

Initial Research for the
Development or Purchase of a
Computerized Synthesizer
For Use as a Composer's Aid

by Scott Vaughan

August 9, 1978

Computer Science 588

Initial Research for the Development or Purchase of a Computerized Synthesizer For Use as a Composer's Aid

The author's primary goal is to begin research leading to the attainment of a low cost computer/music system which will allow the composer to write polyphonic music of up to eight voices into a computer through a terminal, and have the music played back by means of computer synthesized sound or by means of a conventional synthesizer controlled by a computer via digital-to-analog converters.

The system must demand less mental and physical energy than the repeated sight readings often necessary during the creation of a composition. In order to deliver the necessary ease of operation, the system's software must be relatively easy to edit, since compositional processes often require a very large number of minute alterations. The system must also allow the composer to control with relative ease the parameters of pitch, tempo, and volume. Other desired elements are user control of envelope and waveform. In order to be compatible with future music printing applications, the software must be defined linearly, at least within the confines of the measure, or in the case of music not using traditional music notation, within the confines of a structural unit similar in score length to the measure.

The goal system will allow the composer to retreat

and hear his product objectively as the painter steps back to review his canvas.

The search for the goal system began with the discovery of promotional material for the Rowland Micro-Composer, a small specialized computer that allows a composer to enter a score and direct the computer to play the score by emitting up to eight controlled voltages and corresponding triggers to control a standard music synthesizer. An interesting feature of the system is its ability to lay and follow its own click track on a multitrack tape deck. This feature allows the recording of twentyfour computer controlled voices on a conventional tape deck with four tracks and synchronizing capabilities.

For one who is not a composer it may be difficult to fully appreciate a programable music playing machine. As a compositional aid the Rowland Micro Composer far surpasses the piano which cannot sustain notes, has a limited number of voices, cannot sound like a flute section, and is limited to the pianistic skill of its player.

Unfortunately the price, \$4,800.00, which must be added to the price of a large synthesizer, about \$1,200.00, is prohibitive. According to Mark Scovill, an integrated circuit designer for Texas Instruments, the price of the Rowland Micro Composer represents an extremely high markup over the actual cost of the unit's hardware. In light of this information, the development of a

less expensive system or the discovering of a system marketed elsewhere seems sensible.

The next step in this research was to explore the automated music system, AMUS, developed by Dr. Dan W. Scott for use with a computer assisted ear training program being developed by the music theory department at North Texas State University. An outstanding feature of this system is the fact that it is relatively inexpensive, yet it delivers up to seven voices with user control of dynamics, pitch, tempo, rhythm, and even limited control of envelope. For further system description see appendix B. Unfortunately this system would not be usable by all because it is not commercially available, having been designed and built exclusively for use at N.T.S.U.

To further explore the computer/music hardware relevant to the goal system, a letter requesting information about music related computer hardware and software was mailed to numerous companies producing inexpensive computers. The letter and a list of companies to whom it was sent is presented in appendices C and D, respectively. About one month after the mailing a brochure arrived describing a music synthesis board and related hardware produced by a company called Solid State Music operating out of Santa Clara, California.

Solid State Music's system, compatible with any computer using the "Altair Bus", offers one voice per

board and control of pitch, tempo, rhythm, attack and sustain of the envelope, and a simple, high level software which makes simple the writing and editing of the programmed score. For further description see appendix E.

It became evident through this material and a conversation with Mr. Scovill, the integrated circuit designer, that, in most cases, computer music synthesis hardware is less expensive than comparable traditional voltage controlled hardware.

A list of dealers included with the Solid State Music brochure listed the Micro Store in Richardson, a suburb of Dallas, as a retail outlet for Solid State Music's hardware. In a conversation with a Micro Shop salesman who identified himself as Dale, it was learned that the boards produced by the California firm could only be used with a system using an S100 buss, and such a system would cost around \$2,000.00. Since the monophonic boards cost about \$150.00 apiece, the cost for a system using this hardware would be prohibitive.

An additional promotional brochure was received from a company named ALF Products, operating out of Denver. Their music synthesis hardware offered significantly better cost efficiency. Their primary synthesis board, which offers control of rhythm tempo, pitch, envelope, and waveform, and is monophonic,

is priced at \$265.00, \$100.00 more than the Solid State Music board. ALF, however, also markets a product they call Quad Chromatic Pitch Generators, a device which offers four independent voices without control of waveform, envelope, and dynamics. These functions, however, can be added by means of accessory boards. Perhaps the most impressive feature of the Quad Chromatic Pitch Generators is their price, which can be as low as \$111.00 depending on what options are purchased. For further information on the ALF systems, see appendix F.

Contrary to the information on the specification sheets in appendix E, the ALF boards have no software yet. This fact was presented in a letter, dated July 21, 1978, from ALF's Philip J. Tubb. This letter is included in appendix F.

A trip was made to the Compu Store, a computer retail outlet near Dallas, for two reasons. First, it was learned that relatively inexpensive computer systems, using an S100 bus and perhaps compatible with the ALF and Solid State Music boards, might be available at a store such as this. Secondly, the people at this store reportedly knew how to get in touch with members of the various "user's groups" in the Dallas area. Some members of the "user's groups" might know of people working on projects related to the goal system.

Bill Powell, a sales person at the Compu Store, demonstrated various musical applications of the Apple II computer, a relatively inexpensive system which does not use an S100 bus but which can be expanded by the addition of various specialized boards. He directed the Apple II to perform a monophonic tune utilizing the small speaker normally used as a bell or signal device on the terminal. The melody was produced by software synthesis. For an example of this software see appendix C. An interesting aspect of the Apple II's performance of the melody was that the musical tones played were "phase shifted", a popular effect used in electronic music. This effect usually requires a specialized piece of equipment called a phase shifter.

Mr. Powell identified "The Apple Core", a "user's group" associated with the Apple II computer, and he mentioned that some of the members were working on musical applications of the computer.

Mr. Powell also told about Craig Boody who is engaged in research closely related to the goal system. Mr. Boody, once associated with a college in Sherman, Texas, recently moved to Minneapolis, and his address there was not known by Mr. Powell.

A few days after the visit to the Compu Store, Mr. Powell called and told of an Apple Program which played music and printed it on a standard matrix

printer.

A catalog of synthesizer modules which included a description of a microprocessor system for controlling standard synthesizer modules, an inexpensive, tempered, and reportedly precise digital-to-analog converter, and other computer/music hardware and software, was received from Paia Electronics located in Oklahoma City. Parts of the catalog are exhibited in appendix H. The Paia catalog advertised their publication, Friendly Stories About Computers/Synthesizers (Design Analysis)

This publication, its authorship uncertain, was ordered and received. It describes how the Paia 8700 computer/controller works in conjunction with a digital keyboard and specialized software to produce polyphonic capability and digital control of the various parameters of a standard voltage controlled synthesizer. The publication also stated that a Paia music synthesis system is currently being developed which will use the Apple II computer. Friendly Stories About Computers/Synthesizers can be read in its entirety in appendix I.

A phone call to Paia rendered the unfortunate news that their microprocessor, at this time, does not have the capacity to play a polyphonic score of more than a few measures.

The next step in the search for the goal system

was a phone call to ALF Products in Denver where Mr. Phil Tubbs, who deals with the music related hardware, said the boards advertized were being developed as part of a computer music system which will have real time performance capabilities as well as programable capabilities. This system will sell for around four thousand dollars. Further inquiry revealed that the boards produced by ALF were being adapted for use by the Apple II computer. Mr. Tubbs also said that Craig Boody, mentioned by Bill Powell at the visit to the Compu Store, is working with the Apple Computer system in his development of a computerized system for playing and writing music, and that he was given this author's name and address when it was received in this author's first letter to ALF.

A call was placed to the main office of Apple Computers in California where it was learned that the computer/music project is being handled by Dr. Windle Sander. A letter was sent to Dr. Sander inquiring about the capabilities and costs of the music synthesis boards and when they will become available.

What transpires in the future toward the attainment of the goal system will be affected by answers to questions concerning the ALF, Paia, and Apple music related systems and systems yet to be discovered. At this point there seem to be five pragmatic possibilities regarding acquisition of the desired system. First,

The Quad Chromatic Pitch Generators might be adapted to operate with an inexpensive computer not using an S100 bus or with a yet-to-be-discovered inexpensive computer which does use an S100 bus. Second, the Apple music synthesis system might be available at an affordable price. The third possibility is that the Paia system may be adaptable to operation with an inexpensive computer. Fourth, the Paia/Apple system may become available at an affordable price. Fifth, additional research may uncover an available goal system.

APPENDIX B

D R A F T

1 MARCH 1978

AMUS CONTROL COMPUTER

DEPARTMENT OF COMPUTER SCIENCES

North Texas State University

by DAN W. SCOTT, Ph.D.

THE JOB OF THE CONTROL COMPUTER IS TO TRANSLATE THE USER'S SCORE, OR "SOURCE" SCORE, FROM A STRING OF ALPHABETIC AND NUMERIC CHARACTERS INTO A LIST OF PARAMETERS CALLED THE PLAY SCORE, OR "OBJECT" SCORE. THE PLAY SCORE IS USED, AFTER ITS GENERATION, BY THE CONTROL COMPUTER'S PERFORMANCE PROGRAM TO PROVIDE THE DETAILED PARAMETER VALUES REQUIRED BY THE MUSOR VOICE CLOCK AND SIGNAL GENERATORS.

THE PLAY SCORE IS STORED IN THE CONTROL COMPUTER'S MEMORY, AND IT IS THE SIZE OF THIS MEMORY AVAILABLE FOR THE PLAY SCORE WHICH DETERMINES THE LENGTH AND COMPLEXITY OF THE SOURCE SCORE WHICH CAN BE TRANSLATED. THE MINIMUM MEMORY REQUIRED FOR THE RESIDENT COMPUTER PROGRAMS IS 5000 BYTES; WITH AN 8000-BYTE MEMORY, THEN, ABOUT 3000 BYTES ARE AVAILABLE FOR THE OBJECT SCORE, AND THIS WOULD BE MORE THAN AMPLE FOR THE EAR-TRAINING DRILL

SCORES. THE PRESENT MODEL HAS A MEMORY OF 12,000 BYTES, WHICH LEAVES 7,000 BYTES AVAILABLE FOR THE OBJECT SCORE; THIS IS SUFFICIENT FOR A THREE-PAGE FOUR-PART PIECE SUCH AS BACH'S FUGUE 13, ABOUT 2 TO 3 MINUTES OF MODERATELY COMPLEX MUSIC. THE PRESENT MODEL COULD BE PROVIDED WITH UP TO 16,000 BYTES OF MEMORY, 11,000 FOR THE OBJECT SCORE, OR ABOUT 50% MORE THAN PRESENT. AS A RULE OF THUMB, EACH NOTE (FOR EACH VOICE, WITHOUT DURATION) TRANSLATES TO 6 BYTES OF PLAY SCORE (2 BYTES IF LEGATO) PLUS 6 BYTES FOR EACH DURATION. METRONOME VALUES AND OTHER PERFORMANCE INFORMATION IS NOT EXPLICITLY TRANSLATED INTO OBJECT INFORMATION, AND DOES NOT USE MEMORY.

APPENDIX C

Scott Vaughan
1821 Sena
Denton, Texas 76201

Dear Sirs:

I am a doctoral music composition student at North Texas State University, and I am searching for an interactive computer system which will play a multi-voiced musical score entered in character form from a standard terminal, controlled voltage keyboard, or whatever. A unit like the Rowland Micro Composer would be great, but it is out of my price range. I am looking for a system for around \$1,000.00.

Would you please send me any free information about your systems and about any relation they may have to the production of music.

Thank you very much for your time and help.

Sincerely,

Scott Vaughan

APPENDIX D

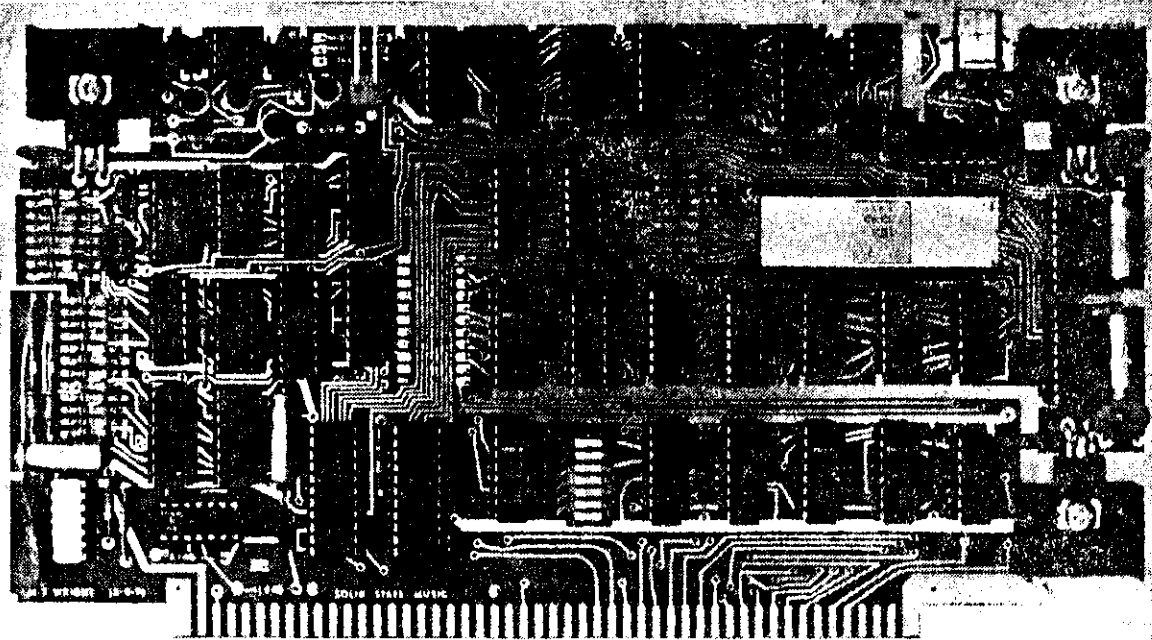
Companies Contacted

| | |
|-------------------------------|-----------------------|
| Alpha Microsystems | Sharp and Associates |
| Apple Computers | Space Byte |
| Audio Engineering | Systems Insight |
| Canada Systems | Technical Design Labs |
| Creative Computing | Terminal Systems |
| Creative Microsystems | |
| Cromenco | |
| Data Dynamics Texhnology | |
| Digital Group | |
| DynaByte | |
| E&L Instruments | |
| Egbert Electronics | |
| Electronic Control Technology | |
| Heathkit | |
| Icom | |
| Integral Data Systems | |
| Kent-Moore Instruments | |
| Micro Design | |
| Micromation | |
| Micro Systems Development | |
| Objective Design | |
| Ohio Scientific Research | |
| Osborne and Associates | |
| Paia | |
| Parasitic Engineering | |
| Processor Technology | |

APPENDIX E

MUSIC SYNTHESIZER BOARD

©1977



FEATURES

Standard

- Plug compatible with the ALTAIR 8800 and IMSAI 8080, or any other system using the "ALTAIR Bus".
- DIP Switch selection of the memory location from 32K and up.
- T.I. low profile sockets provided for all IC's.
- Gold plated edge connector contacts.

Synthesizer

- Frequency range is software controllable from 15HZ to 25KHZ.
- Frequency can be changed by software over a nine octave range.
- Volume of the SB-1 is software controlled over fifteen different levels.
- The waveform from the SB-1 can be defined by the user in 32 bytes of memory.
- The Attack and Sustained levels (Envelope) of a note can be defined by the user.

Software

- "MUS-X1 is a high music interpreter which can drive up to eight SB-1 boards at once.
- Note durations controllable from 1/64 up to a whole note.
- MUS-X1 only occupies 4K of RAM
- MUS-X1 uses standard ANSCII notation for music encoding making it easy for a person to write and correct musical tunes.

Drawn →

main train
 IMSAI 8080
 CRT

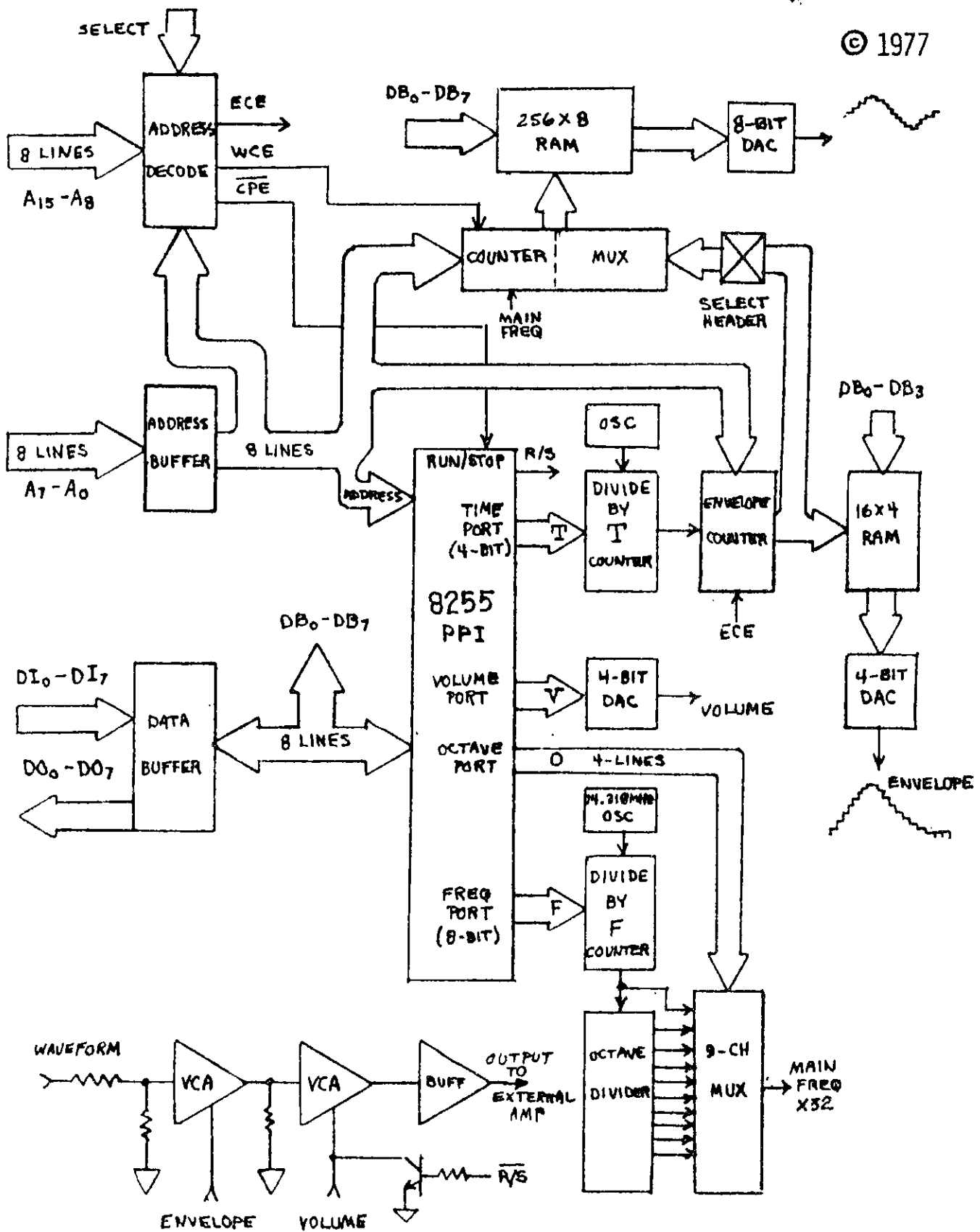
8K MEM

CASSETTE I/O

ROM

SOLID STATE MUSIC SB-1 BLOCK DIAGRAM

© 1977



SB-1 SYNTHESIZER BOARD

1.0 GENERAL DESCRIPTION

The Solid State Music Synthesizer Board(SB-1) is a waveform synthesizer card designed to interface with the S-100 bus. With this card, a microcomputer may be programmed to play a monophonic solo in an instrumental "voice" that is controlled by software. Polyphonic capability is provided for in MUS-X1 through multiple cards. The music is written in a high-level language called MUS-X1 which is entered into the system via an ANSCII keyboard,paper tape, floppy,or other input medium.

MUS-X1 is a real-time interpreter and therefore music can be heard immediately after the musical notation has been typed in,with no special pre-processing needed. The tempo of the music is controlled by a software timer built into MUS-X1 and has a range of 50 to 170 beats per minute in 2/2 to 8/8 time.

1.1 MEMORY MAP LOCATIONS

The SB-1 is a memory map device and can be moved through memory from 8000 Hex to FF00 Hex by presetting a switch on the card.

1.2 OUTPUT INTERFACE

The SB-1 provides signals to a 2-pin molex connector from an emitter follower circuit(low-impedance driver) at approximately 1 volt rms maximum.

1.3 DESIGN (see block diagram)

The SB-1 uses an 8255 programmable peripheral IC which controls the main five functions of the card through three bytes.

Main functions:

- 1....set frequency (1 byte)
- 2....set octave (1/2 byte)
- 3....set volume (1/2 byte)
- 4....set duration of envelope (1/2 byte)
- 5....set 4 special control bits (1/2 byte)

The board has two oscillators. A 20MHZ oscillator is used in the timing of a pre-settable counter for frequency control. The output of the frequency counter(Divide by F) is divided down again by another counter by two's to set up nine different octaves. The resultant frequencies from the octave-divider are selected by a multiplexer and used to drive a counter as a scanner of a 256-byte random-access memory(Waveform storage). 32 bytes of the memory is scanned at a time, and the stored memory data drives a Digital-to-Analog Converter(DAC) to generate a waveform in 32 segments.

1.3 (continued)

The second oscillator is a multivibrator (LM 555) running at a frequency of about 200HZ, and can be adjusted from 90HZ to 600HZ. The multivibrator frequency is used in the timing of a pre-settable counter for envelope duration control. The output of the duration counter (Divide by T) is used to drive another counter as a scanner of a 16X4 bit random-access memory (Envelope storage). The 16X4 memory's data drives a DAC for generating 16 changes in volume (an envelope) of the waveform when the card is commanded to play a note.

The waveform data and envelope data are combined into a composite waveform by a Voltage-Controlled Amplifier (VCA). The composite waveform's level (volume) is turned up or down by the use of another VCA.

The waveform and envelope shapes are defined by a string of bytes written into the SB-1's two memories. As an example, if a squarewave was wanted for a waveform, then 16 bytes of FF hex and 16 bytes of 00 hex would be loaded into the card. The waveform memory is capable of holding two sets of four 32 byte waveforms. The set of waveforms is controlled by the envelope circuitry so that a new waveform could be generated for every 25% of the envelope's duration.

1.4 SOFTWARE

The use of MUS-X1 does not require programming skill. If the user can interpret the notations on a piece of sheet music, he or she can relate the alpha-numeric instruction to it.

