF,&H88,&H3D,&H32,&HFF,&HBF
80 DATA &HFA,&HFF,&H88,&H3D,&HBF,&HFE,&HFF,&H88,&H3D,&HEB,&H06
90 DATA &H90,&H00,&H01,&H00,&H00,&H00,&H2E,&H8B,&H1E,&H2A,&H01
100 DATA &H80,&HFF,&H30,&H7C,&H2D,&H80,&HFF,&H30,&H74,&H08,&H80
110 DATA &HFF,&H31,&H74,&H13,&HE9,&HCD,&H00,&H2E,&HC6,&H06,&H29
115 DATA &H01,&H80,&HBF,&HFA,&HFF,&HB3,&H80,&H88,&H1D,&HE9,&HBD
120 DATA &H00,&H2E,&HC6,&H06,&H29,&H01,&H00,&HBF,&HFA,&HFF,&H32
125 DATA &HDB,&H88,&H1D,&HE9,&HAD,&H00,&H80,&HFF,&H20,&H7C,&H09
130 DATA &H80,&HEF,&H20,&HBF,&H02,&H00,&HEB,&H11,&H90,&H80,&HFF
140 DATA &H10,&H7C,&H09,&H80,&HEF,&H10,&HBF,&H01,&H00,&HEB,&H03
150 DATA &H90,&H33,&HFF,&H53,&H83,&HFF,&H02,&H7C,&H60,&HBF,&HFD
155 DATA &HFF,&HBE,&HFE,&HFF,&H59,&H2E,&H8A,&H3E,&H29,&H01,&H80
160 DATA &HFF,&H80,&H74,&H06,&H33,&HDB,&H53,&HEB,&H04,&H90,&H32
170 DATA &HDB,&H53,&H2E,&H8A,&H16,&H28,&H01,&HB8,&H06,&H00,&HF6
180 DATA &HE2,&H2E,&H02,&H06,&H27,&H01,&H50,&H53,&H2E,&H8A,&H16
185 DATA &H28,&H01,&HB8,&H07,&H00,&HF6,&HE2,&H2E,&H02,&H06,&H27
190 DATA &H01,&H8A,&HFD,&H88,&H3D,&H8A,&HD8,&H88,&H1C,&H5B,&H88
200 DATA &H3C,&H8A,&HD9,&H88,&H1D,&H5B,&H88,&H1C,&H5B,&H88,&H3C
210 DATA &HB3,&H01,&H2E,&H88,&H1E,&H28,&H01,&H32,&HDB,&H2E,&H88
220 DATA &H1E,&H27,&H01,&HEB,&H2A,&H90,&HBE,&HFA,&HFF,&H2E,&H8A
230 DATA &H1E,&H29,&H01,&H80,&HFB,&H80,&H75,&H05,&H2E,&H88,&H1E
235 DATA &H27,&H01,&H83,&HFF,&H00,&H7F,&H0C,&HB3,&H10,&H2E,&H88
240 DATA &H1E,&H28,&H01,&HBF,&HF8,&HFF,&HEB,&H83,&HBF,&HF9,&HFF
245 DATA &HE9,&H7D,&HFF,&H5E,&H5F,&H1F,&H5A,&H59,&H5B,&H58,&HCB
248 '--------------------------------------------------------------
250 INPUT "ENTER REGISTER NUMBER";R1%
260      IF R1% = 99 THEN GOTO 330           'Enter "99" to exit
prog
270      INPUT "ENTER DATA TO WRITE";D1%
280      POKE 298!,D1%                       'Set segment to match
290      POKE 299!,R1%                       ' line 10, but offset
300      OFFST= 256!                         ' must equal 256 or
310      DEF SEG = &H5000                    ' hex 100.
315        CALL OFFST
320 GOTO 250
330      END
```

LISTING 1

```
C>debug                       ;Enter Debug
-A                            ;Go into assemble mode

XXXX:0100 MOV    BX,D000 !    ;Set segment to D000,
XXXX:0103 MOV    DS,BX        ; which is card location.