For example:

| | |
|-----------------|-------------------------|
| PP = Pianissimo | W = Whole note duration |
| -B = B flat | Q = Quarter note |
| +B = B sharp | / = End of a measure |

(100,3,4) = 100 beats per minute in 3/4 time

1.5 POWER REQUIREMENTS (From S-100 bus)

+7 to +9 Vdc unregulated @ 1.3 amps
+12 to +18 Vdc unregulated @ 25 milliamps
-12 to -18 Vdc unregulated @ 25 milliamps

1.6 PRICE (All parts, including IC sockets and software)

Kits - \$250.00

















Assembled and tested - \$350.00

MUSIC DECODING SHEET

Octave 4 Octave 5 Octave 6

NOTE.... C D E F G A B C D E F G A B C D E F G A B

Octave 1 Octave 2 Octave 3

| | | | |
|---|--|---|--------------------------|
|  | ...Whole note (W) |  | ...Quarter rest (QR) |
|  | ...Half note (H) |  | ...Eighth rest (OR) |
|  | ...Quarter note (Q) |  | ...Half rest (HR) |
|  | ...Eighth note (O) |  | ...Whole rest (WR) |
|  | ...Sixteenth note (S) |  | ...Flat (-) |
|  | ...Thirtysecond note (T) |  | ...Sharp (+) |
|  | ...Increase note duration by 50% |  | ...Natural (=) |
|  | ...Repeat (between these brackets repeat the measures as indicated.) |  | ...Measure separator (/) |

Music encoding examples

(K,+F) (120,2,2) I Q 3 B 4 C/ H D B / G Q A G /

(K,-B) (100,3,4) I Q5C4OAB5CF; I Q4A0FGAR /

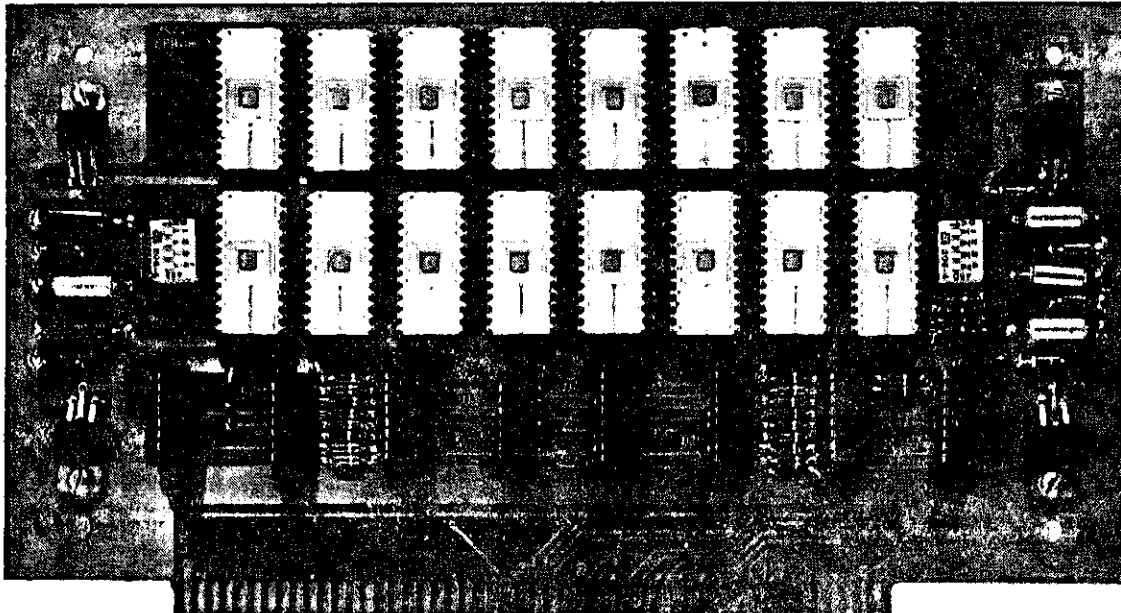
↑ Top part, Card 1

↑ Bottom part, Card 2

(K,-B) (90,3,4) MP 4HDQD;MP 3QDFA/ QAGA; QDFA/

2K/4K EPROM BOARD

© 1976



DESCRIPTION & PRICES

- o ALTAIR 8800 and IMSAI 8080 plug compatible
- o Option of 2K (8) or 4K (16) 1702As
- o Dip switch selection of addressing and wait cycles
- o Reverse voltage protection
- o On board regulators for +5 & -9 v
- o All sockets included in kits
- o Gold plated finger contacts

Complete Kit (less EPROMS) \$65.00
 Complete Kit (8 EPROMS) \$145.00
 Complete Kit (16 EPROMS) \$225.00

Prices include postage for U.S. and Canada. We ship UPS Blue Label when appropriate. UPS COD orders accepted. No credit cards please.

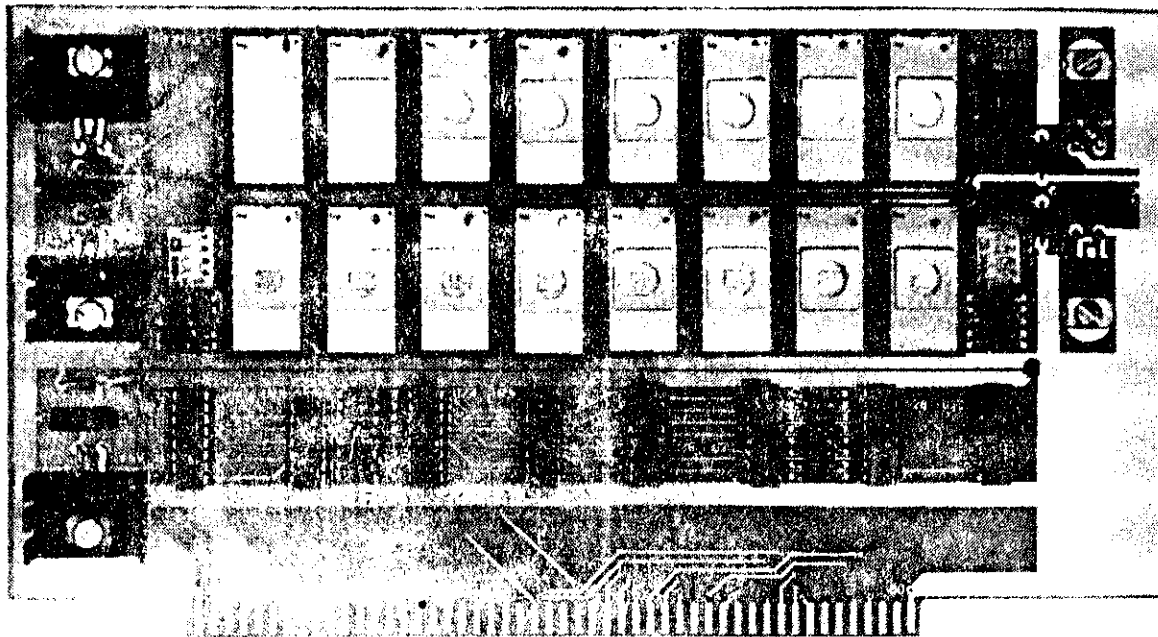
BASIC KIT LIST

- 2 CTS dip switches
- 16 24 pin low profile sockets
- 6 16 pin low profile sockets
- 2 14 pin low profile sockets
- 2 LM340T-5 regulators
- 2 LM320T-8 regulators
- 4 1N4001 diodes
- 2 56 ohm 5% resistors
- 6 470 ohm 5% resistors
- 19 2.7 K ohm resistors
- 6 tantalum capacitors
- 2 .1 ceramic capacitors
- 1 74L00
- 1 74L04
- 2 74142
- 1 74175
- 2 DM8097
- 1 DM8131
- 1 PC board
- 4 sets #6 screws, nuts & washers

2102A Walsh Ave Santa Clara, CA 95050

8K/16K EPROM (2708) BOARD

©1977

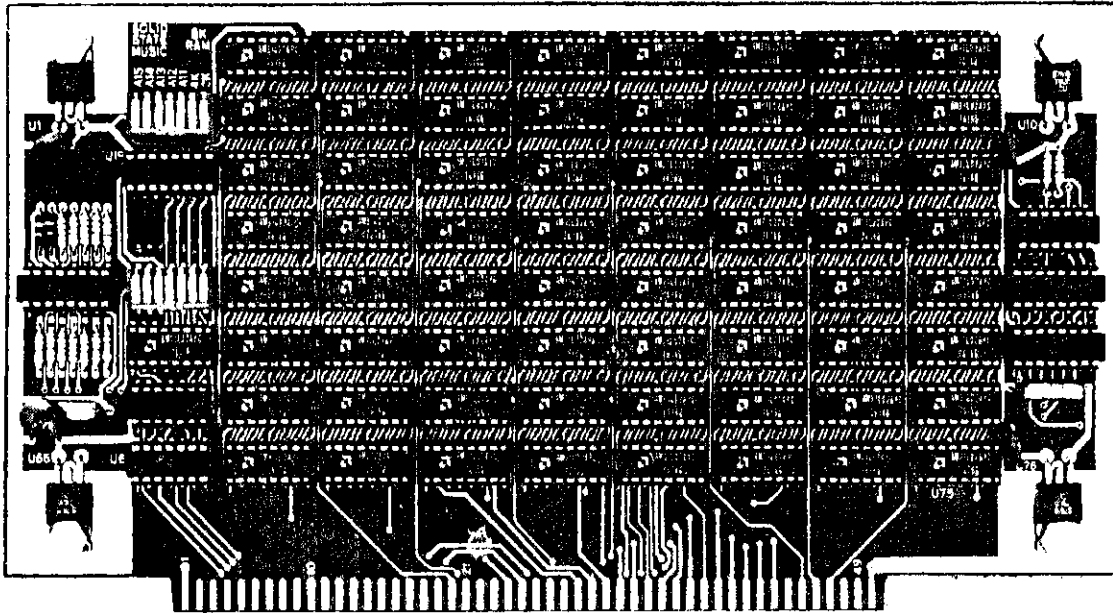


| FEATURES | PARTS LIST | |
|---|---|---|
| <ul style="list-style-type: none"> ○ Plug compatible with all S 100 Bus mainframes (Altair, IMSAI, etc.) | <ul style="list-style-type: none"> 2 U1, 11 (16) U2-9, 12-19 2 U10, 20 | <ul style="list-style-type: none"> 7812/340T-12 2708 (optional) LM340T-5 |
| <ul style="list-style-type: none"> ○ DIP switch selection of memory address assignment. | <ul style="list-style-type: none"> 1 U21 1 U22 1 U23 | <ul style="list-style-type: none"> 7805/LM340T-5 74LS175 74LS00 |
| <ul style="list-style-type: none"> ○ DIP switch selection of memory complement (8K/16K) | <ul style="list-style-type: none"> 2 U24, 28 1 U25 | <ul style="list-style-type: none"> 74LS42 74LS04 |
| <ul style="list-style-type: none"> ○ DIP switch selection of wait cycles (0 to 4) | <ul style="list-style-type: none"> 2 U26, 27 1 U29 | <ul style="list-style-type: none"> LM8097/74LS367 LM8131 |
| <ul style="list-style-type: none"> ○ No wait cycles required with optional 2708 EPROMs. | <ul style="list-style-type: none"> 5 D1-5 3 C1, 8, 9 7 C2-4, 6, 10-12 | <ul style="list-style-type: none"> 1N4002 10uF, 20V .1uF, 25V |
| <ul style="list-style-type: none"> ○ Low-power Schottky support chips. | <ul style="list-style-type: none"> 2 C5, C7 | <ul style="list-style-type: none"> 4.7uF DIP TA 20V |
| <ul style="list-style-type: none"> ○ Reverse voltage protection. | <ul style="list-style-type: none"> 25 R1-21, 23-26 1 R24 | <ul style="list-style-type: none"> 2.7K, 1/4W 82K, 1/4W |
| <ul style="list-style-type: none"> ○ T.I. low profile sockets provided for all ICs. | <ul style="list-style-type: none"> 2 S1, 2 5 5 1 | <ul style="list-style-type: none"> 4PST DIP Switch heatsinks sets #6 hardware board |
| <ul style="list-style-type: none"> ○ Gold plated edge connector contacts. | <ul style="list-style-type: none"> 16 2 6 | <ul style="list-style-type: none"> 24 pin sockets 14 pin sockets 16 pin sockets |

2102A WALSH AVE. SANTA CLARA, CA 95050
 (408) 246-2707

8K RAM BOARD

© 1977



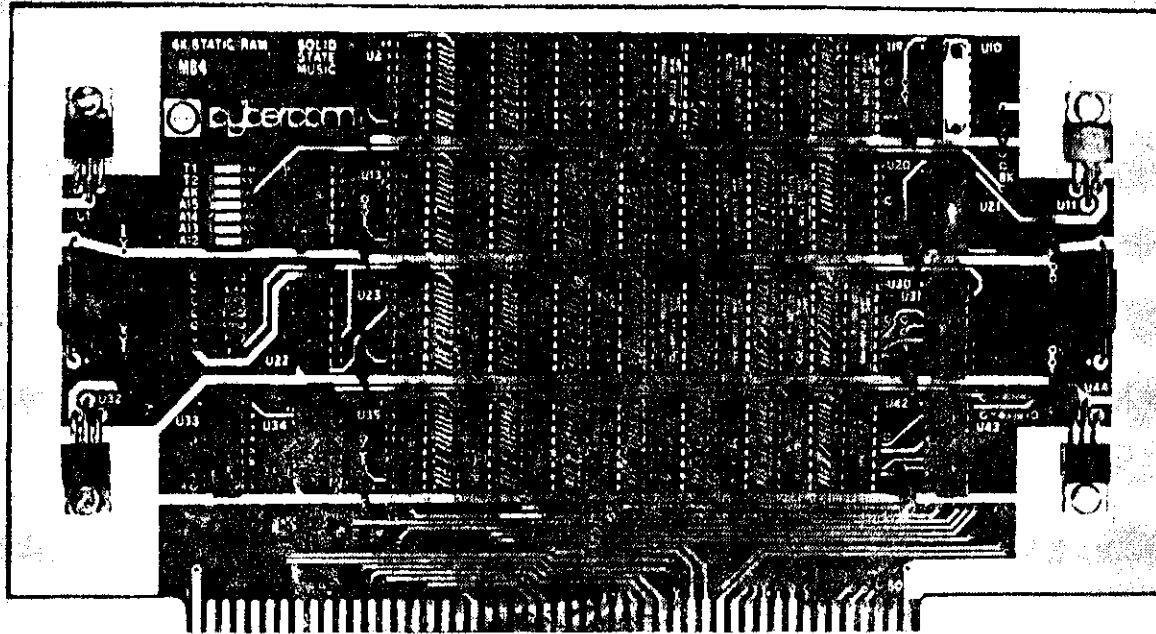
| FEATURES | PARTS LIST |
|---|---|
| <ul style="list-style-type: none"> ○ Plug compatible with the ALTAIR 8800 and IMSAI 8080, or any other system using the "ALTAIR bus." ○ Low-power, 500 nanosecond RAMs. No wait cycles required. ○ Low-power Schottky support chips. ○ DIP switch selection of memory address assignment and wait cycles. ○ Memory protect can be set for increments of 256 bits, 512 bits, 1K, 2K, 4K or 8K by DIP switch. ○ T.I. low profile sockets provided for all RAMs and ICs. ○ Gold plated edge connector contacts. | <ul style="list-style-type: none"> 1 PC Board with gold fingers 69 16 pin low-profile sockets 3 14 pin low-profile sockets 2 7 position DIP switches 65 91L02APC/D2102AL-4 low-power RAMs 1 74LS00 1 7402 1 74LS42 1 74LS74 2 74367 (DM 8097) 1 DM 8131 5 1N4003 diodes 4 7805 or 340T-5 regulators 2 2.7 uF/20V tantalums 4 0.1 uF disc capacitors 14 2.7K ohm, 1/4W resistors 3 39K, 1/4W resistors 4 sets #6 screws, nuts and washers 1 2-pin Molex connector & mate 1 instruction set |

2102A WALSH AVE. SANTA CLARA, CA. 95050

408/ 246-2707

4K/8K RAM BOARD

©1977



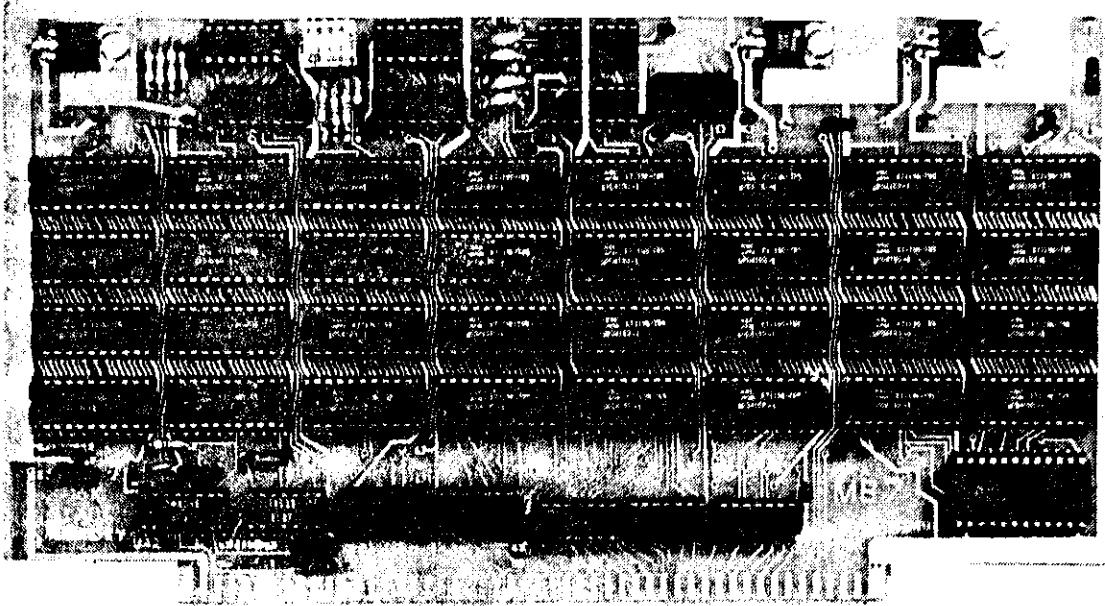
| FEATURES | PARTS LIST |
|--|---|
| <ul style="list-style-type: none"> ○ Plug compatible with the ALTAIR 8800 and IMSAI 8080, or any other system using the "ALTAIR bus." | <ul style="list-style-type: none"> 1 PC board with gold fingers 32 Low-power, 500ns RAMs (4K kit) or 64 Low-power, 500ns RAMs (8K kit) |
| <ul style="list-style-type: none"> ○ Low-power, 500 nanosecond RAMs. No wait cycles required. | <ul style="list-style-type: none"> 2 74100 or 74LS00 1 74102 or 74LS02 1 74142 or 74LS42 |
| <ul style="list-style-type: none"> ○ Low-power or low-power Schottky support chips. | <ul style="list-style-type: none"> 1 74174 or 74LS74 2 74367 or DM8097 1 DM8131 |
| <ul style="list-style-type: none"> ○ DIP switch selection of memory address assignment and wait cycles. | <ul style="list-style-type: none"> 4 7805 or 340T-5 regulators 1 7-position DIP switch |
| <ul style="list-style-type: none"> ○ T.I. low-profile sockets provided for all RAMs and ICs. | <ul style="list-style-type: none"> 36 16 pin low-profile sockets 4 14 pin low-profile sockets |
| <ul style="list-style-type: none"> ○ Gold plated edge connector contacts. | <ul style="list-style-type: none"> 2 tantalum capacitors 12 .1 uf disc capacitors |
| <ul style="list-style-type: none"> ○ Designed to allow "piggy-backing" of RAMs to make 8K of memory. Your option of 4K or 8K kit. | <ul style="list-style-type: none"> 7 1K ohm $\frac{1}{4}$W resistors 2 4.7K ohm $\frac{1}{4}$W resistors 1 10K ohm $\frac{1}{4}$W resistor 1 33K ohm $\frac{1}{4}$W resistor 4 sets of #6 hardware for regulators 1 instruction set |

2102A Walsh Ave. Santa Clara, CA 95050

/408/ 246-2707

16K STATIC RAM BOARD

© 1977



FEATURES

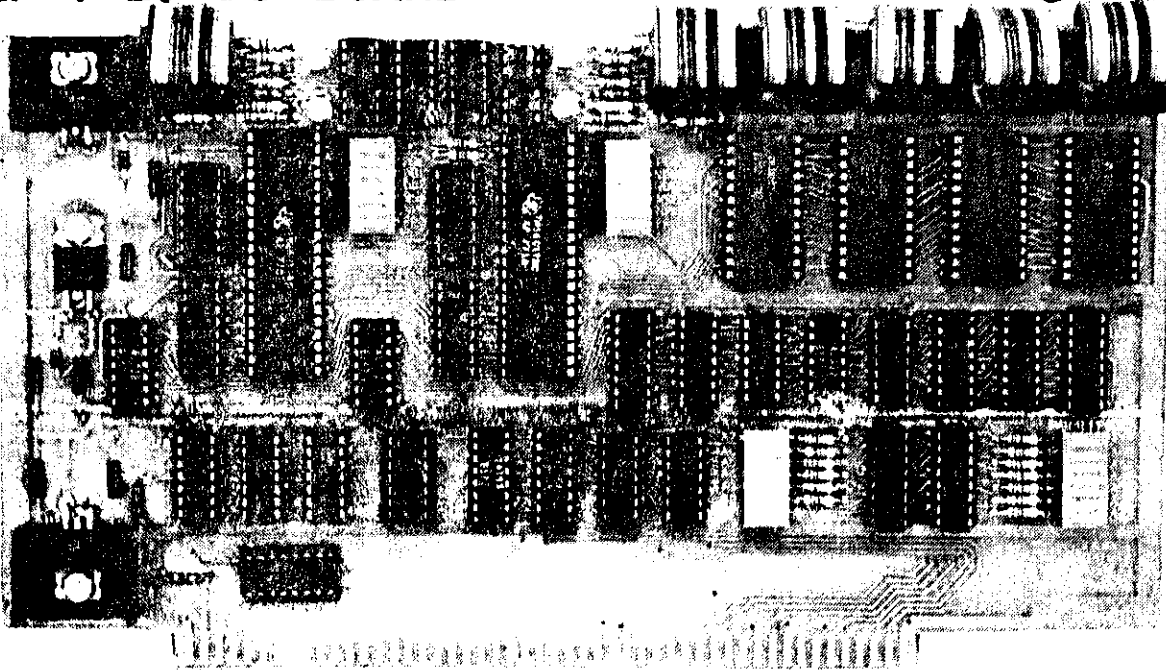
- S-100 bus compatible.
- Fully buffered address & data lines.
- Extra lo-power, 200nsec 4K x 1-bit static RAMs allow operation over 2MHz.
- Board inhibit line allows hardware jumps on reset or power-up and/or memory expansion beyond 64K.
- Address selection at any 4K boundry with DIP switch at top of board.
- Memory protect in 4K blocks with automatic unprotect on power-up.
- T.I. lo-profile sockets for all ICs.
- Hi-grade glass-epoxy board with gold contacts.

PARTS

- | | |
|------------------------------|--------------------------|
| 1- 74L00 | 4- 100pF disc capacitors |
| 4- 74L04 | 1- .001uF ceramic disc |
| 1- 7483 | 18- .1uF ceramic |
| 1- 74LS32 | |
| 1- 74(LS)47 | 1- 100pf 5% |
| 1- 7442 | 1- .0033uF 5% |
| 1- 7481A | 2- 2.7uf @ 20V tantalum |
| 1- 74121 | 1- 4 position DIP switch |
| 1- IM8098 or 74368 | 1- heat sink |
| 1- 8212 | 3- sets #6 hardware |
| 32- uPD410 | 1- PC Board |
| 4- M10026 | 4- 8 pin sockets |
| 1- 7812 or 340T-12 | 1- 24 pin socket |
| 12V regulator | 4- 16 pin sockets |
| 2- 7805 or 340T-5 | 8- 14 pin sockets |
| 5V regulators | 32- 22 pin sockets |
| 1- 1N751 or 1N5231 | |
| 5V zener | |
| 1- 560 ohm 1/2 watt resistor | |
| 1- 68 ohm 1/2w 5% | |
| 1- 100 ohm 1/2w | |
| 6- 1.0K 1/2w | |
| 10- 4.7K 1/2w 5% | |
| 1- 3.92K 1% RN55-type | |

2P + 2S I/O BOARD

© 1977



FEATURES

SYSTEM COMPATIBILITY

S-100 Bus Computer Systems

SERIAL PORTS

- Two serial I/O ports (2-I & 2-O) with full handshaking status from UART's.
- Status sense easily reversible.
- Status bit position jumper selectable on an IC header.
- 20/60 mA current-loop interface with on-board optical isolators or EIA interface.
- Independent baud-rate selection for input and output from 55 to 9600 baud (EIA) or 55 to 4800 baud (I-loop).
- Regulated +5V, +12V, & -12V outputs provided on both serial headers.
- Number of stop bits, parity even or odd, word length and Status/Data port reversible by DIP switch control.