XXXX:0105 MOV    DI,FFF8
XXXX:0108 MOV    BH,80        ;Set PPI ports to OUTPUT -
XXXX:010A MOV    [DI+03],BH   ; first PPI 0,
XXXX:010D MOV    [DI+07],BH   ; then PPI 1.
XXXX:0110 MOV    [DI+02],BL   ;Write zero to all ports, the
XXXX:0112 MOV    [DI+01],BL   ; PSG control reg's first so
XXXX:0115 MOV    [DI],BL      ; that PSG's are in INACT state.
XXXX:0118 MOV    [DI+06],BL
XXXX:011B MOV    [DI+05],BL
XXXX:012E MOV    [DI+04],BL
XXXX:0121 INT    20           ;Return to DOS
XXXX:0123                     ;Hit return to quit assembly
-RBX
BX 0000                       ;Make register BX set to zero
:0
-RCX
CX 0000                       ;Set CX to program length
:23                           ; in hex bytes
-N B:SCINIT.COM               ;Give the program a COM name
-W                            ;Write it to disk
Writing 0023 bytes
-Q                            ;Quit

C>
```

LISTING 2

```
10 DEF SEG = &H5000                     'This loads the
procedure
20 FOR I = 256! TO 535!                 ' outside of BASIC -
30 READ IMBED%                          ' set 10 to where you
like
40 POKE I,IMBED%                        ' but leave offset
alone.
50 NEXT I
55 '------------------------------------------------------------
60 DATA &H50,&H53,&H51,&H52,&H1E
70 DATA &H57,&H56,&HB8,&H00,&HD0,&H8E,&HD8,&HB7,&H80,&HBF,&HFB
75 DATA &HFF,&H88,&H3D,&HBF,&HFF,&HFF,&H88,&H3D,&H32,&HFF,&HBF
80 DATA &HFA,&HFF,&H88,&H3D,&HBF,&HFE,&HFF,&H88,&H3D,&HEB,&H06
90 DATA &H90,&H00,&H01,&H00,&H00,&H00,&H2E,&H8B,&H1E,&H2A,&H01
100 DATA &H80,&HFF,&H30,&H7C,&H2D,&H80,&HFF,&H30,&H74,&H08,&H80
110 DATA &HFF,&H31,&H74,&H13,&HE9,&HCD,&H00,&H2E,&HC6,&H06,&H29
115 DATA &H01,&H80,&HBF,&HFA,&HFF,&HB3,&H80,&H88,&H1D,&HE9,&HBD
120 DATA &H00,&H2E,&HC6,&H06,&H29,&H01,&H00,&HBF,&HFA,&HFF,&H32
125 DATA &HDB,&H88,&H1D,&HE9,&HAD,&H00,&H80,&HFF,&H20,&H7C,&H09
130 DATA &H80,&HEF,&H20,&HBF,&H02,&H00,&HEB,&H11,&H90,&H80,&HFF
140 DATA &H10,&H7C,&H09,&H80,&HEF,&H10,&HBF,&H01,&H00,&HEB,&H03
150 DATA &H90,&H33,&HFF,&H53,&H83,&HFF,&H02,&H7C,&H60,&HBF,&HFD
155 DATA &HFF,&HBE,&HFE,&HFF,&H59,&H2E,&H8A,&H3E,&H29,&H01,&H80
160 DATA &HFF,&H80,&H74,&H06,&H33,&HDB,&H53,&HEB,&H04,&H90,&H32
170 DATA &HDB,&H53,&H2E,&H8A,&H16,&H28,&H01,&HB8,&H06,&H00,&HF6
180 DATA &HE2,&H2E,&H02,&H06,&H27,&H01,&H50,&H53,&H2E,&H8A,&H16
185 DATA &H28,&H01,&HB8,&H07,&H00,&HF6,&HE2,&H2E,&H02,&H06,&H27
190 DATA &H01,&H8A,&HFD,&H88,&H3D,&H8A,&HD8,&H88,&H1C,&H5B,&H88
200 DATA &H3C,&H8A,&HD9,&H88,&H1D,&H5B,&H88,&H1C,&H5B,&H88,&H3C
210 DATA &HB3,&H01,&H2E,&H88,&H1E,&H28,&H01,&H32,&HDB,&H2E,&H88
220 DATA &H1E,&H27,&H01,&HEB,&H2A,&H90,&HBE,&HFA,&HFF,&H2E,&H8A
230 DATA &H1E,&H29,&H01,&H80,&HFB,&H80,&H75,&H05,&H2E,&H88,&H1E