PARALLEL PORTS

- Two latched I/O ports (2-I & 2-O) using 8212 latch-buffers.
- Regulated +5V & -12V outputs provided on all 4 parallel headers.

ADDRESSING

- Independent DIP switches for setting address of the serial and parallel port blocks.

P.C. BOARD

- Hi-grade glass epoxy with plated-thru holes and gold-plated edge connector contacts.
- T.I. low-profile sockets are provided for all IC's and headers.

POWER REQUIREMENTS

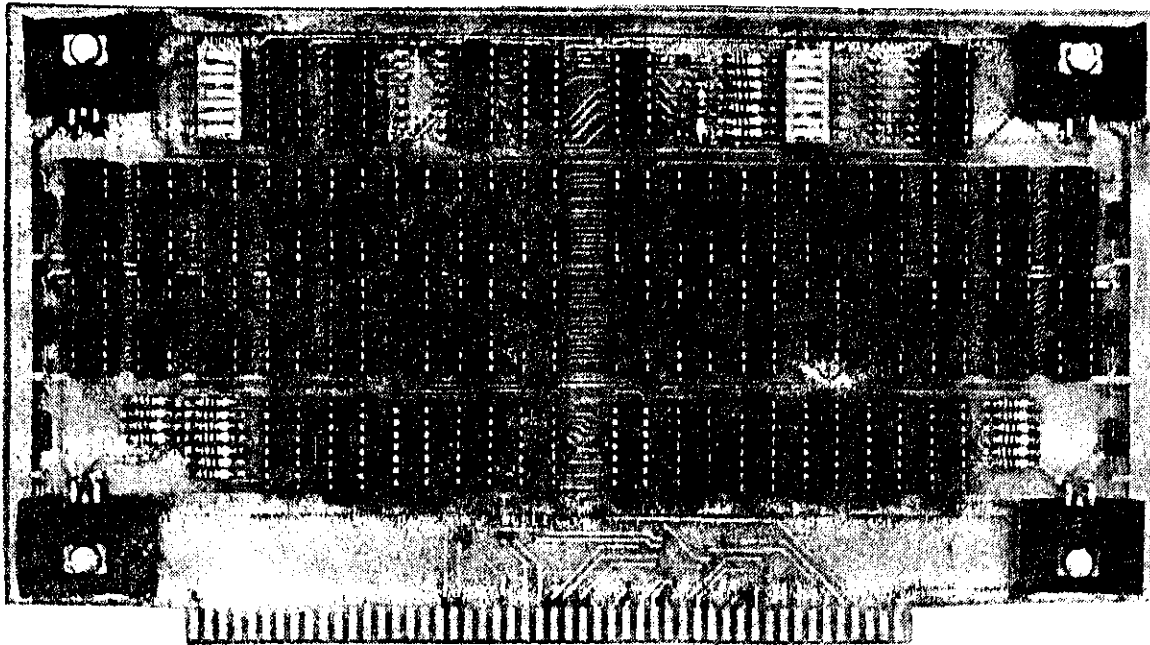
- +8V @ .92A, +16V @ .6A, & -16V @ .08A typical

SPECIAL FEATURES

- Interrupt capability provided for on serial and parallel ports.
- Two spare 16-pin DIP patterns provided.
- All jumpers on IC headers for easy I/O reconfiguration.

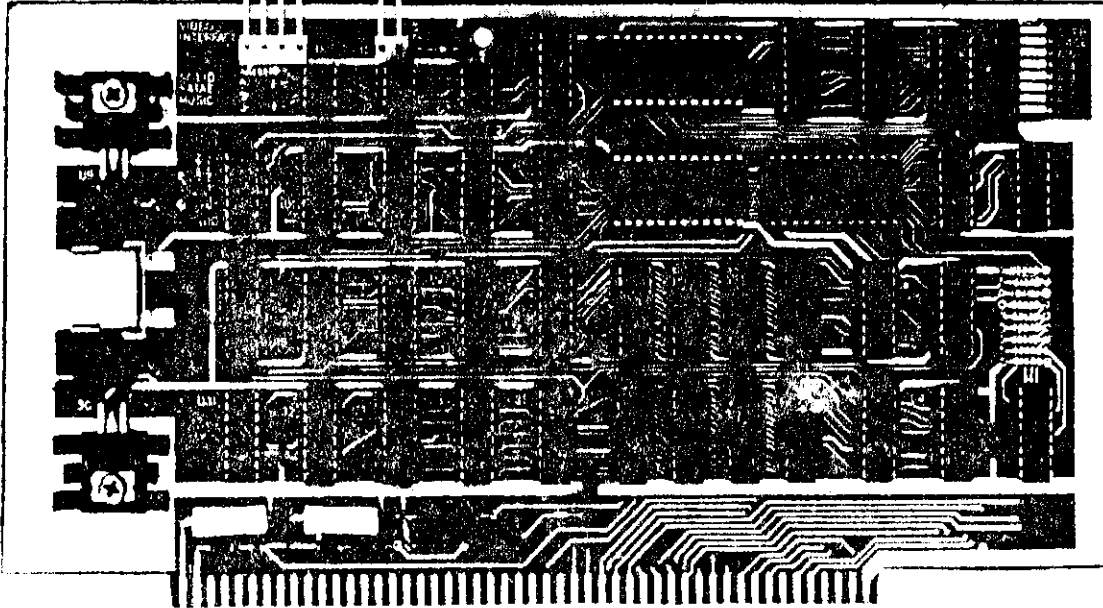
4K STATIC PROM/RAM BOARD

© 1977



| FEATURES | PARTS | |
|---|--|---|
| <p>S-100 bus compatible.</p> <p>Fully buffered address & data lines.</p> <p>Jump on reset circuitry allows execution of programs at any 1K boundary by hitting reset. (May be modified to jump on power-up.)</p> <p>Jump address & enable set by DIP switch at top of board.</p> <p>Memory protect by DIP switch at top of board.</p> <p>Wait states (0-2) selectable by DIP switch at top of board.</p> <p>Uses 2112 RAMs & 82S126/82S129, 74S287/74S387, or DM8573/DM8574 PROMs.</p> <p>RAM & PROM may be mixed in 256-byte increments in any pattern.</p> <p>Power requirements: +8V @ 1.5A typical (4K RAM) +8V @ 2.4A typical (2K PROM & 2K RAM)</p> <p>T.I. lo-profile sockets for all ICs.</p> <p>Hi-grade glass-epoxy board with gold contacts.</p> | <p>4 ea. U1,8,41,53</p> <p>2 ea. U2,3</p> <p>2 ea. U4,7</p> <p>1 ea. U5</p> <p>2 ea. U6,47</p> <p>(2-32 ea.) U9-40</p> <p>5 ea. U42,43,50-52</p> <p>1 ea. U44</p> <p>1 ea. U45</p> <p>1 ea. U46</p> <p>1 ea. U48</p> <p>1 ea. U49</p> <p>17 ea. D1-17</p> <p>6 ea. C1-6</p> <p>1 ea. R1</p> <p>1 ea. R3</p> <p>24 ea. R4-27</p> <p>2 ea. S1,2</p> <p>4 ea.</p> <p>4 ea.</p> <p>1 ea.</p> <p>43 ea.</p> <p>6 ea.</p> | <p>7805/340T-5</p> <p>74(LS)257</p> <p>74LS42</p> <p>74LS175</p> <p>74LS08</p> <p>2112 (optional RAM)</p> <p>82S126/82S129</p> <p>or 74S287/74S387</p> <p>or DM8573/DM8574.</p> <p>(optional PROM)</p> <p>8T97</p> <p>74LS04</p> <p>74LS74</p> <p>74LS32</p> <p>74LS00</p> <p>DM8131</p> <p>1N914/4148/4448</p> <p>2.7-20uF, 15V</p> <p>680 ohm ±W</p> <p>560 ohm ±W</p> <p>2.7K ±W</p> <p>7PST DIP switch</p> <p>heatsinks</p> <p>sets #6 hardware</p> <p>p.c. board</p> <p>16-pin sockets</p> <p>14-pin sockets</p> |

VIDEO INTERFACE



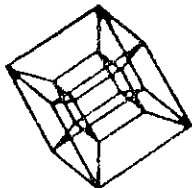
| FEATURES | PARTS | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
|--|--|-------------|----------------------------|------------------|----------------------------|--------------|----------------------------|-----------|-------------------------|----------|-----------------------|---------|--------------|---------|---------------|---------|--------------|---------|-----------------------|----------|----------------------|----------|-------------------|----------|------------------------|----------|--------------------|----------|--------------|----------|------------------------|---------------|-------------------|---------|------------------|----------|--------------------|----------------|-------------------|--------------|---------------------|--------------|--------------------|-----------|--|
| <ul style="list-style-type: none"> ■ S100 Bus Compatible ■ 32 or 64 Characters per line ■ 16 Lines ■ Graphics (128 x 48 matrix) ■ Left & Right horizontal margins of about 8% of the full raster width ■ Upper vertical margin of about 6% ■ Vertical rate- 59.3Hz, Horizontal rate- 16.01KHz ■ Parallel & Composite video ■ On board low power memory ■ Powerful software included for cursor, home, BOL, scroll, Graphics/Character, etc. ■ Upper case, lower case & Greek ■ Output to video monitor or video amplifier in T.V. set ■ Black-on-white & white-on-black ■ Sockets included | <table style="width: 100%; border-collapse: collapse;"> <tbody> <tr> <td style="width: 50%;">1- FC board</td> <td style="width: 50%;">7- 100 ohm $\frac{1}{4}$w</td> </tr> <tr> <td>1- copy software</td> <td>1- 220 ohm $\frac{1}{4}$w</td> </tr> <tr> <td>1- MCM6571AP</td> <td>2- 470 ohm $\frac{1}{4}$w</td> </tr> <tr> <td>1- 74LS00</td> <td>9- 2.7K $\frac{1}{4}$w</td> </tr> <tr> <td>1- 74S04</td> <td>2- 1K $\frac{1}{4}$w</td> </tr> <tr> <td>1- 7408</td> <td>2- 15 ohm 3w</td> </tr> <tr> <td>1- 7432</td> <td>10- .01 discs</td> </tr> <tr> <td>4- 7474</td> <td>1- 50pf disc</td> </tr> <tr> <td>1- 7486</td> <td>2- 2.7uf 20v Tantalum</td> </tr> <tr> <td>2- 74150</td> <td>3- 39uf 10v Tantalum</td> </tr> <tr> <td>1- 74153</td> <td>1- 1N746A ZD 3.3v</td> </tr> <tr> <td>2- 74157</td> <td>1- 1N4742/1N716 ZD 12v</td> </tr> <tr> <td>1- 74166</td> <td>1- 12.3MHz crystal</td> </tr> <tr> <td>2- 74161</td> <td>2- heatsinks</td> </tr> <tr> <td>4- 74193</td> <td>2- sets No. 6 hardware</td> </tr> <tr> <td>6- 74367/8097</td> <td>3- 24 pin sockets</td> </tr> <tr> <td>1- 8131</td> <td>3- 8 pin sockets</td> </tr> <tr> <td>3- 75451</td> <td>25- 16 pin sockets</td> </tr> <tr> <td>2- 340T-5/7805</td> <td>9- 14 pin sockets</td> </tr> <tr> <td>8- 2102AI,-2</td> <td>1- 8 POS DIP switch</td> </tr> <tr> <td>2- Plug sets</td> <td>1- Instruction set</td> </tr> <tr> <td>1- 2N2222</td> <td></td> </tr> </tbody> </table> <p style="text-align: center; margin-top: 20px;">Call or write for information on our complete line of fine products.</p> | 1- FC board | 7- 100 ohm $\frac{1}{4}$ w | 1- copy software | 1- 220 ohm $\frac{1}{4}$ w | 1- MCM6571AP | 2- 470 ohm $\frac{1}{4}$ w | 1- 74LS00 | 9- 2.7K $\frac{1}{4}$ w | 1- 74S04 | 2- 1K $\frac{1}{4}$ w | 1- 7408 | 2- 15 ohm 3w | 1- 7432 | 10- .01 discs | 4- 7474 | 1- 50pf disc | 1- 7486 | 2- 2.7uf 20v Tantalum | 2- 74150 | 3- 39uf 10v Tantalum | 1- 74153 | 1- 1N746A ZD 3.3v | 2- 74157 | 1- 1N4742/1N716 ZD 12v | 1- 74166 | 1- 12.3MHz crystal | 2- 74161 | 2- heatsinks | 4- 74193 | 2- sets No. 6 hardware | 6- 74367/8097 | 3- 24 pin sockets | 1- 8131 | 3- 8 pin sockets | 3- 75451 | 25- 16 pin sockets | 2- 340T-5/7805 | 9- 14 pin sockets | 8- 2102AI,-2 | 1- 8 POS DIP switch | 2- Plug sets | 1- Instruction set | 1- 2N2222 | |
| 1- FC board | 7- 100 ohm $\frac{1}{4}$ w | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1- copy software | 1- 220 ohm $\frac{1}{4}$ w | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1- MCM6571AP | 2- 470 ohm $\frac{1}{4}$ w | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1- 74LS00 | 9- 2.7K $\frac{1}{4}$ w | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1- 74S04 | 2- 1K $\frac{1}{4}$ w | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1- 7408 | 2- 15 ohm 3w | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1- 7432 | 10- .01 discs | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 4- 7474 | 1- 50pf disc | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1- 7486 | 2- 2.7uf 20v Tantalum | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2- 74150 | 3- 39uf 10v Tantalum | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1- 74153 | 1- 1N746A ZD 3.3v | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2- 74157 | 1- 1N4742/1N716 ZD 12v | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1- 74166 | 1- 12.3MHz crystal | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2- 74161 | 2- heatsinks | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 4- 74193 | 2- sets No. 6 hardware | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 6- 74367/8097 | 3- 24 pin sockets | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1- 8131 | 3- 8 pin sockets | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3- 75451 | 25- 16 pin sockets | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2- 340T-5/7805 | 9- 14 pin sockets | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 8- 2102AI,-2 | 1- 8 POS DIP switch | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 2- Plug sets | 1- Instruction set | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 1- 2N2222 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

2102A Walsh Ave., Santa Clara, CA 95050

(408)246-2707

©1977

APPENDIX F



A L F Products Inc.

128 S. Taft; Denver, CO 80228; (303)234-0871

3 July 1978

Scott Vaughan
1821 Sena
Denton, TX 76201

Enclosed is information on our computer controlled music synthesis equipment. We are currently producing a 71 key organ-like keyboard which can be connected into a computer with an RS-232C standard serial input port. The mechanism is custom made for us by Pratt-Reed, makers of keyboards for Arp, Moog, Oberheim (Eμ), Steinway, and others. This unit may be useful for entering scores to a computer system, since it is serial compatible and does not require an A to D converter for connection to digital equipment.

I do not know what the price range of a Roland sequencer is. Prices of our equipment are enclosed. We do not currently offer a computer system, but computers are easily obtained in a wide range of prices. Our Pitch Generators (10-5-9 and 10-5-10) are very low cost, and may be ideal if you do not require control of parameters other than pitch (or if you have filters and envelope generators for external manually controlled use, triggers are easily obtained from the Pitch Generator). The 10-5-10 parallel version will interface to nearly any computer having a TTL parallel output port with 10 or more bits. More than one pitch generator can be used if more than 4 voices are required.

If you need control of pitch, waveform, envelope, volume; then the AD8 series components would be more suitable. Although the 10-5-12 is designed for the S-100 (Mits 8800) bus, it can be connected into nearly any system using an 8080 processor with a little work.

If you need any additional information, please let me know.

A handwritten signature in cursive script, appearing to read "Philip Tubb".

Philip J. Tubb

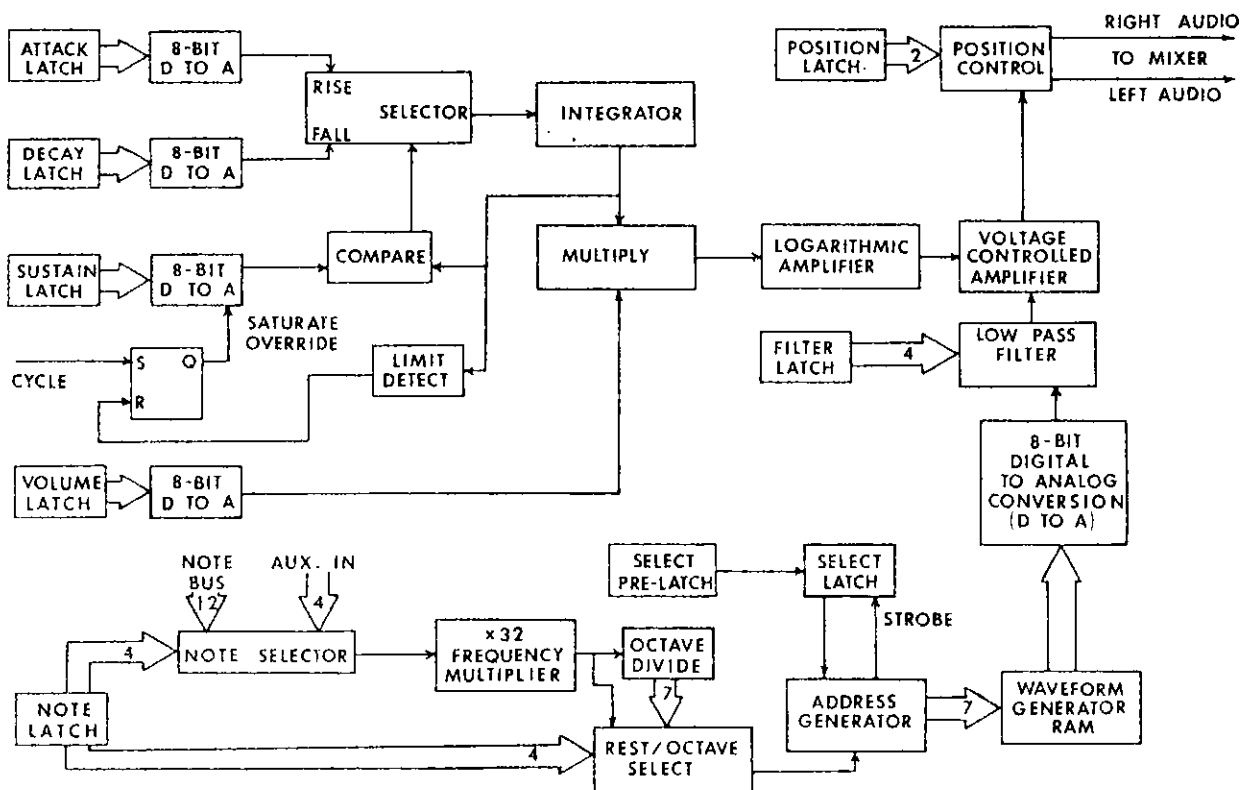
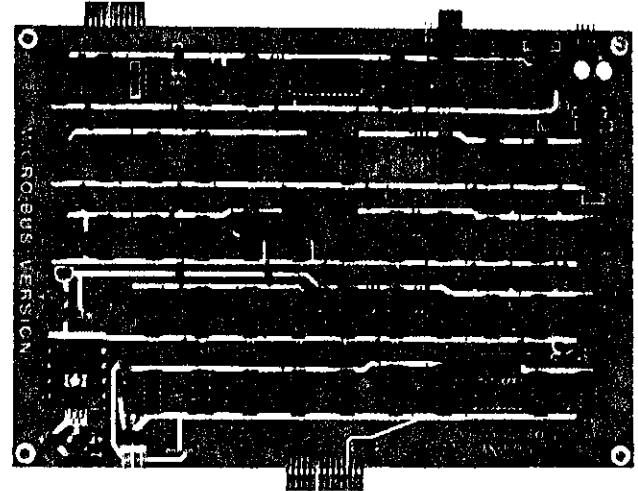
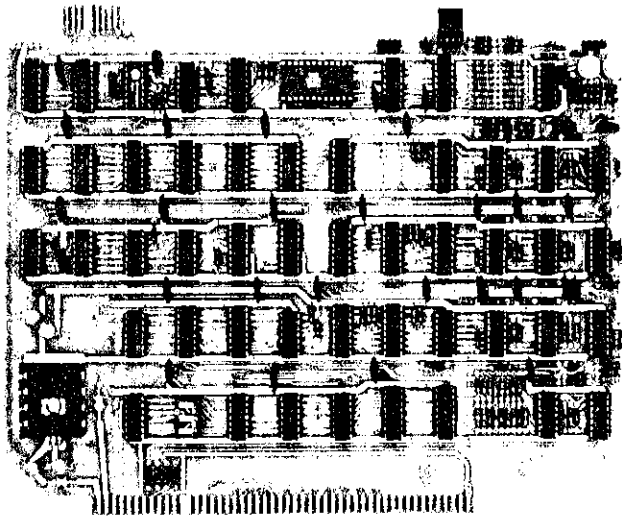


AD8 SERIES

SYNTHESIS BOARDS

AD8 10-5-4

AD8 MICRO-BUS 10-5-11



GENERAL

The 10-5-4 and 10-5-11 Synthesis Boards are synthesizers for use in AD8 series music synthesizer systems. The 10-5-4 synthesizer is compatible with AD8 Controllers and AD8 Adapters, and the 10-5-11 synthesizer is compatible with AD8 Micro-Bus Controllers and AD8 Micro-Bus Adapters.

Detailed information on the functions of these synthesizers is given in the paper "An Introduction to AD8 Music Synthesizers", available free from ALF.

OPTIONS

When ordering a 10-5-4, either option A1 or A2 must be specified. 10-5-4(A1) is the standard model, which uses a 14-pin header for board address selection. 10-5-4(A2) is a slightly higher cost version which uses a DIP switch for board address selection, allowing simple address changes when necessary. When ordering an A1 version assembled, board address (an integer from zero to seven) must be specified. Also available on the 10-5-4 is option 1 which replaces the 2 MHz 6810-1 waveform memory with a 4 MHz 68B10 memory; to order add a comma 1 to the option list, e.g. 10-5-4(A1,1).

When ordering a 10-5-11, either option A1 or A2 must be specified as described above. Option 1 (described above) is also available. The 10-5-11 is available in three power configurations: +9 and +17 volts filtered unregulated (standard), +8 and +16 volts filtered unregulated (S-100), or +5 and +12 volts regulated $\pm 1\%$. When ordered assembled, the power configuration should be added to the option list as P1 (9 and 17), P2 (8 and 16), or P3 (5 and 12). Example: 10-5-11(A1,P1) is the standard model, assembled.

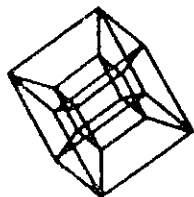
Both synthesizers are available in kit form. ALF kits are intended only for those skilled in electronic construction and testing; request the Kit Policy Sheet prior to ordering kits. Note: synthesis boards are particularly difficult to assemble and test to those not familiar with the circuit. An oscilloscope may be required if solder bridges, incorrect parts placement, or other assembly errors are encountered. To order a synthesis board in kit form, add a comma K to the option list: e.g., 10-5-11(A1,K).

MOUNTING

The 10-5-4 is a 10.5" wide (7.5" tall) card with a 100 conductor (50 pairs at .125" spacing) edge connector 1.5" from the left side, and is designed for use in a card frame with backplane board (motherboard). The 10-5-11 is an 11" wide (7.5" tall) card with a 20 and a 26 conductor (10 and 13 pairs at .1" spacing) edge connector (both for ribbon cable connection), and is designed to be bolted into a stack of boards through four holes (one at each corner) suitable for #8 bolts. Four male/female 8-32 threaded hex spacers plus eight washers are supplied for 7/8" center-to-center mounting.

TYPICAL POWER CONSUMPTION

Per board: 425 mA at +9 volts, 55 mA at +17 volts.