235 DATA &H27,&H01,&H83,&HFF,&H00,&H7F,&H0C,&HB3,&H10,&H2E,&H88
240 DATA &H1E,&H28,&H01,&HBF,&HF8,&HFF,&HEB,&H83,&HBF,&HF9,&HFF
245 DATA &HE9,&H7D,&HFF,&H5E,&H5F,&H1F,&H5A,&H59,&H5B,&H58,&HCB
248 '------------------------------------------------------------
250 INPUT "ENTER REGISTER NUMBER";R1%
260     IF R1% = 99 THEN GOTO 330       'Enter "99" to exit
prog
270     INPUT "ENTER DATA TO WRITE";D1%
280     POKE 298!,D1%                   'Set segment to match
290     POKE 299!,R1%                   ' line 10, but offset
300     OFFST= 256!                     ' must equal 256 or
310     DEF SEG = &H5000                ' hex 100.
315       CALL OFFST
320 GOTO 250
330     END
```

LISTING 3

```
XXXX:0100 MOV     BX,D000       ;Set segment to D000,
XXXX:0103 MOV     DS,BX         ; which is card location.
XXXX:0105 MOV     DI,FFF8
XXXX:0108 MOV     BH,80         ;Set PPI ports to OUTPUT -
XXXX:010A MOV     [DI+03],BH    ; first PPI 0,
XXXX:010D MOV     [DI+07],BH    ; then PPI 1.
XXXX:0110 MOV     [DI+02],BL    ;Write zero to non-I/O ports, the
XXXX:0112 MOV     [DI+01],BL    ; PSG control reg's first so
XXXX:0115 MOV     [DI],BL       ; that PSG's are in INACT state.
XXXX:0118 MOV     AX,F0FF       ;AH sends INACT to PSG 2, and
XXXX:011B MOV     [DI+06],AH    ; makes 4 I/O bits HIGH.
XXXX:011E MOV     [DI+05],BL    ;Writes zero to PSG 2 data lines
XXXX:0121 MOV     [DI+04],AL    ;AL makes 8 I/O bits go HIGH
XXXX:0124 INT     20            ;Return to DOS
```

1–4 and 37–40) should show a high impedance on your logic probe before running the initialization program. Port A of the other PPI behaves the same way.

The B ports (pins 18–25) of both PPI's show varied readings, because they're connected to the data lines of the PSG's, and a PSG does disconcerting but harmless things unless the INACT (inactive) command (000) is present on its control lines (pins 27–29). PSG pins 6–21 are for the two I/O ports, and appear high when in the input mode.

Now let's discuss the initialization program, SCINIT.COM, shown in Listing 1. What it does is to set all PPI ports to outputs, and then loads a zero into each one, driving all port lines low. The lows on the control lines of the PSG's constitute the INACT command, so the PSG's calm down. PSG pins 26 and 39 will still vary, because they are test pins whose function varies depending on which model of the AY-3-8910 IC you have.

Now load and run SCINIT.COM using DEBUG. Enter the assembly-language instructions and debug commands as shown, but don't enter the semicolons or the comments following them. Ignore the *X*'s shown in the first column. After running the program, you'll notice that the data and control lines of the PSG's are now all low.

That initialization program tests the steering logic, but to really check the individual data and I/O lines requires the use of a larger assembly-language procedure, which, unfortunately, we don't have enough space to publish. You can however, download it from the RE-BBS. If you assemble the program, make sure that the last byte in the object file generated is 0CBh. The author found that when using his macro assembler, it is necessary to go in manually and change that byte; otherwise he couldn't get the procedure to return properly to the BASIC program shown next. The last line of the source code may be a dummy, because no matter what's put in there, it doesn't compile right. For some reason, it insists on executing an intra-segment return (0C3h), instead of an inter-segment return.

The assembly-language procedure is also embedded in the BASIC program shown in Listing 2. That program allows you to write any byte-size value to any one of the Synergy Card's 50 registers. Remember, each PSG has 16 registers; registers 0–47 are used to set and reset the PSG lines, and registers 48 and 49 are dummies used to turn the relay on and off (which in turn is used to control the motor of a cassette recorder).

For example, if you write a 2 to any of the registers ·32–47, line D1 of PSG2 will go high. Or by writing a 1 to any of the registers 16–31, D0 of PSG1 will go high.

By forcing one bit high at a time, you can thereby check not only the PSG's, but the PPI's as well. If a PSG line doesn't go high, check the corresponding PPI line. If it went high, there's probably an open somewhere between the two. It doesn't take long to go through and check each line that way, and it can save a lot of headaches later.

TABLE 2—MUSICAL NOTES

Note	Octave	Ideal	Note	Octave	Ideal
C	1	32.703	C	5	523.248
C#	1	34.648	C#	5	554.368
D	1	36.708	D	5	587.328
D#	1	38.891	D#	5	622.256
E	1	41.203	E	5	659.248
F	1	43.654	F	5	698.464
F#	1	46.249	F#	5	730.984
G	1	48.999	G	5	783.984
G#	1	51.913	G#	5	830.608
A	1	55.000	A	5	880.000
A#	1	58.270	A#	5	932.320
B	1	61.735	B	5	987.760
C	2	65.406	C	6	1046.496
C#	2	69.296	C#	6	1108.736
D	2	73.416	D	6	1174.656
D#	2	77.782	D#	6	1244.512
E	2	82.406	E	6	1318.496
F	2	87.308	F	6	1396.928
F#	2	92.498	F#	6	1479.968
G	2	97.998	G	6	1567.968
G#	2	103.826	G#	6	1661.216
A	2	110.000	A	6	1760.000
A#	2	116.540	A#	6	1864.640
B	2	123.470	B	6	1975.520
C	3	130.812	C	7	2092.992
C#	3	138.592	C#	7	2217.472
D	3	146.832	D	7	2349.312
D#	3	155.564	D#	7	2489.024
E	3	164.812	E	7	2636.992
F	3	174.616	F	7	2793.856
F#	3	184.996	F#	7	2959.936
G	3	195.996	G	7	3135.936
G#	3	207.652	G#	7	3322.432
A	3	220.000	A	7	3520.000
A#	3	233.080	A#	7	3729.280
B	3	246.940	B	7	3951.040
C	4	261.624	C	8	4185.984
C#	4	277.184	C#	8	4434.944
D	4	293.664	D	8	4698.624
D#	4	311.128	D#	8	4978.048
E	4	329.624	E	8	5273.984
F	4	349.232	F	8	5587.712
F#	4	369.992	F#	8	5919.872
G	4	391.992	G	8	6271.872
G#	4	415.304	G#	8	6644.864
A	4	440.000	A	8	7040.000
A#	4	466.160	A#	8	7458.560
B	4	493.880	B	8	7902.080

You can also test the relay using that program: If you write to register 48, the relay will close, and if you write to register 49 it will open. It doesn't matter what data you write to the dummy registers.

How do you test the PSG I/O lines? You can use the same routine, because the I/O registers are accessed just like the fourteen sound-control registers. Remember that register 7 of each PSG controls the data direction of its two I/O ports, using bits 6 and 7 (for ports A and B, respectively). When one of those bits is high, the corresponding port is an output port, and when it's low, the port is an input port.