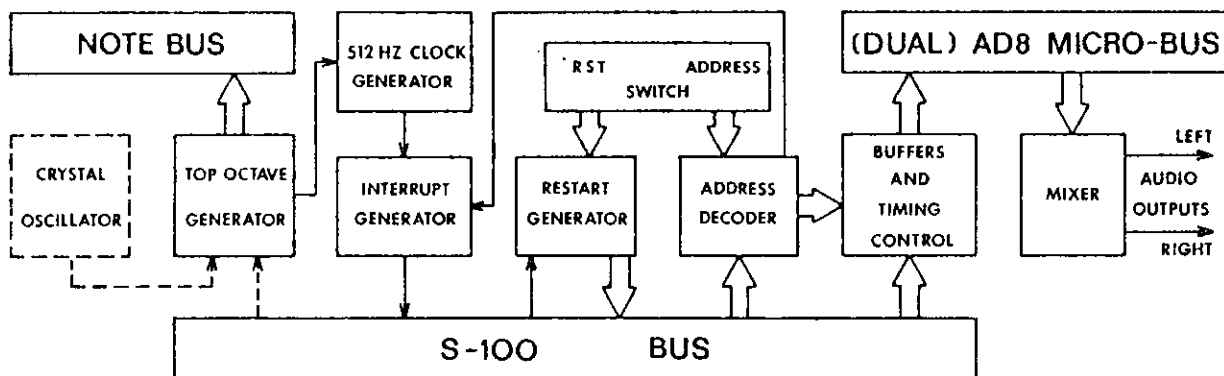
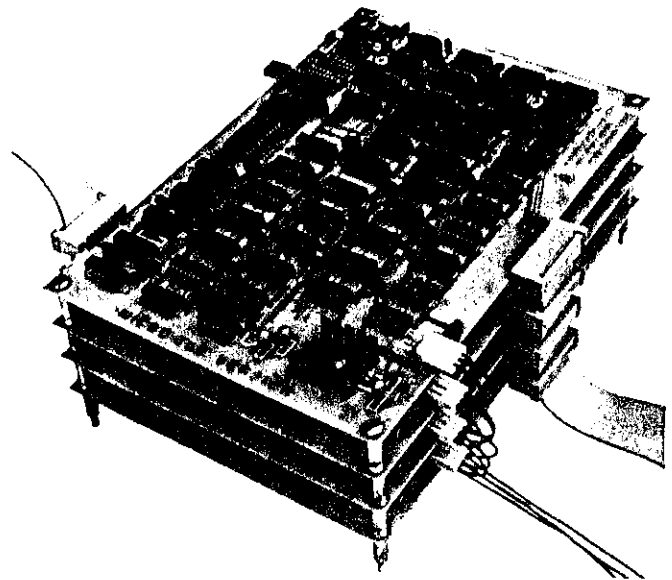
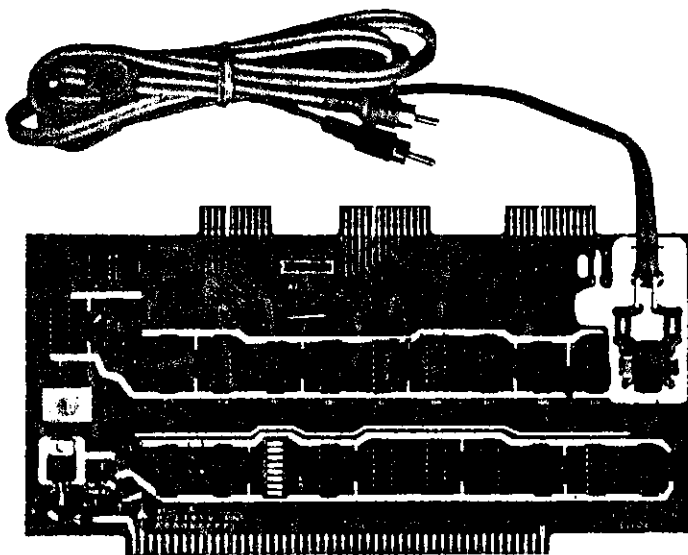


A L F PRODUCTS INC.
128 SOUTH TAFT
DENVER, COLORADO 80228



AD8 SERIES

AD8 MICRO-BUS ADAPTER S-100 COMPATIBLE



- **S-100 PLUG-IN COMPATIBLE**
See owner's manual for details.
- **512 HZ INTERRUPTING CLOCK**
For note duration and interval timing.
- **ON-BOARD INTERRUPT AND RST GENERATORS**
For use with or without priority interrupt cards.
- **LOW POWER SCHOTTKY TTL CIRCUITRY**
+8 volts at 375 mA, +16 volts at 25 mA (typical).
- **ADDRESSED AS 2K OF MEMORY**
Allows fast, easy function changes.
- **DIP SWITCH ADDRESS SELECTION**
To any 2K boundary.
- **PROVISIONS FOR EXTERNAL INPUTS**
Allows for non-standard interrupt or pitch frequencies and connection to unused note bus lines.
- **NOTE BUS PROVIDES AD8 TOP OCTAVE**
From 2 MHz S-100 clock or optional crystal oscillator.

GENERAL

The 10-5-12 Adapter Board is designed to plug into an S-100 computer system, a standard stereo amplifier, and one or more AD8 Micro-Bus Synthesis Boards and/or accessories. The board conforms to most S-100 specifications, and is approximately 5" tall with connectors in place. Complete details on S-100 compatibility and requirements are given in the Owner's Manual (see price list for ordering information). A stereo output cable with dual phono plugs is used for amplifier connections. The AD8 system is described in the paper "An Introduction to AD8 Music Synthesizers", available free from ALF.

The Adapter Board can control one to eight Synthesis Boards (part number 10-5-11). (See photos of Adapter Board and stack of Synthesis Boards on reverse side.) Two edge connectors at the top of the board (one for the Micro-Bus and the other for the Note Bus) are connected to similar edge connectors on the Synthesis Boards by means of two ribbon cable assemblies. These ribbon cables plug into the Adapter Board, run out of the computer system, and connect to each Synthesis Board. Only one slot in the S-100 system is required even when using eight Synthesis Boards. An additional edge connector at the top of the Adapter (which duplicates the Micro-Bus) is used for connecting AD8 Micro-Bus accessories. Additional Adapter Boards can be used when it is desirable to control more than eight Synthesis Boards.

The Adapter Board also contains a clock which interrupts 512 times per second. This clock can be used as a time reference in order to produce notes of various durations or to generate other precision delays. The clock can be used in conjunction with existing interrupt hardware (e.g. priority interrupt cards), or the on-board interrupt and RST generators can be used in systems with no interrupt hardware.

TYPICAL POWER REQUIREMENTS

The Adapter Board requires about 375 mA from the +8 volt supply and about 25 mA from the +16. When using the S-100 power supply to drive the Synthesis Boards, an additional 425 mA from the +8 volt supply and 55 mA from the +16 volt supply per board is required. (For eight boards, this is approximately 3400 mA and 440 mA.)

SYNTHESIS BOARD CONTROL

The Adapter Board is addressed as 2K (2,048 bytes) of memory. When a byte of data is written into the "memory", it is sent over the Micro-Bus to the Synthesis Boards and/or other accessories. The address written to is also available on the Micro-Bus. Each function and waveform element of each Synthesis Board has a unique memory address assigned to it. By writing a byte of data to that memory address, the function or waveform element can be programmed. The Adapter Board is not active during memory read (except as described below for the interrupting clock). S-100 compatibility has been tested for various 2 MHz and 4 MHz systems; additional information is available from ALF (please indicate system type of interest). For 4 MHz operation, Synthesis Boards must be ordered with option 1.

TOP OCTAVE GENERATION

Twelve pitch references ranging from approximately 3520 Hz to 6645 Hz are provided for use by the Synthesis Boards. These pitches are derived from a single high frequency clock input. This clock input can be supplied as an external input (at the option header) for special tuning or pitch bending; the input range is 100 kHz to 2.5 MHz; 2,00024 MHz results in standard tuning. Using a 2 MHz frequency from pin 49 (CLOCK) of the S-100 bus or the 2,00024 MHz frequency from the optional on-board crystal oscillator, the twelve pitches are within 0.1% of the A=440 Hz tuning standard. Each pitch's frequency is the twelfth root of two times the previous pitch's frequency.

INTERRUPTING CLOCK

The on-board clock is designed to interrupt the computer at a rate of 512 times per second. (Actual rate with 2 MHz top octave frequency input: 511.12 Hz. Timing accuracy with crystal option: within 0.1 seconds per minute; timing self-consistent to 0.002%.) Interrupts may be counted by software to determine note durations or other intervals. For example, to produce a note 1/4th second long, the note is started, 128 interrupts are counted, and the note ended. Accurate durations can be created regardless of variations in software speed.

On systems with a priority interrupt generator card (which allows the V10 through V17 lines to be used with automatic RST or CALL generation), the board can be jumpered to any one of the 8 vectored interrupt lines.

For systems with no interrupt provisions, on-board circuitry allows for interrupt generation and RST generation. The interrupt line is jumpered to PINT for automatic interrupt generation, and the RST generator is enabled to SINTA. On interrupt, the RST generator will automatically cause a RST instruction to be executed, causing a "call" to an interrupt routine. Routine address can be selected from 0, 10, 20, 30, 40, 50, 60, or 70 (octal) by means of a DIP selection switch. No external hardware for interrupt processing is required. Note: a priority interrupt generator card is recommended if more than one interrupting device is present in the S-100 system.

The interrupt routine must acknowledge the interrupt to clear the interrupt request and reset the circuitry for the next clock cycle. The interrupt is acknowledged by reading a byte from any memory address within the 2K selected for the board.

SOFTWARE

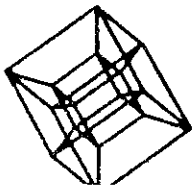
8080 software is available for reproduction costs from ALF. (Music data may be higher priced due to royalties to the artists.) For information contact: ALF Products, Software Division; 128 South Taft, Denver, CO 80228.

OPTIONS

The Adapter Board is available assembled or in kit form. ALF kits are intended only for those skilled in electronic construction and testing; request the Kit Policy Sheet prior to ordering kits. The kit form part number is 10-5-12(K).

When ordering assembled, the part number is 10-5-12(P1) or 10-5-12(P2) depending on power requirements. P1 is ALF standard for use where +9 and +17 volts are available. P2 is S-100 standard for use where +8 and +16 volts are available.

Option 1 adds the crystal oscillator for use on computers where CLOCK (pin 49) is not 2 MHz and for systems where this signal may not be accurate enough for pitch generation. To order, add a comma 1 to the option list: 10-5-12(K,1), 10-5-12(P1,1), or 1-5-12(P2,1).

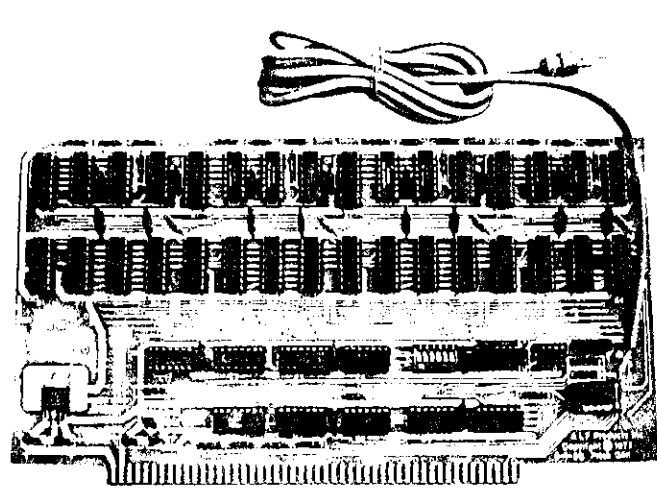


ALF PRODUCTS INC.
128 SOUTH TAFT
DENVER, COLORADO 80228

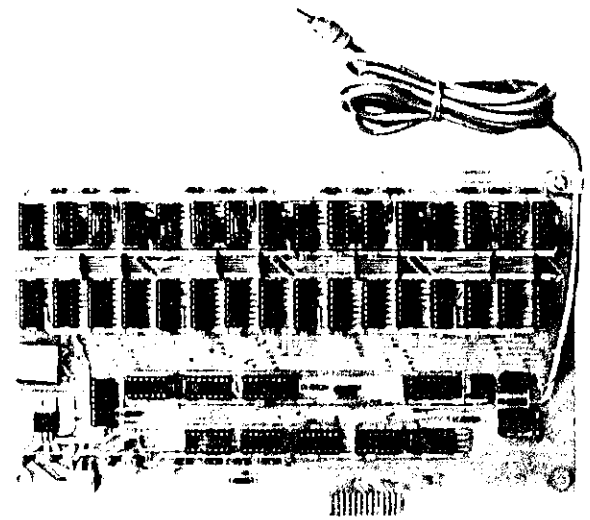


IMSAI
North Star
Creative

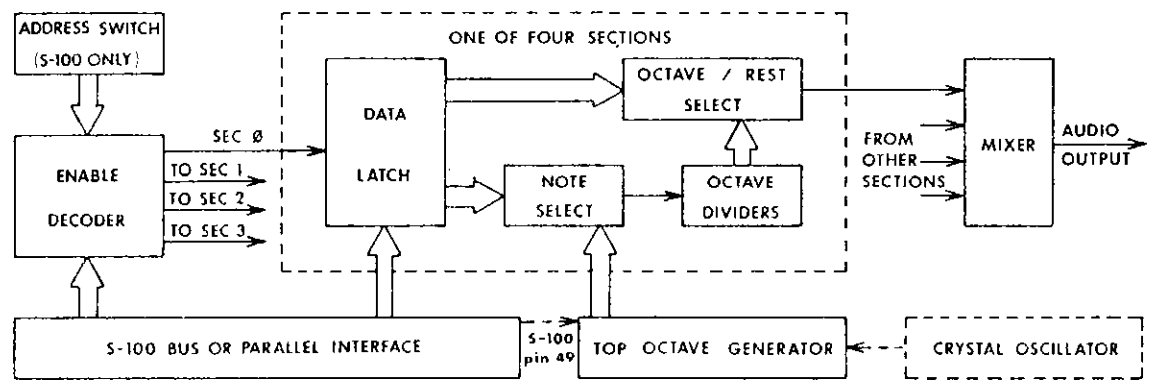
QUAD CHROMATIC PITCH GENERATORS



10-5-9 (A4) S-100 COMPATIBLE



10-5-10 (A4, I, P2) PARALLEL INTERFACE



- **PRODUCES 1-4 TONES SIMULTANEOUSLY**
Two boards produce 2-8 tones in stereo.
- **LOW COST, EXPANDABLE**
Accessory boards add additional functions.
- **TOP OCTAVE GENERATION FROM SINGLE 2 MHZ CLOCK**
From pin 49 on S-100 version, external input on parallel version. On-board crystal oscillator is an available option.
- **96 PITCHES FORM AN EIGHT OCTAVE RANGE**
Includes full standard piano range. All pitches within 0.1% (equal spacing, A=440 Hz) with 2 MHz input.

- **AUDIO OUTPUT CONNECTS TO STANDARD AMPLIFIER**
- **A SUBSET OF THE AD8 SYNTHESIZER SERIES**
Other AD8 functions can be added with accessories.
- **LOW POWER SCHOTTKY TTL CIRCUITRY**
+8 volts at 485 mA, +16 volts at 40 mA (typical).
- **SOFTWARE AVAILABLE FROM ALF**
Most software available at reproduction costs.
- **RECREATIONAL**
Relative enjoyment factor (REF) exceeds 82 (typical).

DESCRIPTION

The 10-5-9 Quad Chromatic Pitch Generator is a single board audio pitch generator for musical applications. It is S-100 compatible, and will plug directly into an S-100 computer and an audio amplifier. (Complete details on S-100 signals used and timing expected are given in the Owner's Manual.) This board is also available in a parallel output compatible version using a SIMP-A interface, as described in CSL publication #1-77A (available from ALF for \$1); interface requirements are also described in the Owner's Manual. The parallel version is part number 10-5-10.

The board consists of a control section, and four channel sections. The control section contains the S-100 address selection circuitry (or SIMP-A interface circuitry) and the top octave frequency synthesizer. Each channel section selects frequencies from the control section to produce one of 96 possible tones. Each channel section is independent from any other channel section, and each is identical in function. The board may contain the components for only one section, for two or three sections, or for all four channel sections.

Each channel produces a pitch which is selected programmatically from any of 96 frequencies which form an eight octave range. The entire standard piano range is included, plus 8 higher frequencies. Frequency range is about 27.5 Hz to 6645 Hz. Using all four channels, four pitches can be produced simultaneously.

APPLICATIONS

The Pitch Generator series is ideal for starting out in computer controlled music. The simple, straight-forward design makes it easy to create accessories, and a wide variety of accessory boards will be available from ALF. The Pitch Generator consists of four duplications of the Note Selection circuitry from the AD8 Synthesis Board. The remaining AD8 synthesis functions (waveform, envelope, and volume control) will be available on accessory boards. The programming format of the Pitch Generator is the same as the AD8 series, allowing for simple conversion of AD8 software for use with Pitch Generators.

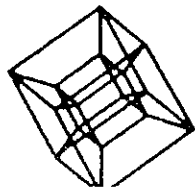
To produce more than four simultaneous tones, two boards are easily connected to a stereo amplifier (or a mono amp with a simple Y connector) to allow up to 8 simultaneous tones. (More tones can be achieved with additional boards.) Individual outputs from each section are available (on the option header) for connection to external synthesizers and for accessory boards. Manual volume controls can also be connected, as shown in the Owner's Manual. The Pitch Generator is ideal as a computer controlled pitch source for use with existing synthesizer systems. It is also ideal for educational applications (computer assisted instruction) in melody construction and similar tasks where envelope and waveform control are not required.

PROGRAMMING

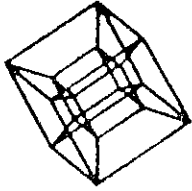
The Pitch Generator requires very little processor attention, leaving the computer free to perform additional tasks while producing tones. The Pitch Generator is very simple to program. In the S-100 version, the board takes four consecutive output addresses, the first of which must be divisible by four. Thus, each channel has its own port. By outputting a byte to the channel's port, a pitch or rest is programmed into that channel. The channel will continue producing the indicated pitch (or rest) until another output byte is received. The output byte contains the rest/play flag in bit 7, the octave number in bits 6 through 4, and the note number in bits 3 through 0. Outputs are processed immediately so no busy flags are required. For those new to music, the Owner's Manual includes a section on reading sheet music.

OPTIONS

When ordered in kit form, the board can be ordered with 1, 2, 3, or 4 channels. (Channels can later be added with add-on kits.) When ordered assembled, the board is available only with all 4 channels. On S-100 versions, a 2 MHz clock is expected from pin 49 (CLOCK); on parallel versions a 2 MHz signal may be supplied to the option header. If a 2 MHz clock is not available (on either version), order the on-board crystal oscillator option (option 1). The parallel version is available for use with +8 and +16 volts (filtered, unregulated) as option P2 or for use with +5 and +12 volts (regulated, $\pm 5\%$) as option P3; if the 10-5-10 is ordered assembled, either P2 or P3 must be specified.



ALF PRODUCTS INC.
128 SOUTH TAFT
DENVER, COLORADO 80228



A L F Products Inc.

128 S. Taft; Denver, CO 80228; (303)234-0871

21 July 1978

Scott Vaughan
1821 Sena
Denton, TX 78201

I believe I sent you all information available on the QCPG and the AD8 series aside from the Owner's Manuals, which you could order if you wish. If you have any specific questions I would be glad to answer them. (I will enclose data and price information in case they were not sent previously.) We do not currently have software available, although we are working on 8080 machine language software for both the QCPG and the AD8.

There are no data sheets available for the 71-key keyboard since it is still be developed. The keyboard consists of the 71 keys, 9 pushbuttons, and the electronics and enclosure. The keyboard requires a small amount of +8 or +12^{volt} power, filtered. The keys and switches form 80 possible keys. When one of these 80 keys is pressed, its code (0-79) is sent serially at rates from 300 to 9600 baud (DIP switch selectable); using RS-232C transmission and standard 1 start, 8 data, 1 stop bit format. When a key is released, its code (0-79) plus 128 is sent. Plans were also made to include a switch which shifts the code up into the ASCII printing character range; but I don't know if this will be present in the final model. For computers which do not allow use of the parity bit (the most significant bit), software can simulate this bit since it is always the complement of its state in the previous transmission

(with the same code); and the computer can be programmed to assume an initial transmission of 0 ("pressed") since initially no keys are pressed. The keyboard is about 4 feet long, 9 inches deep, and 5 inches tall; and cannot be shipped by post office or UPS. The price is expected to fall somewhere near the \$400 range (plus shipping), but I'm not sure how accurate that is. In real-time performance use, either for performance or entry of compositions, baud rates of 4800 or 9600 are recommended. Lower baud rates may result in poor "feel" due to the lag in transmission. We find 2400 to be marginal, and notice no problems at 4800; we therefore tend to use 9600 as an extra precaution. (With 10 bits per transmission code, including start and stop bits, 9600 baud is of course a rate of up to 960 codes per second, or 480 press/release combinations per second. With an 8 channel (voice) system, this allows the performer to use up to 60 8-note chords pressed and released in a single second.) I believe this covers most the details of the keyboard.

We feel there is no difficulty in using the synthesizers for real-time performance; we plan to produce whole systems (including the computer and software) which allow real-time and pre-programmed material to be produced simultaneously.

Let me know if you need any additional information.



Philip J. Tubb

An Introduction

to

AD8 MUSIC SYNTHESIZERS

CONTENTS

- 1 THE BASIC SOUND ELEMENTS
 - 1 Pitch
 - 2 Waveform
 - 3 Envelope and Volume
- 5 TYPICAL VOLTAGE CONTROLLED SYNTHESIS METHOD
- 7 THE AD8 SERIES SYNTHESIS METHOD
- 11 GRAPHS OF AD8 FUNCTIONS
- 13 AD8 SERIES SYNTHESIS DETAILS
 - 13 Pitch and Waveform
 - 18 Oscilloscope Photos
 - 19 Envelope and Volume
 - 20 Remainder of the Synthesis Board
- 21 COMPLETE SYSTEMS
- 23 DISCUSSION

© Copyright 1978 by ALF Products Inc.

Written by: Philip Tubb.

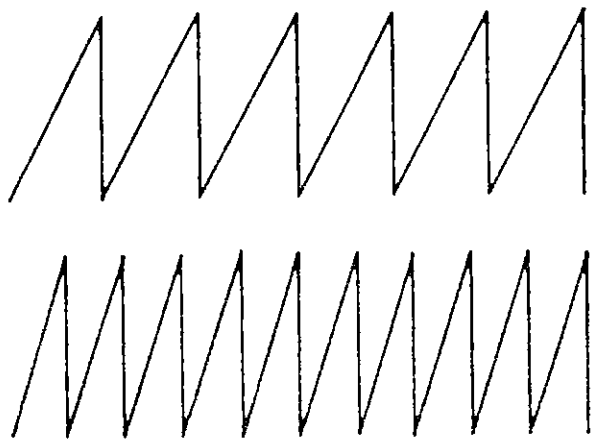
Graphics, photos: Rick Harman,
John Ridges, and Forrest Thiessen.

THE BASIC SOUND ELEMENTS

There are many ways to think of sound. In this paper, we will discuss only one method because it is used in many music synthesizers, including the AD8 series. If we think of a sound in terms of pitch, waveform, envelope, and volume; then we know the basic parameters which synthesizers use to create sounds.

PITCH

Most sounds create a pattern of air movement which repeats over and over. When the sound is produced electronically, usually a speaker cone will move in that pattern to produce the sound. If we were to draw the movement of two tones with different pitches, it might look like this:



A single section of the repeating pattern is called a "cycle" of the tone. The number of cycles occurring in some fixed amount of time determines the pitch of the sound. Usually this is measured in cycles per second, which are called Hertz (Hz). The top drawing has 60 cycles per second (only 1/10 second is shown), the bottom drawing has 100; thus the bottom drawing has a higher pitch.

If a pitch had about 262 cycles per second (a 262 Hz tone) then it would be what musicians call "middle C", which is "C3" in AD8 terminology. If it were twice that (524 Hz) it would be a C one octave higher ("C4" in AD8 terminology).