So, to test the output ports first you must write a decimal 192 to registers 7, 23, and 39. Then, once again, simply make each bit go high one at a time. For example, by writing 192 to register 39, and then 128 to register 47, bit B7 (the second port) of PSG2 will go high.

LISTING 3

```
XXXX:0100 MOV     BX,D000        ;Set segment to D000,
XXXX:0103 MOV     DS,BX          ; which is card location.
XXXX:0105 MOV     DI,FFF8
XXXX:0108 MOV     BH,80          ;Set PPI ports to OUTPUT -
XXXX:010A MOV     [DI+03],BH     ; first PPI 0,
XXXX:010D MOV     [DI+07],BH     ; then PPI 1.
XXXX:0110 MOV     [DI+02],BL     ;Write zero to non-I/O ports, the
XXXX:0112 MOV     [DI+01],BL     ; PSG control reg's first so
XXXX:0115 MOV     [DI],BL        ; that PSG's are in INACT state.
XXXX:0118 MOV     AX,F0FF        ;AH sends INACT to PSG 2, and
XXXX:011B MOV     [DI+06],AH     ; makes 4 I/O bits HIGH.
XXXX:011E MOV     [DI+05],BL     ;Writes zero to PSG 2 data lines
XXXX:0121 MOV     [DI+04],AL     ;AL makes 8 I/O bits go HIGH
XXXX:0124 INT     20             ;Return to DOS
```

TABLE 2—MUSICAL NOTES

Note	Octave	Ideal	Note	Octave	Ideal
C	1	32.703	C	5	523.248
C#	1	34.648	C#	5	554.368
D	1	36.708	D	5	587.328
D#	1	38.891	D#	5	622.256
E	1	41.203	E	5	659.248
F	1	43.654	F	5	698.464
F#	1	46.249	F#	5	730.984
G	1	48.999	G	5	783.984
G#	1	51.913	G#	5	830.608
A	1	55.000	A	5	880.000
A#	1	58.270	A#	5	932.320
B	1	61.735	B	5	987.760
C	2	65.406	C	6	1046.496
C#	2	69.296	C#	6	1108.736
D	2	73.416	D	6	1174.656
D#	2	77.782	D#	6	1244.512
E	2	82.406	E	6	1318.496
F	2	87.308	F	6	1396.928
F#	2	92.498	F#	6	1479.968
G	2	97.998	G	6	1567.968
G#	2	103.826	G#	6	1661.216
A	2	110.000	A	6	1760.000
A#	2	116.540	A#	6	1864.640
B	2	123.470	B	6	1975.520
C	3	130.812	C	7	2092.992
C#	3	138.592	C#	7	2217.472
D	3	146.832	D	7	2349.312
D#	3	155.564	D#	7	2489.024
E	3	164.812	E	7	2636.992
F	3	174.616	F	7	2793.856
F#	3	184.996	F#	7	2959.936
G	3	195.996	G	7	3135.936
G#	3	207.652	G#	7	3322.432
A	3	220.000	A	7	3520.000
A#	3	233.080	A#	7	3729.280
B	3	246.940	B	7	3951.040
C	4	261.624	C	8	4185.984
C#	4	277.184	C#	8	4434.944
D	4	293.664	D	8	4698.624
D#	4	311.128	D#	8	4978.048
E	4	329.624	E	8	5273.984
F	4	349.232	F	8	5587.712
F#	4	369.992	F#	8	5919.872
G	4	391.992	G	8	6271.872
G#	4	415.304	G#	8	6644.864
A	4	440.000	A	8	7040.000
A#	4	466.160	A#	8	7458.560
B	4	493.880	B	8	7902.080

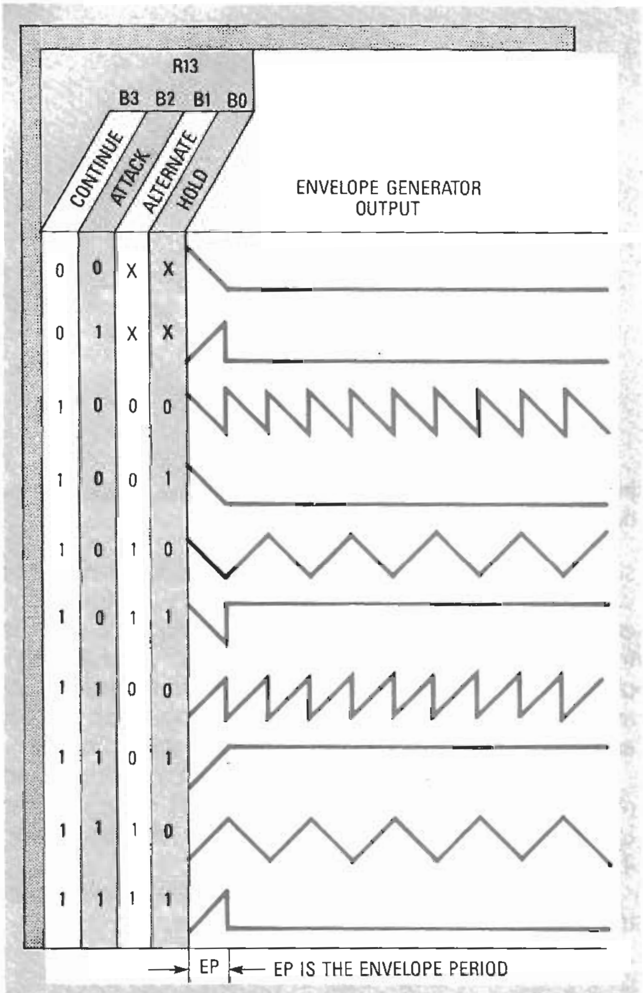