In most music, the pitches used are not randomly selected, but are picked from a set of traditional pitches. There are twelve such pitches, and they are called A, A sharp, B, C, C sharp, D, D sharp, E, F, F sharp, G, and G sharp. (Other names include B sharp which is C, E sharp is F, A flat is G sharp, B flat is A sharp, C flat is B, D flat is C sharp, E flat is D sharp, F flat is E, and G flat which is F sharp.) The lowest pitch available on the AD8 series is an A (A0) whose pitch (or "frequency") is about 27.5 Hz. The remaining 11 pitches are spaced evenly to the A in the next octave (A1) whose pitch is about 55 Hz (twice 27.5).

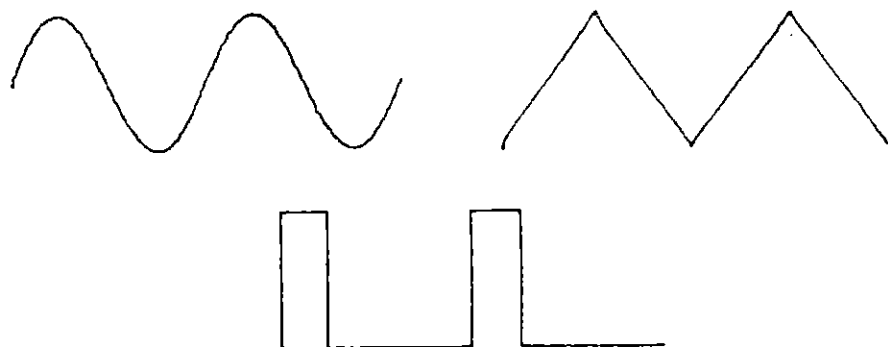
The AD8 has 8 octaves (numbered 0 through 7) each of which have these 12 traditional pitches. The lowest pitch of each octave is the A (or "A natural") whose frequency is 27.5, 55, 110, 220, 440, 880, 1760, and in the highest octave (A7), 3520 Hz. The pitch of each A sharp is about 1.06 (the 12th root of 2) times the pitch of the A in the same octave; the pitch of each B is about 1.06 times the pitch of each A sharp; and so forth. This gives 96 pitches (from the lowest A, A0, to the highest G sharp, GS7) with each pitch having a frequency of 1.06 times the previous pitch.

WAVEFORM

Here is a drawing of two sounds of the same pitch but with different waveforms. Two cycles of each sound are shown:



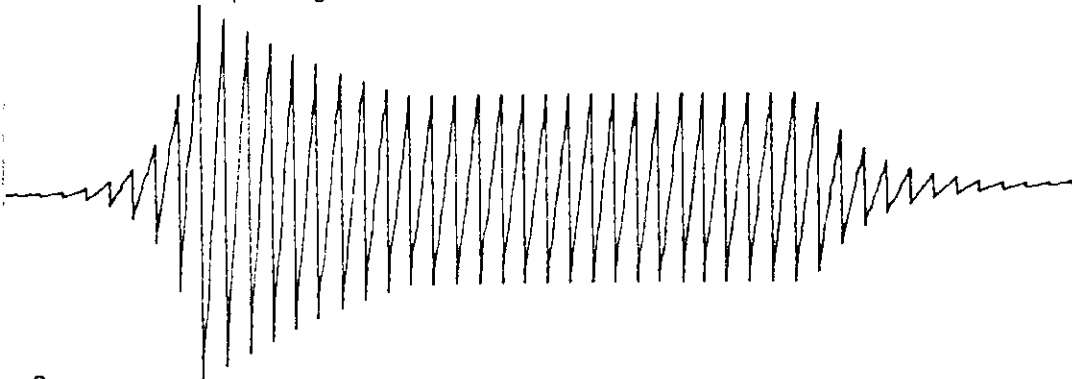
The sound on the left has a "square" waveform and the one on the right a "sawtooth". The names are descriptions of the way each cycle looks when drawn. Some other commonly used waveforms are sine (left), triangle (right), and pulse (bottom):



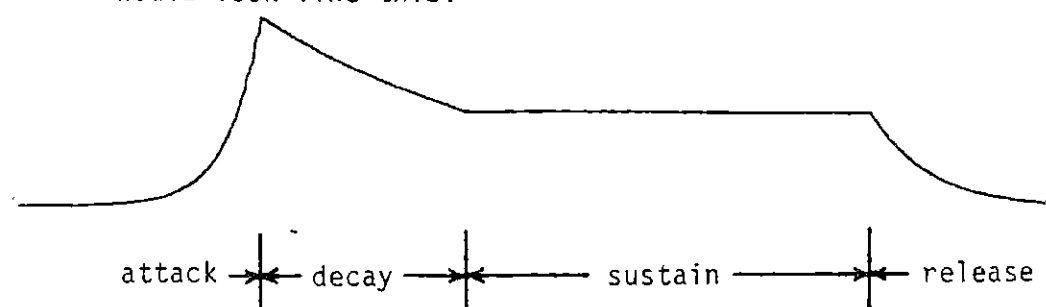
Different waveforms produce different sounds, but the differences are best heard rather than described in writing. There are a variety of ways waveforms can be produced, and some of them are described later.

ENVELOPE AND VOLUME

Most sounds do not simply start at full volume and cut off when they stop. The way sounds begin and end is called the "envelope" of the sound (in terms of volume only). Since the AD8 uses the term "volume" for another feature, the volume of a sound at different points in its envelope is referred to as "loudness" (and the volume at different points in its waveform is called "amplitude"). The loudness (and volume) of a sound has to do with the size of its movement. A typical sound with envelope might look like this: (sawtooth wave)



The envelope of a sound is generally drawn with the outline of the sound only. The above sound's envelope would look like this:



In the above drawing, the envelope is divided into four stages. The first is called the "attack". The loudness of the sound goes from zero to its maximum. Attack is usually referred to as a rate; a sound where the attack takes a long time (and the loudness rises slowly) is called a slow attack rate, and one where the attack takes a short amount of time (the loudness rises quickly) is called a fast attack.

The next stage is called "decay". The loudness drops from maximum to a lower level. Decay is also considered a rate; and is slow or fast (or in between) just as an attack rate is.

The next stage is called "sustain". The loudness stays about the same during the sustain stage. This level (which the decay dropped to) is called the "sustain level". It is usually considered to be a fraction or percentage of the maximum loudness (reached at the switch from attack to decay, which is the "volume level" in AD8 terminology).

The final stage is the "release" stage. It is identical to decay except that rather than dropping from the volume level to the sustain level, the loudness drops from the sustain level to the zero level. The loudness is thus returned to the same level it started at (before the attack stage) so the next envelope may begin.

TYPICAL VOLTAGE CONTROLLED SYNTHESIS METHOD

In the traditional synthesizer, such as those developed by Moog, all parameters are handled as voltages. Signal sources produce a voltage which is controlled manually. A keyboard produces a different voltage for each key; a simple oscillator produces a voltage which changes at a rate selectable by a knob. Signal processors take in a control voltage and produce an output voltage which is related to the control voltage. A voltage controlled oscillator uses the control input to control the rate at which the output voltage varies. The outputs of signal sources are connected to signal processors; and the outputs of the signal processors can go into other processors (perhaps along with additional signal sources) or can be the final audio output.

A typical synthesizer might start with a keyboard as a signal source. The output voltage of the keyboard would be run into the control voltage input of a voltage controlled oscillator (VCO). The VCO will put out a pitch which is determined by which key on the keyboard is pressed. The output voltage of the VCO is that pitch, and the waveform would probably be one of the five waveforms shown on pages 2 and 3. To create other waveforms, the VCO output is connected to the input of a filter (another signal processor). There are many types of filters which are used to create different changes in the VCO's waveform. By selecting the proper type or types of filters, and adjusting the various controls of the filters, a variety of waveforms can be created. To add the envelope, the signal is put into an envelope generator which is triggered by an additional output of the keyboard. When a key is pressed, the envelope generator changes the loudness of the signal it is processing to follow the contours of the desired envelope. Overall volume and stereo positioning would be added by means of a mixer or additional processors.

The main theory of the traditional synthesizer is quite simple. All manual inputs (such as knobs and keyboard keys) are converted into voltages. All signal sources

put out voltages. All devices which affect the sound (the signal processors) are controlled by voltages and output voltages. If all these voltages are compatible, then any output may be used to control any input; and any output may be used as an audio signal.

The main advantage of all this is speed and precision. Since the human performer cannot easily control, for example, the loudness of the pitch (envelope) while playing, the envelope generator does this for him each time he presses a key. If the performer could manually adjust the level while playing, there would be no need for an envelope generator; the performer could generate his own envelopes with much more variety than an envelope generator can. There are usually so many parameters to control that even simpler controls than envelope generation are performed automatically since the performer prefers to directly control only a few of the most important details. Other advantages besides speed include freedom from the boredom of repetition, and the ability to produce consistently accurate and precise results. Thus, most synthesizers include devices to perform those tasks which require high speed, accuracy, or large amounts of repetition. The performer controls only those parameters which he chooses to have maximum control over.

The major flaw in this theory is the cost of such a device. Since it is not known at the design time which functions the performer will wish to directly control, all functions must have provisions for automatic control. In theory, this is done by making the controls of all functions voltage inputs, and making controllers with the necessary voltage outputs. Thus, either controllers or manual signal sources can be used as the controlling agent. This is very expensive. Most signal processors have only a few control inputs and the majority of the parameters are controlled only manually using knobs on the processor. Usually the knobs are used to control parameters which (in the opinion of the designers) do not need to be changed often or do not require extreme accuracy in the settings. For example, most envelope generators have only a trigger input (which starts the cycle); and the attack rate, decay rate, sustain level,

and release rate are controlled manually by knobs. If the performer decides that all notes will have a certain envelope except the note "C" which will have a different envelope, most envelope generators do not have inputs which would allow the connections required for this particular effect.

An additional problem is the physical interconnection of all the voltage inputs and outputs. Originally wires called "patch cords" were manually connected between each input and output used. (Since the wiring is changed for different functions, it could not be wired at the factory; each configuration was wired by the performer and changed when the performer desired a different function.) Changes in the patch cord setup required a great deal of time, and the large number of cords required resulted in a complex tangle of cords routed across the synthesizer. Later, various clever interconnection schemes were developed (most of which resulted in reduced versatility) but in general all interconnections must be set in some manner by the performer and cannot be quickly changed and recreated. For this and other reasons, most stage performers use several synthesizers so each can be set for different connections.

THE AD8 SERIES SYNTHESIS METHOD

The method of control in the AD8 series is somewhat more intangible. The most simple configuration, for use with computer systems only, has no manual inputs. All of the functions are controlled by the computer system, as directed by programs written by the performers. In systems for performance use, the manual inputs (such as keyboards and knobs) are converted into (binary) numbers. For example, each key on a keyboard is numbered, and when a key is pressed or released, its number can be sent to the computer. All signal sources and processors are controlled by numbers.

In a simple system, the performer presses a key, and the key's number is sent to the external computer. The computer converts this number into the proper format for

a pitch generator (if necessary) and sends it to the pitch generator. This is similar to connecting a keyboard to a VCO in the traditional synthesis method. The computer also sends a number to the envelope generator causing it to start the envelope. This description is quite similar to that of the traditional synthesizer; just change "voltage" to "number". The difference is that the voltages in the traditional method go directly from the keyboard to the VCO and so forth; whereas the number in the AD8 all go to and from the computer. The programmer in the computer does the routing rather than actual wiring.

This programming is very easy to change. It can be stored on paper and magnetic tape, cards, discs, and other storage media. For example, a performer can set up the routing program, make any changes he decides are needed, and store the program on a magnetic disc. Any time the same settings are desired, they can be read off the disc. In addition to function settings, background melodies or entire scores can also be stored.

The programming is also more versatile than patch cords. The computer can change the programming in a fraction of a second, in response to inputs from the performer. For example, pressing the "C" key can be a cue to the computer to change the envelope; allowing C to have a different envelope as discussed above. The computer can run through a sequence of processes each time a key is pressed. The need for several synthesizers on stage is reduced since the AD8 can change all parameters in a fraction of a second; the performer need only define the parameters in advance and program them to be recalled at the touch of a button.

Additionally, there is the versatility of using a computer. Programs can be written to change key signatures, supply chord progressions, program multiple boards at once, store previous melodies and play them back for the performer to hear while he plays additional themes (which can also be memorized by the computer), and a variety of other functions.

Naturally, there are disadvantages as well. The performer who wishes to buy an AD8 with performance keyboard and controls, then use only pre-programmed features will not be able to take full advantage of the computer system. Although the pre-programming supplied with a performance system may be more versatile than other synthesizers, even more control can be attained through the use of custom computer programs. On some synthesizers, creating a new effect not possible with the original synthesizer requires the addition of new hardware; this new hardware (electronics) may have to be custom designed and built. In the AD8 series, a new effect may also require such additional equipment. It may require only custom computer programs. You may wish to create such programs yourself. There are a number of books and classes available at local computer stores on computer programming; and programming is not difficult to learn (although it is usually time consuming). It is also possible to hire a programmer, and there are many talented programmers especially at universities with computers. Custom programming is also available from ALF.

Another disadvantage is that numeric control is, like voltage control, expensive. It is difficult to decide how extensive the numeric control should be. If the amount of control is small, the price is low but there may not be sufficient degrees of control for one or more applications. If the amount of control is large, the price is high and there is the chance that some degree of control is superfluous. The amount of control available on any given function is usually specified by the number of "bits" used to control the function. For example, if 3 bits are used to control the attack rate, then only $2 \times 2 \times 2$ or 8 different levels (i.e., 8 different attack rates) would be possible. The AD8 uses 8 bits ($2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2 \times 2$ or 256 levels) for most functions.

Even the number of bits available for control is not an exact measurement of quality or of versatility. Perhaps in your application, you only use attack rates ranging from 10 ms (milliseconds) to 200 ms. Then a synthesizer with 32 levels (5 bits) and a range from 10 ms to 250 ms might be a better deal for you than one with 512 levels

(9 bits) ranging from 8 ms to 10,000 ms (10 seconds); especially considering that the one with only 5 bits probably less expensive. Another factor is that maybe only every 10th level is useful in your application; a synthesizer with fewer levels at the needed spacing would be better.

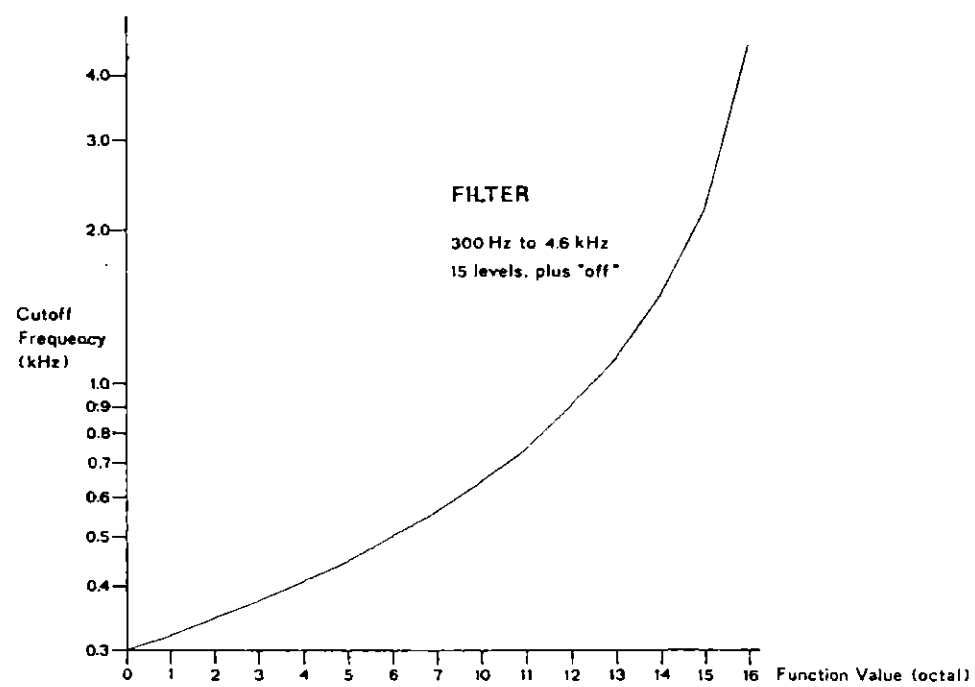
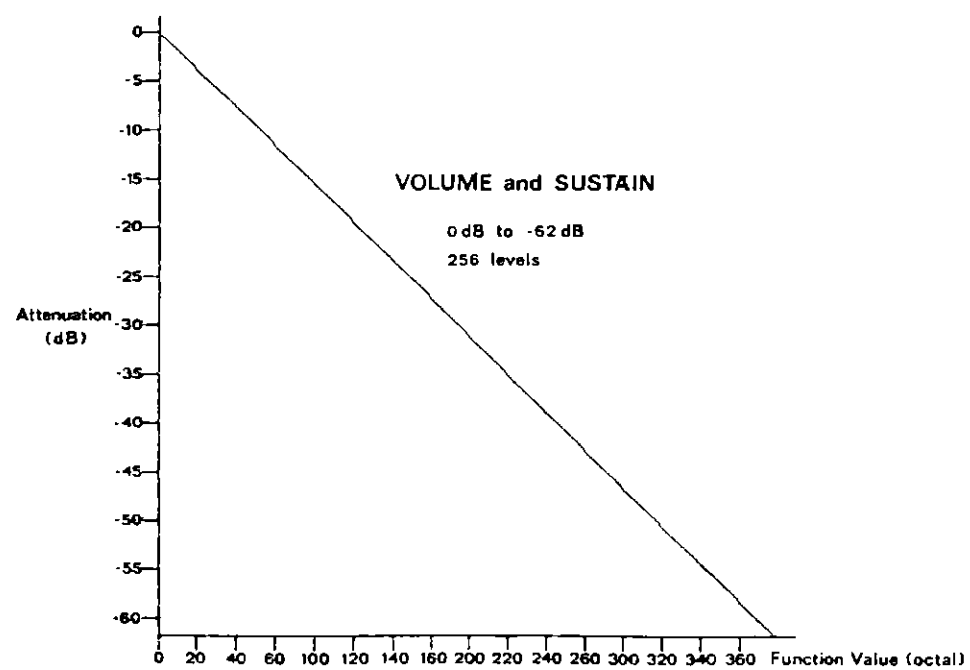
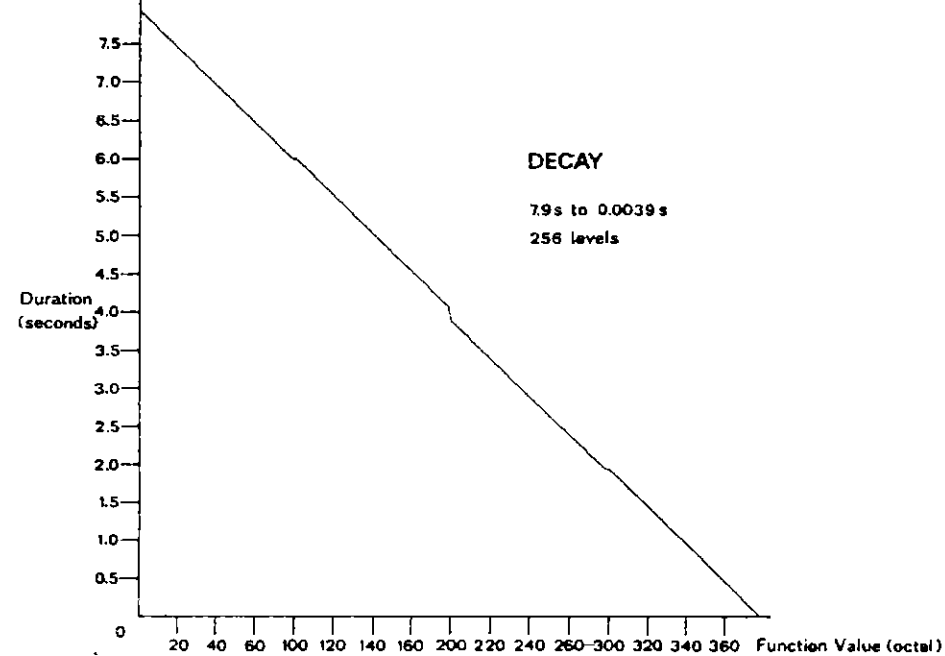
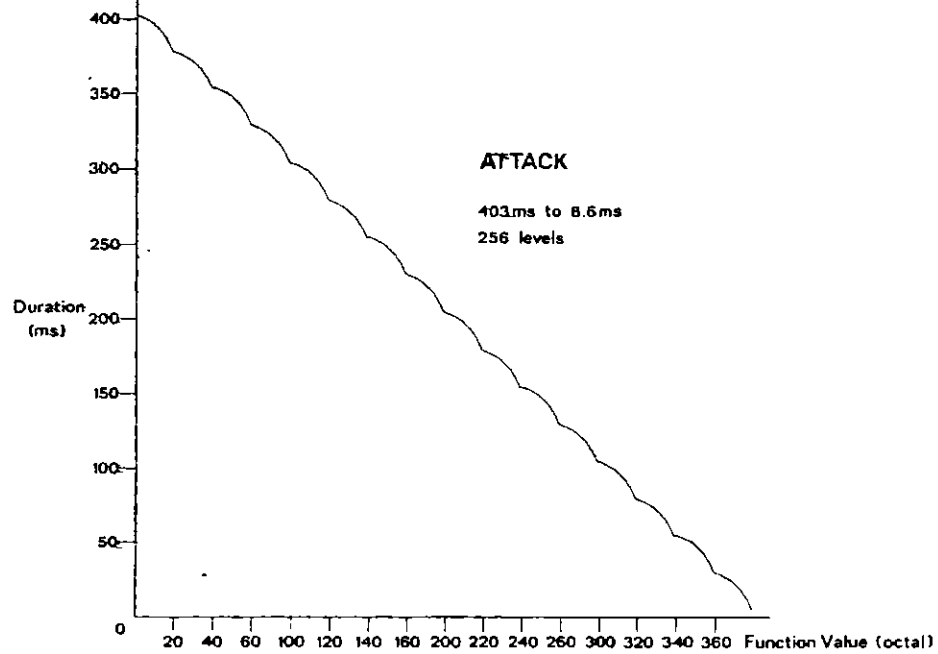
It is usually the case that each company thinks that its choice of control (both range and levels) is best. Any fewer levels, they claim, and there wouldn't be enough levels. Any more levels, and the unit would be more expensive without adding perceptible improvements. Expand the range, and there would be too great a distance between levels; compress the range, and necessary values would be eliminated. Each designer makes several tests and establishes the levels and ranges based on the desired applications and the desired price ranges.

Therefore the consumer must decide for himself which is best for his needs. Perhaps the best way to do this is to arrange for a demonstration, although this is not always possible and it may take you quite a while to learn enough about each unit to gauge its usefulness. Another way is to listen to demonstration tapes or records. Keep in mind that only a certain amount of material can be presented on a demo tape, and therefore not very many of the synthesizer's capabilities can be shown. The most important thing to listen for may be the general over-all sound quality. There shouldn't be a lot of noise (you must exclude tape or record noise); volume changes should not always be in large steps, and be wary if every sound is harsh or raspy or if "steps" are always evident in changes.

Information from data sheets and owner's manuals can also be useful in determining the usefulness of a synthesizer. In this paper, graphs of various AD8 functions are shown. These can give you a general idea of the range of each function.

As in any synthesizer, only a certain number of functions are included. This allows the cost of the synthesizer to be in some desired range. It is important that provisions

Cont. on page

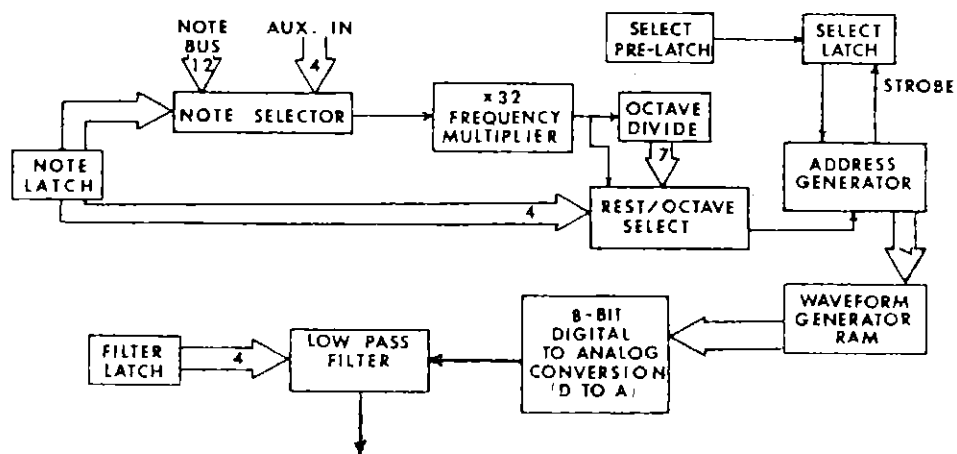


are made to add additional functions later. This allows you to start with a minimal system, and decide which other functions you need after you're used to the system. The AD8 series is based on a bus design which allows additional boards to be plugged into the bus. Connectors on the synthesis boards themselves allow for the connection of accessories such as portamento, pitch bend, echo, reverb, etc.

AD8 SERIES SYNTHESIS DETAILS

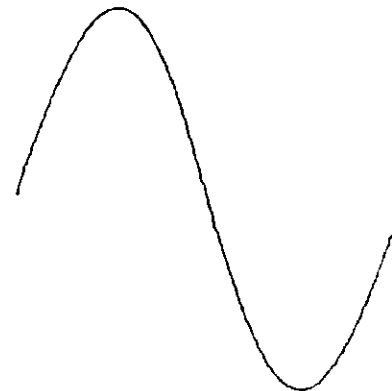
The typical AD8 synthesis board has two sections which are independent until the very end of the signal path. One controls the pitch and waveform, and the other the envelope and volume.

PITCH AND WAVEFORM

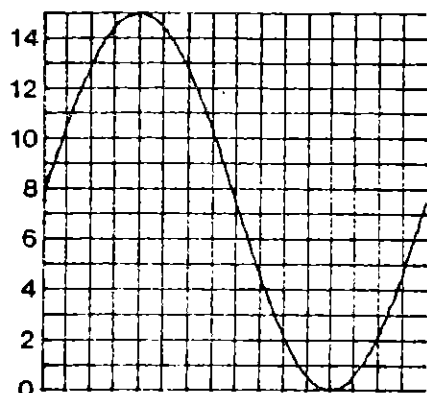


When the computer sends the number to select the pitch, it is stored in the "note latch". Part of this number is sent to the note selector, which selects one of the 12 pitches in an octave. The selected pitch is then multiplied by 32 and put into the octave dividers. The other half of the number from the note latch is sent to the rest/octave selector which selects the proper octave or a "rest" (no tone). The output of the rest/octave

selector is a frequency 32 times the desired pitch. This frequency is run into an address generator which goes into a 128 element Random Access Memory (RAM). Since both edges of the 32 times frequency are used, 64 elements of the RAM are addressed in a single cycle of the pitch. Each element of the RAM has 8 bits which allow a number from 0 to 255 to be remembered. (The numbers in the RAM were written into the RAM by the external computer system.) The number from each element of the RAM is put into a Digital to Analog Converter (DAC) which changes the number into a voltage. This combination of the address generator, RAM, and DAC is called a scanned-RAM voltage generator (SRVG). The object of this device is quite simple. For example, we may wish to generate a sine wave which looks like this:

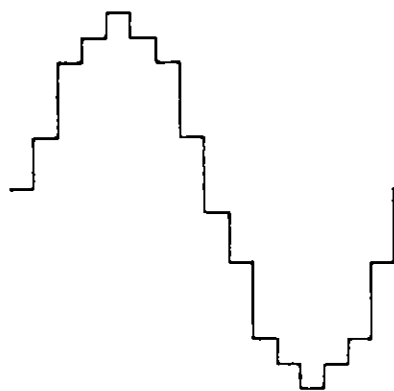


To simplify the example, let's assume a SRVG with 16 elements (rather than 64) and only a number from 0 to 15 (rather than 0 to 255) in each element. First, we divide the sine wave into equal time intervals (the vertical lines in the following drawing) and draw lines dividing the vertical area into equal spaces (the horizontal lines):



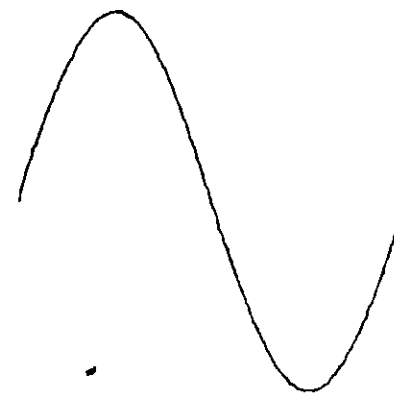
The height of the sine wave at any of the vertical lines is called its amplitude. The amplitudes of the sine wave at the 16 points are: 8, 10, 13, 14, 15, 14, 13, 10, 7, 5, 2, 1, 0, 1, 2, and 5. These numbers are written into the memory, and to produce a sound the memory is scanned over and over again. The numbers read from the memory as it is scanned are sent to the DAC. Note that the grid drawn over the sine wave (above) extends further to the right than the last amplitude value used. If the next vertical line were to be drawn, it would represent the first element of the RAM again, as it is scanned the next time.

The DAC produces a voltage which looks like this:



This is very much like a sine wave except it is produced in "steps". Obviously, the more steps (or elements in the RAM) there are and the larger the range the amplitude numbers can be, the more accurate the reproduction will be. Therefore, the AD8 uses 64 elements and 256 different amplitude values.

To produce a different waveform than sine, you simply use the amplitude numbers for the wave you wish to produce. To further increase the accuracy of the waveform generator, the output of the DAC goes into a programmable Low-Pass Filter. This filter, which the computer programs to different cut-off frequencies depending on the pitch being produced, reduces the steps in the waveform. On waves with sharp edges (such as a square wave) the filter can be turned off. After the filter, the sine wave of the previous example would look something like this:



This compares well with the original sine wave which we were attempting to produce. An actual oscilloscope photo of a sine wave on the AD8 before and after the filter is given on page 18.

You may have noticed that there are 128 elements in the RAM, but only 64 are scanned for a waveform. This allows two different waveforms to be defined in the RAM at once so that one can be reprogrammed to a new sound while the other is actually being used to produce the current

sound. Without a dual memory, it would be necessary to reprogram the sound while it was being produced (which would be audible) or only change the waveform during a rest. An additional difficulty is encountered on the switch over from one half of the memory to the other.



being played

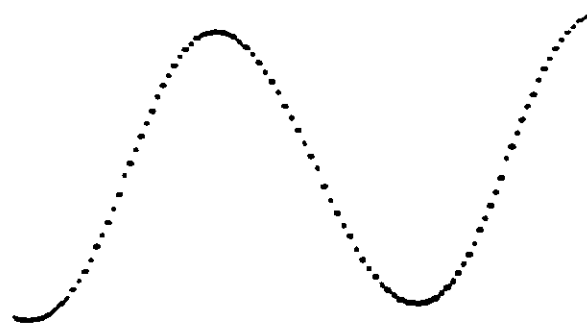
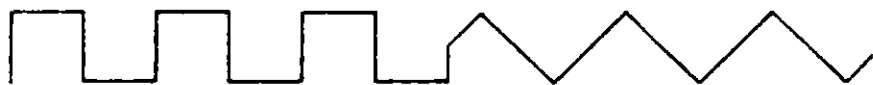


stored

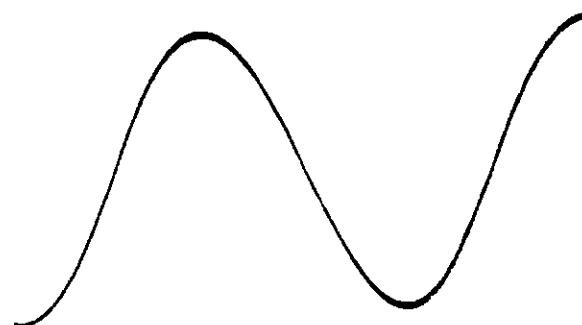


random switch-over

As shown above, if the switch from one waveform to another is done at a random time (such as switching as soon as the command to switch is sent by the computer) the result can be a sharp transition. The Select Pre-Latch and Select Latch allow the switch-over to be delayed until the start of the waveform definition. This guarantees a predictable switch-over, and allows the start (or zero crossing point) of all waveforms to be defined at the switch-over point, resulting in a smooth transition:

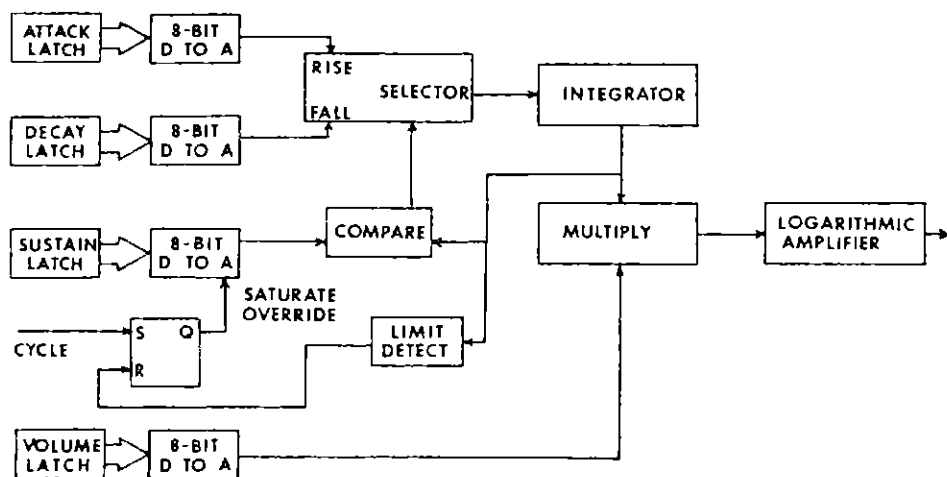


AD8 sine wave at Middle C
before filter



AD8 sine wave at Middle C
filter setting: 15 octal

ENVELOPE AND VOLUME



The three parameters used by the envelope generator (attack rate, decay rate, and sustain level) are stored in the Attack Latch, the Decay Latch, and the Sustain Latch. All three numbers are turned into voltages by 8-bit DACs (Digital to Analog Converters). The general operation of the envelope generator is as follows.

The sustain level and the current loudness level are compared, and the result put into the rise/fall selector. If the current loudness is less than the sustain level, "rise" is selected and the output of the rise/fall selector is the attack rate. If the current loudness is greater than the sustain level, "fall" is selected and the output of the rise/fall selector is the decay rate.

This output is put into an integrator which causes the loudness to rise or fall at the selected rate. If the current loudness is equal to the sustain level, the loudness will stay the same. Thus, as the sustain level is changed, the loudness will change to match the sustain level, but only at the rates allowed by the Attack Rate and the Decay Rate.

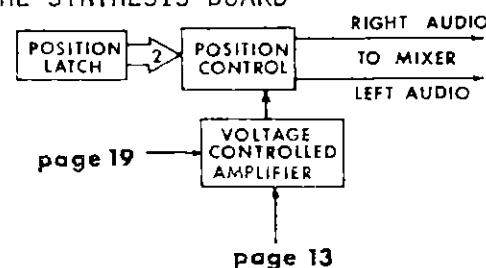
The "cycle" input causes the output of the Sustain Level DAC to be fully high. Thus, if the computer sends a

Cycle command, the loudness will go to the highest level at the attack rate. Once the loudness is at the highest level, the Limit Detector causes the Sustain Level DAC to go back to normal. This scheme allows complex envelopes to be created by setting or altering the Attack and Decay rates, and changing the Sustain Level to determine the contour. The Cycle command is used for envelopes which go fully high before going to another level, with automatic switching between attack and decay

For the usual attack/decay/sustain/release envelope, the Cycle command simplifies envelope production. Prior to starting a note, the Attack and Decay rates and Sustain Level are set as desired. The pitch is changed (if necessary) and a Cycle command is issued. The synthesis board will then automatically perform the attack, decay, and sustain portions. When the end of the note is to occur, the release portion is started by changing the Decay Rate to the desired release rate (if different than the normal decay rate) and setting the Sustain Level to zero.

The desired volume level is stored in the Volume Latch. An 8-bit DAC converts this number to a voltage which is multiplied with the signal from the integrator (the current loudness). The result of the multiplication is put through a Logarithmic Amplifier.

REMAINDER OF THE SYNTHESIS BOARD



The outputs of the Logarithmic Amplifier and the Low-Pass Filter are connected to a Voltage Controlled Amplifier. The output of this goes to the Position Control which sends the audio signal to the left or right mixer, or to both or neither. Connectors allow for more extensive stereo positioning with accessories.

COMPLETE SYSTEMS

Presently, there are two basic models of the AD8. The first model, called simply the AD8, uses a bus construction which is familiar to most computer users. All synthesis boards and the controller board plug into sockets on a backplane board (or "motherboard") which connects the signals of the various boards together, and supplies power (from a power supply connected to the backplane) to all boards. The controller board has a cable which connects to a SIMP-A interface in the external computer. With this system, the AD8 boards plug into their own backplane (separate from the external computer's backplane) and generally have their own cabinet (with card frame) and power supply.

A less expensive approach is used in the other basic model, called the AD8 Micro-Bus system. The adapter board plugs directly into an S-100 computer system, and ribbon cables run outside the computer to the synthesis boards. Two cables are used, one for the top octave frequency sources (the "note bus"), and one for the control signals (the "Micro-Bus"). The synthesis boards are held together by spacers forming a stack of boards. The ribbon cables connect to each board, providing the signals provided by the backplane in the AD8 system. The power is supplied by a separate cable connected with Molex connectors to each board. The power may be supplied by the S-100 system's power supply, or by a separate power supply.

The circuitry of the synthesis boards is the same in both models. Both models can accept accessory boards plugged into the backplane or to added ribbon cables. In addition to the controller board (or adapter board) and power supply requirements explained above, each AD8 system consists of an external computer and one to eight synthesis boards. Although computer requirements vary depending on the application, any computer system with 16 K bytes or more of memory should be sufficient; interrupt capability is quite useful.

The number of synthesis boards you need depends on your application. Each board can play only one tone at a time. If a song uses four tone chords plus a melody line, you would need five synthesis boards to produce that song. More synthesis boards are required if the release of one tone overlaps the attack of the next tone, as two boards would be required to perform that effect.

Each synthesis board responds to different function codes, so that they may each be programmed independently. Boards are identified by an integer from 0 to 7, and two boards with the same number (or "address") are generally not used in a single system. There are two methods of indicating the address of a board. The standard method uses a header with jumpers soldered to it. There are 8 such headers (for 0 through 7) and they are plugged into a board to establish the number of that board. The optional method is a DIP switch which allows the address of a board to be set by a switch on the board. Usually, synthesis boards are assigned numbers from 0 up, adding 1 to the number of each board to create the number of the next board.

"Controller" boards contain a microprocessor to aid in music production, "adapter" boards do not. At present, the AD8 system is available only with a SIMP-A compatible controller board; the AD8 Micro-Bus system is available only with an S-100 compatible adapter board. (S-100 products will work only in S-100 systems, see owner's manuals for details. SIMP-A products are compatible with the Standard Interface for Micro-Peripherals as defined in CSL publication #1-77A, available from ALF for \$1. SIMP-A is a ten-bit parallel interface suitable for use with PIAs and other digital interface circuitry.) Planned products include a SIMP-A adapter board for the AD8 series, and SIMP-A adapter and controller boards for the AD8 Micro-Bus series.

At this time, only the basic systems are available. Planned accessories include a 71 key digital manual (organ keyboard), performance oriented digital controls (knobs), reverb and echo accessories, and a unique noise source. Other accessories may also be produced.

DISCUSSION

The analog circuitry of the envelope generator was chosen for its smooth (step free) envelope production and ease of programming. Many synthesizers use the scanned-RAM technique for envelope production. There are advantages to this method when it is implemented correctly, but the cost is prohibitive. For satisfactory results, a very large RAM is required along with variable speed scanning circuitry and special smoothing filters. To produce the usual attack/decay/sustain/release envelope, it is necessary to either run through the RAM for the attack/decay/sustain portion (holding at the last element for sustain) then reprogram for release and run through again; or have a special circuit which allows both to be stored and allow scanning to stop at the end of sustain, then resume for release. The second method is more desirable (since less data transmission is required) but requires an even larger RAM. The number of elements in the RAM in either case should be much greater than the number of elements acceptable in a waveform generator, and the scanning speed would require precise control over a broad range of values. Additionally, large RAMs require many bits of addressing when sending new data to the RAM. This limits the number of bits available for sending data to accessories. We feel that the simplicity of the AD8's analog envelope generator is advantageous compared to the scanned-RAM method. Although the scanned-RAM method can, when it is properly implemented, result in increased versatility, it is much more difficult to program in terms of the amount of transmission required. When producing a complex song while also interpreting a large number of inputs from manual controls, the external computer's processing time can become quite valuable, especially with the limited speed of modern inexpensive microcomputers.

Many people have suggested that only a DAC and an output port is necessary to create music. To match the precision of the AD8 at the highest note, over 425,000 bytes per second of output would be required. (This is the number of bytes sent to DAC's per synthesis board in an AD8 system for the highest pitch.) This is most likely

beyond the capabilities of the external computer; and if pre-stored data is used to reduce calculation time, a one minute song would require 24 megabytes of storage. (Most short songs require only a few K bytes of storage in formats usually used in AD8 systems.)

Most synthesizer work is of an experimental nature, expanding the limits of current technology in sound production. It is therefore important to demand very little from the external computer system so that experimental work can proceed with simple programs, written casually and inefficiently. Probably most such tests are written as patches into existing drivers and performance software. The AD8 is designed with this in mind, and the transmission requirements are kept to a minimum. There are systems which play one to four tones by sheer software. These systems are written to the limit, and are unable to take on additional tasks such as keyboard scanning (or producing that fifth tone). Most AD8 performances can be done by interrupts while the computer system is simultaneously running another program (such as an operating system or perhaps BASIC).

Unlike the four-tone software systems, the AD8 has virtually no upper limit in terms of simultaneous tone production; each controller or adapter can connect to one to eight synthesis boards, and several adapters can be used in one computer system. You may not want to create whole symphonies now, but at least the capability is there should you ever need it.

Also available: Quad Chromatic Pitch Generators for a low cost start in music synthesis. Each board contains the pitch selection circuitry of 4 AD8 boards, and has provisions for adding accessory boards with additional AD8 functions for future expansion. Write to ALF for data on these products.

APPENDIX G

```

>
*** SYNTAX ERR
>L
10 GOSUB 32000:V1=32766:V2=32764

20 M=12
30 DD=32760
100 V=V1: GOSUB 1000: GOSUB 1020
   : GOSUB 1010: GOSUB 1010: GOSUB
   1030: GOSUB 1020: GOSUB 1020

110 GOSUB 1070: GOSUB 1060: GOSUB
   1070: GOSUB 1040: GOSUB 1020
   : GOSUB 1000: GOSUB 1010: GOSUB
   1020: GOSUB 1030

120 GOSUB 1040: GOSUB 1050: GOSUB
   1040: GOSUB 1030: GOSUB 1020
   : GOSUB 1010

130 V=V2: GOSUB 1040: GOSUB 1000
   : GOSUB 1100: GOSUB 1090: GOSUB
   1080

135 M=(2*M)/3
140 K=1020: GOSUB DD:K=1000: GOSUB
   DD

150 K=1010: GOSUB DD:K=1020: GOSUB
   DD:K=1040: GOSUB DD:K=1030:
   GOSUB DD: GOSUB DD
   K=1050: GOSUB DD:K=1040: GOSUB
   DD: GOSUB DD:K=1070: GOSUB
   DD:K=1060: GOSUB DD:K=1070:
   GOSUB DD

170 K=1040: GOSUB DD:K=1020: GOSUB
   DD

180 M=(3*M)/2:V=V2: GOSUB 1000:
   GOSUB 1040: GOSUB 1030: GOSUB
   1020: GOSUB 1010: GOSUB 1000

190 V=V1: GOSUB 1040: GOSUB 1070
   : GOSUB 1060

200 M=M*16:V=V2: GOSUB 1000
210 FOR I=1 TO 3000: NEXT I
220 M=6
230 V=V2
240 GOSUB 1070: GOSUB 1060:M=M*
   2: GOSUB 1070
250 M=M/2: GOSUB 1040: GOSUB 1030
   :M=M*2: GOSUB 1040:M=M/2: GOSUB
   1070: GOSUB 1060:M=M*2: GOSUB
   1070
260 M=M/2: GOSUB 1020: GOSUB 1010
   :M=M*2: GOSUB 1020
270 M=M/2: GOSUB 1070: GOSUB 1060
   :M=2*M: GOSUB 1070:M=M/2: GOSUB
   1000: GOSUB 1010:M=M*2: GOSUB
   1020: GOSUB 1035
280 M=M/2: GOSUB 1040: GOSUB 1035
   : GOSUB 1040: GOSUB 1050: GOSUB
   1040: GOSUB 1060: GOSUB 1040
   : GOSUB 1070
290 GOSUB 1040: GOSUB 1035: GOSUB
   1040: GOSUB 1050: GOSUB 1040

```

A. Software Synthesis
Program for Use With
the Apple II Computer

```

300 GOSUB 1080: GOSUB 1040: GOSUB
    1090: GOSUB 1040: GOSUB 1035
    : GOSUB 1040: GOSUB 1050: GOSUB
    1040: GOSUB 1100: GOSUB 1040
    : GOSUB 1110
310 M=M*2: GOSUB 1090:M=M/2: GOSUB
    1090: GOSUB 1070:M=M*2: GOSUB
    1090:M=M/2: GOSUB 1070: GOSUB
    1060:M=M*2: GOSUB 1070:M=M/
    2
320 GOSUB 1060: GOSUB 1050: GOSUB
    1040: GOSUB 1070: GOSUB 1040
    : GOSUB 1070:M=2*M: GOSUB 1050
    :M=M/2: GOSUB 1040: GOSUB 1030
    : GOSUB 1020: GOSUB 1070: GOSUB
    1020: GOSUB 1070
330 M=M*2: GOSUB 1030:M=M/2: GOSUB
    1020: GOSUB 1010: GOSUB 1000
    : GOSUB 1070: GOSUB 1010: GOSUB
    1070: GOSUB 1020: GOSUB 1070

335 GOSUB 1030: GOSUB 1070
340 GOSUB 1040: GOSUB 1060: GOSUB
    1040: GOSUB 1070: GOSUB 1040
    : GOSUB 1090: GOSUB 1040: GOSUB
    1090
350 GOSUB 1040: GOSUB 1100: GOSUB
    1040: GOSUB 1110:M=M*2: GOSUB
    1090
360 M=M/2: GOSUB 1090: GOSUB 1070
    :M=M*2: GOSUB 1040: GOSUB 1060
    : GOSUB 1070:M=M/2
... GOSUB 1060: GOSUB 1050: GOSUB
    1040: GOSUB 1030: GOSUB 1020
    : GOSUB 1010:M=2*M: GOSUB 1020
    :M=M/2: GOSUB 1010: GOSUB 1000

380 M=M*3: GOSUB 1040: GOSUB 1040
    :M=M*3: GOSUB 1000
390 FOR I=1 TO 3000: NEXT I
400 M=0
410 V=V1
420 GOSUB 1000: GOSUB 1000: GOSUB
    1010: GOSUB 1020: GOSUB 1000
    : GOSUB 1020:M=2*M: GOSUB 1010

430 V=V2:M=M/2: GOSUB 1070: GOSUB
    1070: GOSUB 1080: GOSUB 1090
    :M=M*2: GOSUB 1070: GOSUB 1060

440 M=(2*M)/6
450 K=1070: GOSUB DD: GOSUB DD:
    K=1080: GOSUB DD:K=1090: GOSUB
    DD:K=1100: GOSUB DD:K=1090:
    GOSUB DD:K=1080: GOSUB DD:
    K=1070: GOSUB DD:K=1060: GOSUB
    DD
    K=1040: GOSUB DD:K=1050: GOSUB
    DD:K=1060: GOSUB DD:M=(6*M)
    /2:V=V1: GOSUB 1000:V=V2: GOSUB
    1070
470 FOR I=1 TO 3000: NEXT I
480 V=V1:M=12: GOSUB 1070:M=M/3
    : GOSUB 1040: GOSUB 1035: GOSUB
    1040: GOSUB 1045: GOSUB 1055