FIG. 2—THE CONTENTS OF PSG REGISTER 13 determines the shape of the envelope.

To test the twelve bits of I/O coming straight from IC5, you need to use a procedure similar to that used in SCINIT.COM. Remember that Port A and the top 4 bits of Port C supply that I/O, so all you have to do is set the PPI to output mode, and then write to the ports. Listing 3 shows a DEBUG procedure that will set all 12 bits high.

That completes the digital testing. Track down and correct any problems before proceeding.

Make beautiful (?) music

Install the Synergy Card in your PC, and connect it to your audio system as described last time. Turn on the computer and make sure that it boots. Then turn on the audio system with the volume low, and run the initialization program.

To begin, let's make a single layer (one tone pair) of Synergistic Sound. Since one tone will go to each channel, write to PSG1. In it, voices A and B go to the left channel, and C goes to the right. (That way we only have to enable the tones in one PSG.) To enable A and C, and disable noise in all three channels, write a decimal 58 to register 23.

Next write the appropriate values into the coarse- and fine-tune registers to get the desired frequency. For this example we'll create tones of 200 and 202 Hz, which provide a synergistic sound frequency of 2 Hz, a nice delta brain-wave frequency.

PARTS LIST

All resistors are ¼-watt, 5% unless otherwise noted.
R1–R3—1000 Ohms
R4, R5—500 Ohms
R6, R7—68,000 Ohms
R8, R9—15,000 Ohms
R10, R11—10 Ohms
R12, R13—10,000 Ohms Trimmer Potentiometer
R14, R15—27,000 Ohms

Capacitors
C1, C2—330 pF, mica
C3, C4—2.2 μF, 15 volts, tantalum
C5, C6—0.01 μF, ceramic disc
C7, C8—10 μF, 15 volts, tantalum
C9, C10—0.1 μF, 15 volts, tantalum
C11, C12—0.047 μF, ceramic disc
C13—22 μF, 15 volts, tantalum
C14–C18—0.1 μF, 15 volts, tantalum
C19, C20—220 μF, axial, 15 volts, electrolytic

Semiconductors
IC1—74LS30N, 8-input NAND gate
IC2—74LS10P, triple 3-input NAND gate
IC3—74LS138, 3-to-8 decoder
IC4—1.8432-MHz, clock
IC5, IC6—8255A-5, programmable peripheral interface
IC7–IC9—AY-3-8910A, programmable sound generator
IC10, IC11—LM386N-1, audio amplifier
Q1—2N3904, NPN transistor

Other components
J1—9-pin "D" connector, PC mount
J2—15-pin "D" connector, PC mount
RY1—5-volt SPST reed relay (Radio Shack)

Miscellaneous
Metalized hood for 9- and 15-pin connectors.
Shielded plugs and cables for stereo hookup.
3/32" plug for cassette remote jack.

Note: The following are available from Perceptual Research Ventures, P.O. Box 20151, Missoula, MT 59801: Etched, drilled, tin-plated, and silk-screened PC board (PR-10), $36.00; assembled, tested, and coated card (PR-48), $319.95; custom cabling (PR-8), $28.95; Sleep Lab software, compiled, runs card as a background task, leaving CPU free for other work, (PR-100), $25. Unfortunately, due to FCC regulations, the assembled and tested unit may be sold only to qualified research institutions. All orders add $5 for postage and handling.

We calculated the register values for the 200-Hz carrier frequency last month: CT = 2, and FT = 64. Using the same technique for 202 Hz, we get CT = 2, and FT = 58. Working backward, those values will actually yield a frequency of 202.1053 Hz (assuming the clock frequency is exact). If FT were 59, we would get a frequency of 201.7513 Hz, so the first value yields the most accurate results.

To generate those tones, write a 64 to register 16, 2 to register 17, 58 to register 20, and 2 to register 21. Now just turn up the volume. Increase the on-board volume to maximum by writing a 15 to register 24 and register 26. Writing a 16 to those registers would switch from fixed-amplitude to envelope.

Table 2 shows the ideal frequency for eight octaves of notes ranging from C1 to B8. You'll find that the card is capable of getting within 0.01 Hz of the ideal frequency for low notes, and within about 88.0 Hz at the upper end, due to nonlinear distribution.

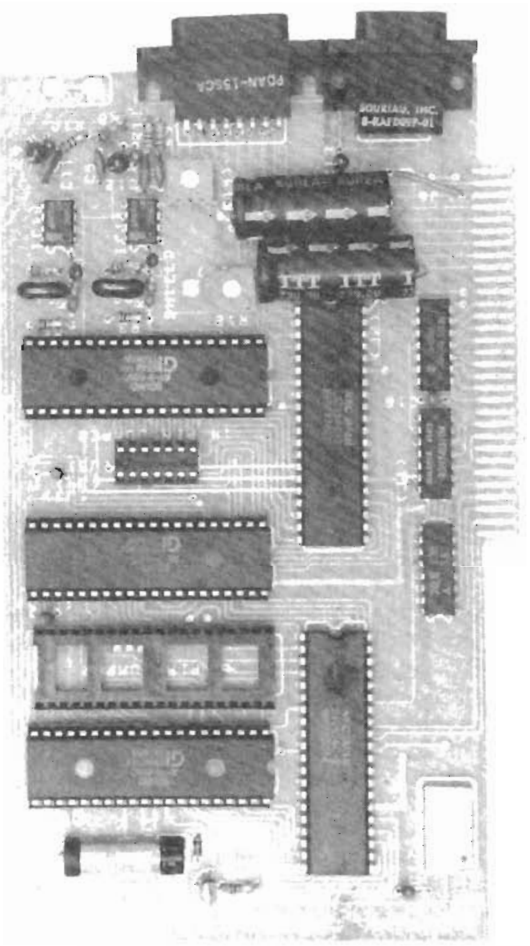


FIG. 3—THE COMPLETED BOARD. Note that IC sockets are used for IC7–IC9, and for the jumper headers. The jumper headers, if desired, can provide 48 bits of I/O in addition to the 12 bits available at J2.

Now let's see how to use amplitude envelopes and noise generators. This demonstration provides a mix that sounds like ocean surf. You can turn off the previous tones by writing a 0 to registers 24 and 26.