```

```

: GOSUB 1045: M=M*3: GOSUB 1060
: M=M/3: GOSUB 1060
490 M=M*5: GOSUB 1070
500 FOR I=1 TO 3000: NEXT I: V=V1:
M=12
510 M=M*3: GOSUB 1110: GOSUB 1090
: GOSUB 1070: GOSUB 1040: M=
M/3: GOSUB 1050: GOSUB 1060
: GOSUB 1070: M=2*M: GOSUB 1050
: M=M/2: GOSUB 1070
520 M=M*6: GOSUB 1040
530 V=V1: M=M/2: GOSUB 1080: GOSUB
1110: GOSUB 1090: GOSUB 1070

540 V=V2: M=M/3: GOSUB 1050: GOSUB
1060: GOSUB 1070: M=M*2: GOSUB
1080: M=M/2: GOSUB 1090: M=M*
5: GOSUB 1080
550 M=M/5: GOSUB 1090: GOSUB 1100
: GOSUB 1090: GOSUB 1080: M=
M*2: GOSUB 1110: M=M/2: GOSUB
1090: GOSUB 1080: M=M*4: GOSUB
1070
997 FOR I=1 TO 3000: NEXT I: GOTO
20
998 LIST 1,999
999 END
1000 I=250: GOTO V
1010 I=220: GOTO V
1020 I=200: GOTO V
1030 I=185: GOTO V
1035 I=175: GOTO V
1040 I=165: GOTO V
1045 I=145: GOTO V
1050 I=145: GOTO V
1055 I=137: GOTO V
1060 I=130: GOTO V
1070 I=122: GOTO V
1080 I=110: GOTO V
1090 I=99: GOTO V
1100 I=94: GOTO V
1110 I=82: GOTO V
1120 I=72: GOTO V
9987 LIST 1,999
32000 POKE 16128,165: POKE 16129,
5: POKE 16130,133: POKE 16131
,7: POKE 16132,165: POKE 16133
,4: POKE 16134,133: POKE 16135
,6
32010 POKE 16136,165: POKE 16137,
0: POKE 16138,133: POKE 16139
,2: POKE 16140,133: POKE 16141
,1: POKE 16142,173: POKE 16143
,48
32020 POKE 16144,192: POKE 16145,
165: POKE 16146,1: POKE 16147
,233: POKE 16148,1: POKE 16149
,133: POKE 16150,1: POKE 16151
,208
32030 POKE 16152,5: POKE 16153,165
: POKE 16154,0: POKE 16155,
76: POKE 16156,12: POKE 16157
,63: POKE 16158,165: POKE 16159
,2
32040 POKE 16160,233: POKE 16161,
1: POKE 16162,133: POKE 16163

```

```

,2: POKE 16164,208: POKE 16165
,235: POKE 16166,173: POKE
16167,48
32050 POKE 16168,192: POKE 16169,
165: POKE 16170,0: POKE 16171
,133: POKE 16172,2: POKE 16173
,165: POKE 16174,7: POKE 16175
,233
32060 POKE 16176,1: POKE 16177,133
: POKE 16178,7: POKE 16179,
208: POKE 16180,220: POKE 16181
,165: POKE 16182,5: POKE 16183
,133
32070 POKE 16184,7: POKE 16185,165
: POKE 16186,6: POKE 16187,
233: POKE 16188,1: POKE 16189
,133: POKE 16190,6: POKE 16191
,208
32080 POKE 16192,208: POKE 16193,
96: POKE 16194,165: POKE 16195
,0: POKE 16196,133: POKE 16197
,4: POKE 16198,165: POKE 16199
,1
32090 POKE 16200,133: POKE 16201,
5: POKE 16202,165: POKE 16203
,2: POKE 16204,133: POKE 16205
,6: POKE 16206,165: POKE 16207
,3
32100 POKE 16208,133: POKE 16209,
7: POKE 16210,173: POKE 16211
,48: POKE 16212,192: POKE 16213
,165: POKE 16214,5: POKE 16215
,233
32110 POKE 16216,1: POKE 16217,133
: POKE 16218,5: POKE 16219,
208: POKE 16220,250: POKE 16221
,165: POKE 16222,4: POKE 16223
,233
32120 POKE 16224,1: POKE 16225,133
: POKE 16226,4: POKE 16227,
208: POKE 16228,240: POKE 16229
,165: POKE 16230,0: POKE 16231
,133
32130 POKE 16232,4: POKE 16233,165
: POKE 16234,1: POKE 16235,
133: POKE 16236,5: POKE 16237
,198: POKE 16238,7: POKE 16239
,208
32140 POKE 16240,225: POKE 16241,
165: POKE 16242,3: POKE 16243
,165: POKE 16244,6: POKE 16245
,233: POKE 16246,1: POKE 16247
,133
32150 POKE 16248,6: POKE 16249,208
: POKE 16250,215: POKE 16251
,96: RETURN
32760 M=M/2:V=V1: GOSUB K:V=V2: GOSUB
K:M=M*2: RETURN
32 Q=32767/(40*I)*M: POKE 0,I MOD
255+1: POKE 1,I/255+1: POKE
4,Q MOD 255+1: POKE 5,Q/255
+1: CALL 16128: RETURN
32766 I=(20*I)/19:Q=32767/(10*I)*
M: POKE 1,I MOD 255+1: POKE
0,I/255+1: POKE 3,Q MOD 255

```


APPENDIX H

PAIA 8700 COMPUTER/CONTROLLER



8700 Processor: 6503 MPU. Wear free "ActiveKeyboard", Micro-Diagnostic. Extensive documentation, Fully Socketed.

Piebug Monitor: User Subroutines, Relative address calculator, Pointer High-low, Back-step key.

Cassette Interface: Load & Dump by file #, Positive indication of operation, Tape motion control.

The Answer For... Student, Hobbyist, Manufacturer.

An exceptional price on an applications oriented 6503 based microprocessor system featuring: 1K bytes RAM locations (512 bytes supplied), 1K bytes ROM locations (256 byte monitor included), two 8 bit input ports, two 8 bit output ports, one latched and one buffered.

A 24 key touch operated keypad is used by the monitor to allow entry and execution of user programs as well as controlling features not normally found on low-cost single board computers; including a relative address calculator that completely eliminates this normally tedious hexadecimal calculation and back-space key that eases entry and editing of programs. Pointer High and Pointer Low keys allow the 8700's twin seven segment displays to serve the multiple functions of indicating both address location and data.

The 8700 fits in a space reserved in the 8782 encoded keyboard's case. PAIA software support available for Electronic Music Synthesizer interface.

8700 COMPUTER/CONTROLLER KIT \$149.95

wt. 4 lbs.

CASSETTE INTERFACE OPTION

This is one of the most reliable and easy to use cassette systems that we've seen. A single LED indicates proper cassette volume control setting and provides a positive indication of data flow. We've even made software and hardware provision for tape motion control (relays must be added). The CS-87 option fits entirely on the 8700 circuit board and consists of the POT-SHOT PROM and a handful of additional components.

CS-87 Cassette Interface Option

\$22.50 plus postage

POWER SUPPLY

The logical choice to power the 8700. Fully regulated 5v. at 1 amp, -9 volts @ 300 ma. Also provides a 60 Hz. output for real-time clock applications.

PS-87 Power Supply Kit

\$24.95

wt. 3 lbs.

DIGITAL INTERFACE MODULES



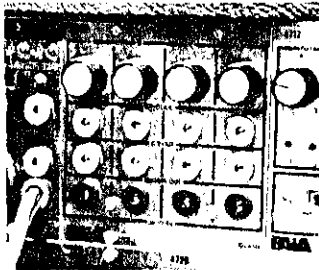
EQUALLY TEMPERED DIGITAL TO ANALOG CONVERTER

Unlike more conventional R-2R ladder type digital to analog converters, the PAIA 8780 kit is based on a multiplying principle that allows the module to generate the exact exponential stair-step function required to make even the simplest linear response oscillators and filters produce equally tempered musical intervals. The 8780 uses only 6 bits of data to generate over five octaves of control voltage. In an 8 bit system, the remaining two bits are ordinarily reserved for trigger flags, but may be used to provide micro-tonal tunings.

The module is physically and electrically compatible with the complete line of PAIA music synthesizer modules and is easily interfaced to any micro-processor.

8780 Digital to Analog Converter

\$34.95 wt. 1 lb.

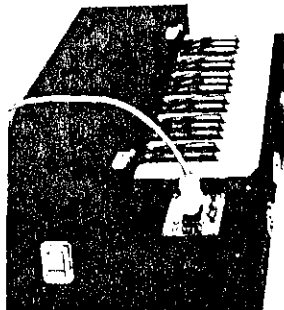


QuASH (Quad Addressable Sample and Hold)

The least expensive way to provide multiple control voltage channels in a computer based synthesizer system is to multiplex the output of the D/A using multiple, computer addressable, sample and hold circuits. The QuASH provides four of these outputs in a single module. Other features of the module include: adjustable glide rate for each channel with the glide selected or de-selected under computer control, individual trigger "gate" signal associated with each output channel and an individual modulation input for each channel that is non-interactive with other channels. On-board address decoding allows up to four QuASH to be bussed together in a single system without external logic and Bank Select input allows external logic expansion beyond this point. NOTE: THE QuASH MUST BE OPERATED UNDER COMPUTER/PROCESSOR CONTROL.

8781 Quad Addressable Sample & Hold Kit

\$34.95 wt. 1 lb.



DIGITALLY ENCODED KEYBOARD

A scanning, matrix encoder tied to a 37 note AGO keyboard provides 6 bits of data and both STROBE and STROBE control outputs. Input control lines to the encoder include SCAN (starts and stops the encoder clock), RESET, START and RANDOM making the keyboard universally applicable to all computer/processors from the very largest to the very smallest. Housed in a trim and sturdy vinyl covered road case, the kit consists of all parts including keyboard, power supply and detailed assembly instructions; software overview for computer applications and detailed instructions for digital sample and hold.

8782 ENCODED KEYBOARD KIT

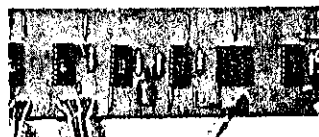
\$119.95

(shipped freight collect)

KEYBOARD ENCODER CIRCUIT

Convert PAIA or other keyboards to the new digital format.

details on page 23



****WANT TO KNOW MORE? ****

PAIA's FRIENDLY STORIES ABOUT COMPUTERS/SYNTHESIZERS

Mating a computer with an electronic music synthesizer presents almost unlimited possibilities for performer and composer alike, and true polyphonic synthesis is only the beginning. Let us tell you the whole story with this package of off-prints from Polyphony, using manuals and more, telling how it works and what it does.

4

I-COMP Friendly Stories about Computer/Synthesizers

\$3.00 postpaid

PROGRAMS

PAIA 8700 SOFTWARE

PAIA software is written to produce a system that is as easy to use as possible. In all cases, program features are controlled directly from the 8700 keyboard (for example, with DRUMSYS the RUN key plays the stored score; TAPE loads scores from tape, etc.) Each package consists of tape and full documentation including program listings and interfacing details where required.

PMUS

Polyphonic synthesizer (with sequencing and software transient generators) multi-track recording/playback, compositional programs, supports up to 16 voices. This cassette is FREE with MUS-1 PROM. (MUS-1 required for operation)

POLY

Polyphonic synthesizer program including software transient generators. Supports up to 16 output channels.

DRUMSYS

An 8700/EK-2 Computer Drums operating system that produces the most advanced automatic percussion unit ever offered. Provides all of the features of a 3750 Programmable Drum Set plus dumps and loads drum scores to tape, computer control of drum dynamics, extensive editing of scores, special effects that make drums almost a lead instrument and unique tempo setting control that requires only the tap of a finger.

SEQUE 1.0

The ultimate sequencer. Store sequences from the keyboard AS YOU PLAY THEM. Use the sequence as a voice as you transpose it from the keyboard. Easily controlled retrograde and inversion capabilities make 12 tone row compositions a snap. Save and load sequences using easily managed cassette tapes. No tempo indicator to watch—just play, then speed up or slow down as required. AND THESE ARE ONLY THE START.

Each of the above software and documentation packages includes tape, documented listings and interfacing details. Each is priced at \$4.95 plus \$1.00 postage and handling. Documentation, less tape is available for \$2.00 postpaid.

MUS-1 FIRMWARE

To optimize a system for music applications some commonly used programs are best stored in Programmable Read Only Memory. This is the PROM. MUS-1 contains keyboard reading routine, 16 channels of QuASH drivers, polytonic algorithm and software transient generator subroutines.

MUS-1 PROM

\$22.00 plus postage - (\$1.00)

** ATTENTION KIM FANS **

We're sort of KIM fans too (though, as you might expect, we feel that our own 8700 is more price effective in many cases — and in some ways easier to use). In addition to our exceptional PVI-KIM video display (page 12), we have this software for the KIM.

CASS's CODE

Cass Lewart has published over 50 technical articles within the past few years and here are some things he's done for the KIM. This double tape for the KIM/PVI system features a mini-disassembler which recognizes, organizes and displays OP-CODES and operands for one, two or three byte instructions. Displays in hex. The second program on the tape is a morse code teacher which generates random 5 character words and sends them in morse code. To check your accuracy, the word sent may be displayed on request. (requires speaker and driver as in KIM owners manual)

Cass's Code # CLP-1

\$4.95 plus postage

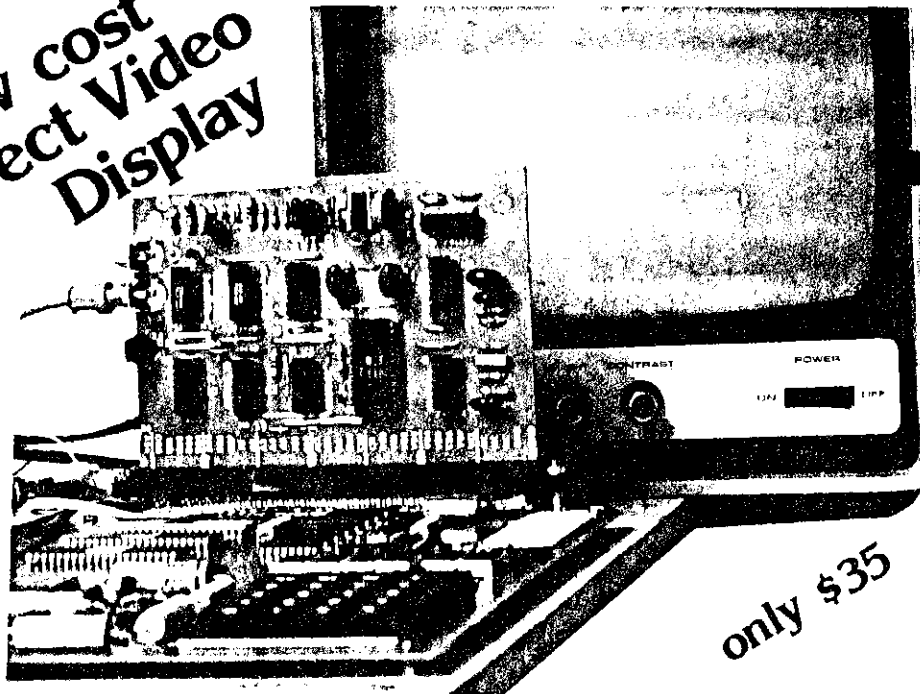
KIM MUSIC PACKAGE

The KIM also makes an excellent control system for PAIA synthesizers and this package provides interfacing details and documentation of programs for polyphonic synthesizers and sequencers, compositional programs and others.

KIM MUSIC PACKAGE # KMUS

\$4.95 plus postage

**Low cost
Direct Video
Display**



Don Lancaster's ingenious design provides software controllable options including:

- **Scrolling**
- **Over 2K on-screen characters with only 3MHz bandwidth**
- **Full performance cursor**
- **Variety of line/character formats including 16/32, 16/64 even 32/64**
- **User selectable line lengths**

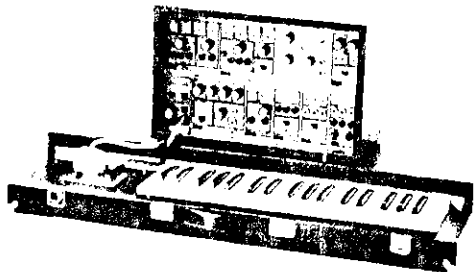
You'll want to see the operational details on this first. The PVI-1K is not the universal answer to every video display requirement. In applications where its minor limitations present insurmountable obstacles to a design, more expensive techniques should be used. If you are in doubt, the PVI is completely described in the July and August, 1977 issues of Popular Electronics. Reprints of these articles are the instruction set for this kit and are available separately for \$2.00 postpaid refundable upon purchase of the PVI kit.

Complete kit includes circuit board, all parts and instructions and is available in either of two forms.

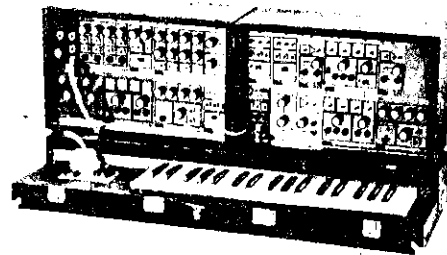
| | | |
|---|---------|-----------|
| PVI-KM Ready to go with KIM's | \$34.95 | wt. 1 lb. |
| PVI-MT For other processors (requires PROM coding) | \$34.95 | wt. 1 lb. |

... COST EFFECTIVE DESIGNS

computer controlled digital systems



4700/C



4700/J

4700/C

The ideal monotononic starter system. Can be used without a computer/processor as a conventional electronic music synthesizer. But, by simply unplugging the synthesizer head from the keyboard, a computer can be put into the loop to provide power and versatility never before possible for synthesizers of any cost. The 4700/C module complement includes the 8782 Encoded Keyboard, Digital to Analog Converter, 2720-5 Control Oscillator/Noise source, 4710 Balanced Modulator/VCA, Reverb, 4720 VCO, 4730 VCF, Envelope Generator, two Watt Block power supplies and a 4761 Wing Cabinet, complete step-by-step assembly instructions and using manual.

No. 4700/C Synthesizer Kit

\$325.00

(shipped freight collect)

4700/J

By anyone's standards this is a BIG synthesizer, as you can see by reviewing the module complement. Like our other packages, it may be used without a computer as a normal monotononic synthesizer. With a computer in the loop, you are ready to do polyphonic instruments, multi-track recording work, and innumerable composer and performer assisting functions that are only possible with a computer/synthesizer combination. The 4700/J module complement consists of: the 8782 Encoded Keyboard, Digital to Analog Converter, QuASH, two 4710 Balanced Modulator/VCA's, three 4720 VCO's, 2720-5 Control Oscillator/Noise Source, 4730 Filter, 4711 Stereo Mixer, two Envelope Generators, Reverb, three Watt Block power supplies and two Road Module Cabinets. Included are step-by-step assembly instructions and using manual.

No. 4700/J Synthesizer Kit

\$549.00

(shipped freight collect)

P-4700/C & P-4700/J

These P-4700 series packages pull it all together; synthesizer, computer and software ready to load from any cassette recorder and begin playing. Each package includes all of the synthesizer modules listed above as well as an 8700 Computer/Controller fully loaded with RAM, CS-87 cassette interface, power supply and all required hardware and connectors. Each represents a significant savings when purchased in this package configuration.

Music software and firmware provided with the P-4700/J includes both the MUS-1 PROM and PMUS cassette. The P-4700/C package includes the SEQUE 1.0 sequencer operating system.

P-4700/C Synthesizer with Computer Controller \$499.00
(shipped freight collect)

P-4700/J Synthesizer with Computer Controller \$749.00
(shipped freight collect)



keyboard with computer/controller
as featured in P-4700 packages.
(Cassette recorder not included.)

APPENDIX I



FRIENDLY STORIES ABOUT COMPUTERS/SYNTHESIZERS (Design Analysis)

- (1) WHAT THE COMPUTER DOES.
- (2) COMPUTER MUSIC, WITHOUT THE COMPUTER.
- (3) EQUALLY TEMPERED DIGITAL TO ANALOG CONVERTER
- (4) IN PURSUIT OF THE WILD QuASH
- (5) THE POLYPHONIC SYNTHESIZER
- (6) MUS 1 WITH THE NEW MIRACLE INGREDIENT - STG

WHAT THE COMPUTER DOES

The computer in our system does not itself generate any sound. It is simply acting as a performer/composer assisting control system for a more or less normal synthesizer. Providing what amounts to an extra set (or several sets) of hands.

From a system standpoint, it fits between the keyboard and synthesizer like this:

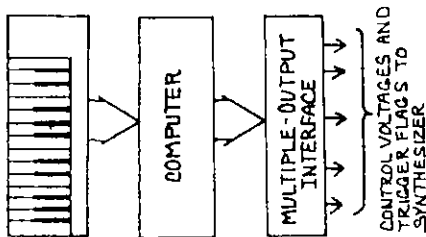


figure (a)

We said above, "more or less normal" synthesizer because there are three special elements involved in the synthesizer/computer interface:

- 1) a digitally encoded AGO keyboard (see "Computer music without the computer" and product summary)
- 2) a Digital to Analog Converter (see "Equally Tempered Digital to Analog Converter")
- 3) a multiple S/H circuit to allow several simultaneous outputs from the Digital to Analog converter.

The computer runs programs (either supplied by PAIA or user written) that receive data from the synthesizer keyboard and issue instructions to the D/A and multiple S/H which in turn control the synthesizer.

PROGRAMMING OVERVIEW

Just saying that the computer controls the synthesizer is hardly a satisfactory explanation of the system. Hardly satisfactory because it leaves out a

VERY IMPORTANT CONCEPT

which is that it is not really the computer that is controlling the synthesizer, it's the programs. In a very real sense, the computer is there only because it's a way to run the programs.

One of the programs (for example) "reads" the synthesizer keyboard and builds a table of what it finds there.

If the phrase "builds a table" is

unfamiliar to you, it simply means that when the program finds that a given key is down on the keyboard it records in a special place (location or address) in memory which key it is. The next key that it finds down, it records in the next memory location; and so on. When the program has finished looking at the entire keyboard the result is a list or "table" of the keys that were down during that scan. If you were holding down a C chord for example, the table might look like this:

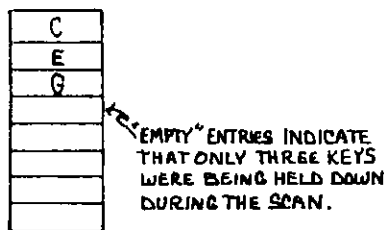


figure (b)

That's not really all there is to this program - there are some subtleties that would probably be confusing at this point. We'll get to them later. For right now, we'll just think of this program as a list-builder.

Also, so that I won't have to keep typing "the program that builds the list of keys that are down on the keyboard", we'll agree among ourselves that we'll call this program by the name "LOOK". From now on, when I say something like "we LOOK at the keyboard" you'll know that I mean we "execute" (run) this program.

And, while we're hanging labels on things, we may also just as well name the list that LOOK generates "key-table", or, since I'm a lazy typist, just KTABLE.

Got that? LOOK builds KTABLE.

OK, next.

There is another program that we'll

call NOTEOUT, because it takes care of outputting the notes.

Like LOOK, this one can be stated in simple terms; it reads the first entry from a table and causes the D/A to convert that key data to a control voltage which it then strobes into the first S/H. It then gets the second entry from the table, converts it to a control voltage and assigns it to the second S/H. Gets the third entry, etc.