Our surf will use a logarithmic, U-shaped envelope of white noise. A 0.10-Hz signal goes to the left channel, and 0.12 Hz to the right. The frequency difference means the phase relationship will change constantly, making the sound realistic.

The first thing we need to do is calculate our coarse- and fine-tune register values for the envelopes. The equation is almost identical to that used for tones, and can be solved in the same way:

$$f_{EN} = f_{CL} / [256 \times (256CT + FT)]$$

In that equation, f_{EN} represents the desired envelope frequency, CT and FT represent the coarse- and fine-tune registers, and f_{CL} represents the clock speed. For proper envelope generation, CT and FT cannot both be zero.

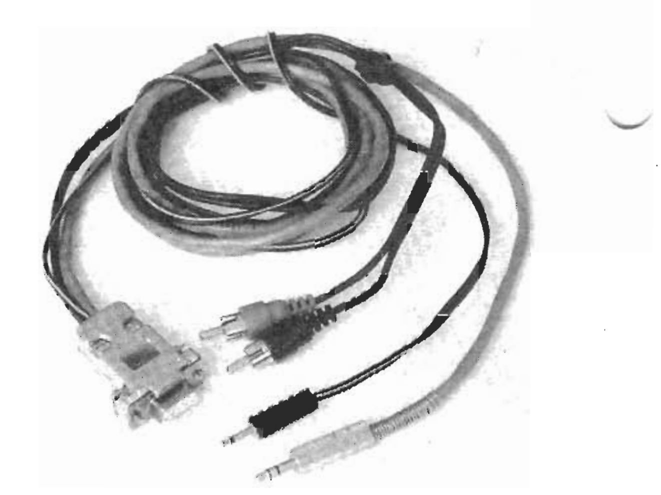


FIG. 4—THE AUTHOR'S CUSTOM CABLE, which is used to connect the Synergy Card to an audio system.

Envelope shapes and corresponding binary codes are shown in Fig. 1. We want a triangle wave (on a log scale) starting at amplitude zero. There is one trick to remember when calculating frequencies for that waveform: The equation is set up for the sawtooth envelopes, and the triangle wave takes twice as much time per wave. So we'll use 0.20 Hz and 0.24 Hz to obtain the correct values. For an f_{EN} of 0.20, CT should be 140 and FT should be 160. For an f_{EN} of 0.24, CT should be 117 and FT should be 48.

Enable Voice A in PSG 0 and PSG 2 by writing 55 to registers 7 and 39. Write the CT and FT values to the appropriate registers. Next write a 14 to registers 13 and 45; that selects our envelope shape.

The noise frequency is governed by the simple equation:

$$f_N = f_{CL} / (16 \times P_N)$$

In that equation, f_N is the desired noise frequency, f_{CL} is the clock speed, and P_N is the value in the *Noise Period* register. NP can range from 1 to 31. For this example, write a 1 to registers 6 and 38. Last, turn on the volume by writing a 16 to registers 8 and 40. Now you should hear a surf-like sound coming from your audio system.

At this time, evaluate the signal level arriving at your stereo. The potentiometers on the Synergy Card control that level; one for each channel. Adjust them for maximum volume, and balance between the two channels.

Conclusion

You'll probably want to experiment with envelope shape(s), volume, number of layers, carrier frequency, and Synergistic Sound frequency. **Warning: Avoid 13–15 Hz beat frequencies, which have been known to cause epileptic seizures in those prone to them.**

Much could be said about how you can use the 60 bits of I/O. For example, to build an inexpensive isolated controller, you could use the Synergy Card to drive transistors, which would in turn drive relays. Or you could drive a DAC, which would drive a VCO, and use it to transmit data over phone lines. Or have it read in-coming digital data. As a reader of **Computer Digest,** you probably have some ideas of your own.

The author wishes to thank Montana Micro for technical assistance, Jameco Electronics for supplies and equipment, and Brady Books for technical references.