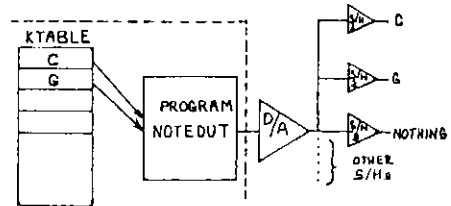
Also, like LOOK, there are subtleties that we'll look at later but the important point is that this routine works quickly. A block of 32 Sample and Holds can easily be refreshed and up-dated in about 16 ms. - more than fast enough.

The table that is read by NOTEOUT we will call the "note-table" or, simply NTABLE.

LOOK builds KTABLE and NOTEOUT reads NTABLE. Maybe you're wondering why two tables - why not just one.

Well, we could do it that way - if we did, a simplified diagram of the system should look like figure c.

You will recognize that we're still holding down that C chord. Now suppose we let the E go. On the next scan of the keyboard, LOOK up-dates KTABLE to reflect the fact that the E is no longer held down. KTABLE now looks like this:



THIS MIGHT NOT BE TOO BAD - MANY ORGANS DO NO MORE. figure (d)

And when NOTEOUT reads this table and up-dates the S/H circuits, guess what? The G has moved to the loca-

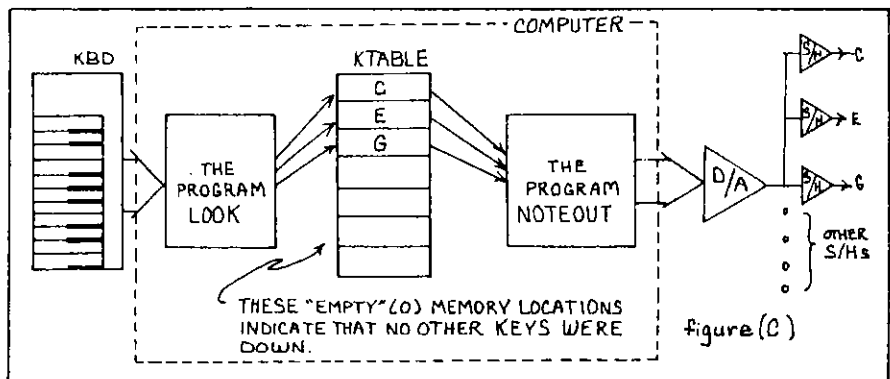


figure (c)

tion previously occupied by the E and from the S/H that previously was producing the control voltage for the G we now have nothing.

As if it weren't bad enough that the VCO which was previously producing an E is now playing a G (and we can hear when it makes this change), we can't do any decay processing on the E - the way a natural instrument would - because it's not there anymore.

Maybe this isn't too bad. A lot of organs produce results very similar to this - and all multiple output analog keyboards do this exact same "guess where the note's going to come out" trick. Still, it seems that there would be a more pleasing way to do it.

There is.

Because we're using two tables, we can generate a large (very large) family of programs that make decisions on how to transfer the information from KTABLE to NTABLE. This produces a machine which diagrammatically might look like this:

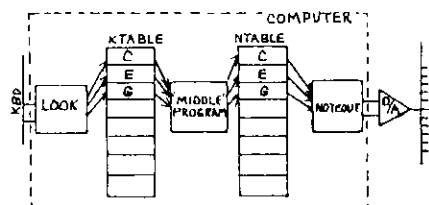


figure (e)

How this new middle program makes transfers from KTABLE to NTABLE determines completely the "personality" of the instrument.

For instance, a better way to handle the multiple -output problem would be to have the "middle" program not delete an entry from NTABLE simply because it no longer appeared in KTABLE, but rather to indicate that while the note should still be played, the key corresponding to it was no longer being held down and decay processing should begin. This is where the concept of "flags" associated with each note comes in and while it is slightly out of sequence, we should examine this important feature now.

The data that goes out to the synthesizer interface is a collection of 8 binary digits (bits - "1" or "0"). Like this:

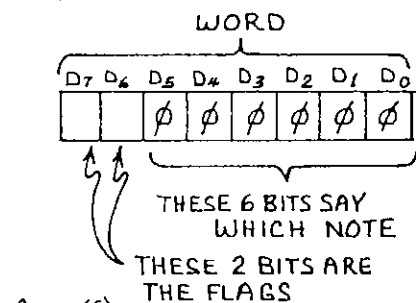


figure (f)

If we want to indicate to the synthesizer that the note that the data represents is one which currently corresponds to a key that is being held down on the keyboard, then we set bit #7 (D₇) to a "1". If the data does not correspond to a key that is currently down then this bit is a zero. As you can see, if you're already familiar with synthesizers, this flag bit corresponds to the "gate" signal that you get out of most synthesizer keyboards.

As you will see when you review the included 8780 information, both of these higher order bits are buffered and brought out to the front panel of the Equally Tempered Digital to Analog Converter.

This leaves us with a "left-over" flag that can be used in a variety of ways. It can, for instance, be used simply as an independent gate signal allowing the processor to select between one of two patching arrangements that we've set up. Or, and I believe that this is the preferable use, it can be used as a GLIDE SELECT bit that turns glissando on and off - under computer control.

But, to get back to the real subject at hand, the polyphonic output procedure described above is not the only (or, in my opinion, the most) interesting thing that the "middle" program can do.

It can examine the entries in KTABLE and if they are lower than a given note on the keyboard assign them to one group of outputs and if they are higher assign them to a second group of outputs. Which has the effect of "splitting" the keyboard into two different voices - one for low keys and a second for high keys.

The "middle" program can take notes from the keyboard and not only play them immediately, but also store them in another permanent table in the machine's memory for playback again later.

The "middle" program can take notes from the permanent table mentioned above, assign them to outputs and simultaneously assign current keyboard activity to other outputs - so that you can play along with something that was previously "recorded".

These same programs can allow independent recording and simultaneous playback of multiple "tracks". Like a multi-track recording studio only without the hassle of tape splicing, editing and (worst of all) over-dubbing noise.

The "middle" program can do tricks like making a chord played on the keyboard seem to be rising in pitch, constantly, without ever actually going beyond a pre-defined limit. It's not magic, it involves forming a "stack" of the notes and allowing the program to increase the pitch of the notes in the stack until they reach a pre-determined limit at which time the note is "faded out" and placed in the bottom of the stack.

The "middle" program can do lots of

different things. So many, that it's going to be a while (possibly a long, long while) before we know what they all are.

If you're looking for something that will reach a "finished" state beyond which there is nothing further to do, this isn't the product for you.

SO MANY "DIFFERENT" PROGRAMS

One thing that you may notice in the discussion above is that all of these very different "resource allocation" schemes have in common the fact that they all use LOOK and NOTEOUT. We could make these two routines a part of each of the larger programs if we wished - there wouldn't be any problems with that - except that they are long-ish and would take a while to "load" into the machine's memory. Particularly if you're not using the computer's optional cassette interface. I think there's a much better way.

We can write the LOOK and NOTEOUT programs so that they're what's known as "subroutines".

Now ordinarily, computer programs proceed sequentially through memory an instruction at a time. Like this:

INSTRUCTION → INST. → INST. → INST.

figure (g)

But a subroutine allows a block of programming to be stored out of sequence in the machine so that when you "call" or "jump to" a subroutine it's like this:

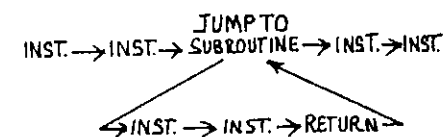


Figure (h)

The "return" causes the computer to go back to the place that it was before the subroutine was called and continue executing the main program.

Maybe the "subroutine" concept confuses you (though after such a terrific explanation it's hard to imagine how). If it does, here's another way that you can think of them:

SOFTWARE MODULES

You're certainly used to synthesizer "hardware" modules by now - all those little processing elements (VCO's, VCF's etc.) that we tie together with patch cords to produce different sounds or effects.

Here we have their equivalent in computer instructions - little modules of programming that are patched together (not with wire, of course, with more programming) which, depending on how they're tied together, produce different effects.

LOOK and NOTEOUT are not the

only software modules that are useful, others include SAVE (the "recording" module, SREPRO (the "playback" module), DELAY (a time delay routine), POLY (a useful polytonic resource allocation algorithm), and others.

These various modules are available in a number of different forms. They're available just as program listings (which can be manually entered into the computer - very tedious but about as cheap as you can get) or they're also available on cassette tape that can be loaded into the computer using the optional cassette interface.

First choice for a place to save these universally useful programs, though, is Read Only Memory.

This is the most expensive alternative (ROMs have to go for about \$20/each - one would be filled by the programs mentioned above) but it has the advantage of NOT HAVING TO LOAD THE PROGRAMS AT ALL. Every time you turn on the machine, they're there, waiting to be used.

SOUNDS INTERESTING WHAT DO I NEED TO GET STARTED?

If you already have some PAIA synthesis equipment, you're well on the way, but you need to convert to the new digital format. We've tried to make that as easy and inexpensive as possible by providing a retro-fit kit to digitally encode your present PAIA keyboard, the EK-3 Keyboard Encoder Kit mentioned in the POLYPHONY "Lab Notes" reprint included in this package.

This encoder is primarily designed to fit 4700 series keyboards, but will of course fit 2720 series equipment as well. It is one of our experimenter's kit series and does not include step-by-step instructions. In fact, the EK-3 re-print that is part of this package is the instruction set.

If you want to start over with a new keyboard, we have the 8782 Encoded Keyboard - one of our full kits with complete instructions.

If you already have an organ and would like to use that keyboard for either synthesizer or synthesizer/computer interface, we have the EK-4 Organ Keyboard Encoder as described in the accompanying package.

The advantage to this is that the keyboard already in the organ may be used for both synthesizer/computer and organ - all at the same time. Even if there are no "spare" contacts on the keyboard.

BUT I DON'T HAVE A SYNTHESIZER!

Looking back over the text to this point I notice an important point that has not been prominently mentioned. This

system - because of the properties of the D/A - will work only with low-cost LINEAR synthesizer modules. Synthesis modules whose characteristics are exponential cannot be used (though it is an easy matter to substitute another D/A for ours).

It is difficult to tell someone what the configuration of their synthesizer should be. Particularly with modular equipment like our current line. The modules that make up the system are so much a function of the use to which the system is to be put.

Never the less, we have two systems configured as starting points. "Starting points" because it has been our experience that most people add and make changes to their system as time goes on. Customizing it to their application.

These two packages are the 4700/C (primarily a monolithic system) and the 4700/J (suitable for polyphonic work, limited multi-track recording, etc.). These are both systems that we originally put together to take to shows. Each for its intended purpose, they have proven to be reliable and versatile; each capable (by design) of turning someone from an "I don't like synthesizers" person into a "I never realized they could do that" person. Maximum usefulness and versatility within minimum "waste" capacity.

The module complement of each of these systems is itemized in the product summary, but this would seem an appropriate place to discuss the "philosophical" (if you will, just this once, excuse so pretentious a term) implication of the systems.

The 4700/C is a minimal, useable system. It has roughly the capabilities of the "mini" this and that that you see advertised. It's made for people who find synthesis interesting but aren't really sure that they're going to get into it in a big way. It is (briefly) an ideal place to start. And since all of our gear is modular and available separately, it is a system which will easily grow as your interest grows.

The 4700/J is by the standards of the industry a "good-sized" system. It's difficult to make comparisons, since some of the modules (particularly those that are the computer interface) aren't available from other manufacturers; but, if these modules were available and you purchased them assembled through the normal distribution chain the 'J would be on the order of \$2,500 to \$3,500 worth of equipment. And, again, it's not a dead-end system, but one that can grow.

One final comment in this section is in order, and it may seem strange for someone who is, after all, trying to sell you equipment:

DON'T OVER-BUY

There are two reasons for making a statement like this - both imminently practical; 1) our experience has been that you will probably like the equipment a lot and will be a customer for many years, but if you don't (and aren't) you don't have a bunch of money sunk in something you're not going to use. We won't have someone wandering around bad-mouthing the gear.

2) Without committing to anything in print, development goes on all the time - to the practical synthesist, the versatility of modular equipment makes it desirable to have some of it around (ask anyone seriously involved in electronic music synthesis). But, well, look at any issue of POLYPHONY - development goes on and you never can tell what's just around the corner.

WHICH COMPUTER?

This one is almost as bad as which synthesizer. For the same reasons - the decisions are very personal and user related. Also like the "which synthesizer?" though, we have suggestions.

Our first, and strongest, suggestion is our own 8700 Computer/Controller. High on the list of compelling reasons to select this machine should be the fact that it will have our fullest software support (all of the programs mentioned earlier are available now), it is physically designed to fit into a space that has been kept free in our 4700 and 8700 series keyboards and is a machine designed to the PAIA ideal of "maximum impact for minimum bucks".

The 8700 is based on a 6503 processor (a fully software compatible version of the increasingly popular 6502) and has features as described in the product summary. This processor was chosen over others which were - at the time that the decision was made - more popular for a variety of reasons, but by far the biggest was that it is an easy machine to use. Even if you're programming in machine language (and don't kid yourself, the day will probably come that you will want to do something completely different - something not available either from us or from the independent user's group program exchange - and the only way to do it will be to write the code yourself, it's easier than it looks).

But let's suppose that you already have a computer. If that computer happens to be something like a KIM-1, you're in great shape. We will shortly have a complete KIM-1 package showing how to interface and almost as complete a selection of programs as for our own machine (we like the KIM series stuff - and since it, too, uses a 6502....)

If you have a SWTP 6800 system, the 8780 and 8782 instructions already outline using one of their MP-1's for

interfacing (sorry, no software support from us right now, but surely the user's group will come up with some - Southwest has a really nice, popular system).

Coincidentally, there are other machines that use the 6502 processor for which all of our software is written; if you haven't heard of them yet, you will.

They are:

Commodore's PET (personal electronic transactor) which looks at this point like it will sell in the \$600.00 range.

Certainly you're all familiar with Commodore - they're an old-line (if there is such a thing) calculator company.

and

Apple Computer Company's
APPLE II

We like the APPLE II machine a lot and probably a single glance at the enclosed literature will tell you why. It not only looks nice and can grow up to be a VERY LARGE system, but it has all the bells and whistles including FULL-COLOR VIDEO GRAPHICS capabilities (vectored, no less). I own one (one of the very first, I'm led to believe) and I can tell you - it's a very impressive system. You will be seeing a system available from PAIA (by October, we hope) based on this true "appliance" computer.

The PAIA/APPLE system is not yet fully configured, but target price is approximately \$2,500.

LAB NOTES

COMPUTER MUSIC, WITHOUT THE COMPUTER. or: What to do 'til your processor arrives.

By: John S. Simonton, Jr.

I realize that a lot of you will respond to the introduction of the 8780 Equally Tempered D/A with a frustrated, "But, I don't HAVE a computer."

Here's a little surprise. You don't really need a computer to do some very interesting and useful things with the 8780. You are going to need some additional hardware, as we'll see in a moment, but it's not only inexpensive, it's also equipment that you'll need for processor interfacing later on anyway. You're not building something that will be scrapped when your computer arrives, just getting a head start. Getting READY: so to speak.

Let's shift our mental gears for a minute, and instead of thinking of the 8780 as a computer peripheral, we'll consider it in terms of being a digital sample and hold.

Our analog S/H circuits are acceptable, but they will always drift because they store information by charging a capacitor. Even if we were able to miraculously devise a capacitor with no leakage, we still have to measure the charge on the capacitor; and whatever circuit we use to do that will itself eventually drain away all the charge (I think that a Mr. Heisenberg had something to say about this, but I'm not certain). With a digital S/H, we don't have that problem, because we're storing the information as a pattern of 1's and 0's.

To use our new digital S/H we need some way to provide it with the 1's and 0's it needs to decide what voltage to produce. We need some way to "encode" our AGO keyboards.

There are lots of ways to do this, including the simple expedient shown in figure 1.

This is frequently referred to as a "brute force" encoder. When a switch closes, any diode connected to the switch line forward biases, causing a 1 to appear on the data line connected to it. The diodes are there in the first place to prevent "sneak" current paths back through the matrix. This is an acceptable encoder as long as you assume that only one key is going to be down at a time. But, when two keys are pressed

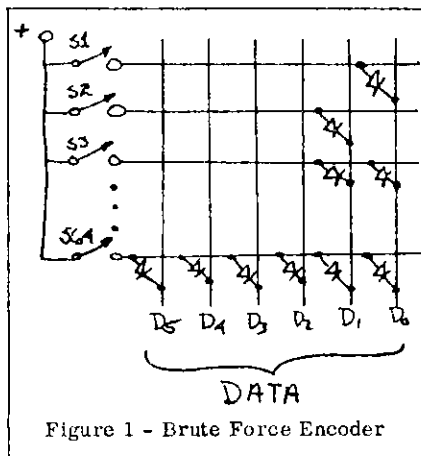


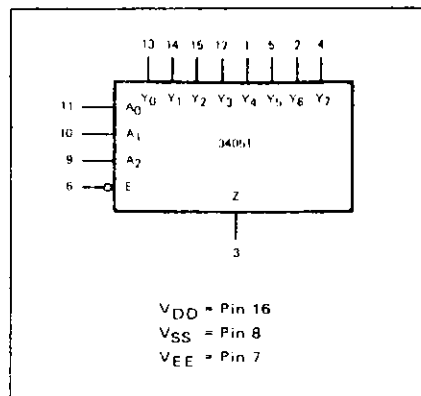
Figure 1 - Brute Force Encoder

simultaneously, the diodes act like OR gates and the data that comes out may or may not (most probably not) represent those keys. If, for example, we were to press the first two keys down at the same time, data lines D_0 and D_1 would both go high. Exactly the same situation that we had defined in figure 1 as being an indication that key 3 was down:

BUMMER

A more popular approach (because it works better) is to "scan" the keyboard a switch at a time to see if any are closed. There are LSI chips that do this with a single integrated circuit package; but, while saving design time is a great temptation, we're not going to use them. They're too expensive, and worse yet, not versatile enough to do all the things that I have in mind.

So that you can follow the design that I prefer, let me turn you on to a new part:



VDD = Pin 16
VSS = Pin 8
VEE = Pin 7

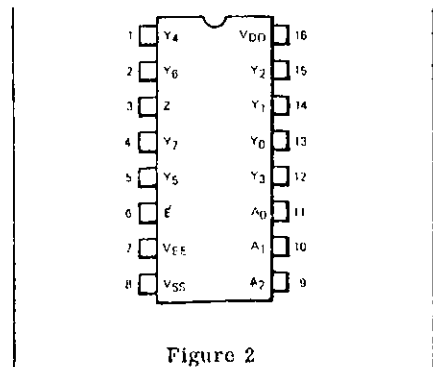


Figure 2

This is called a "4051 8 channel analog multiplexer/demultiplexer". Or, just 4051. Inside the package are 8 bilateral CMOS switches. While one side of each of these switches is tied to one of the pins $Y_0 - Y_7$, the other side of all the switches are commoned and connect to pin Z. In mechanical terms, it looks like this:

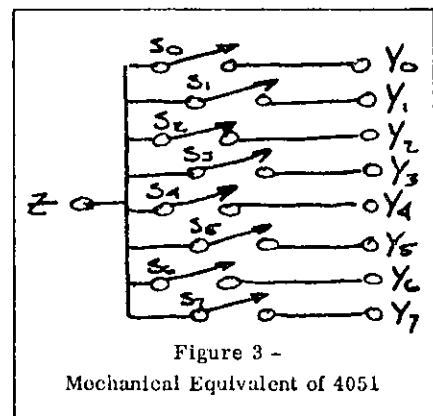


Figure 3 -
Mechanical Equivalent of 4051

One of the neater things about the 4051 is that each of those switches is individually "addressable" from the pins marked $A_0 - A_2$. If I put the binary number 000 into the address pins, switch S_0 will "close". 001 causes switch S_1 to be activated, and so on to 111 which addresses S_7 .

You will also notice a pin labeled E. This is an enable pin that sort of says "GO" to the rest of the circuitry in the package. As long as this pin is held at a high voltage, all of the switches will be "off", but when the E pin is grounded, the switch specified by the address currently on the A pins will close.

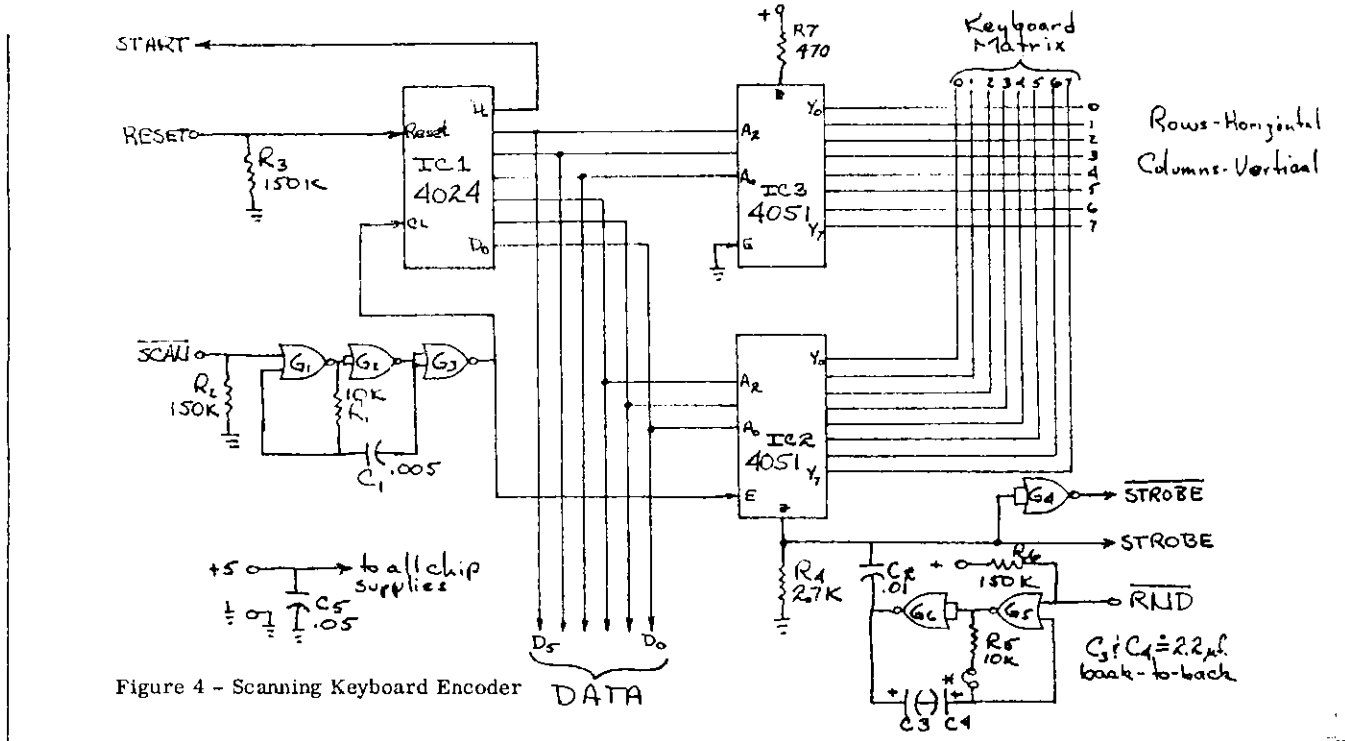


Figure 4 - Scanning Keyboard Encoder

What a terrific part. We really need to spend some time soon looking at all of the potential applications for this device. Not today, though. Today we have too many other things to do.

You're already familiar with the 4024 CMOS seven stage divider, we've used it before in other applications. Now we're going to use it again in a circuit that looks like figure 4.

This is our keyboard encoder. As far as parts go, there's not a lot to it. But it does a lot, watch.

Gates G1 and G2 along with R1 and C1 form an astable clock buffered by gate G3 that feeds the seven bit counter IC1. Notice that I can stop and start the clock by raising or lowering, respectively, the line labeled SCAN. If I'm not using this line, I can simply leave it disconnected and the pull-down resistor R2 will keep the clock running.

Notice that the three LSB's from the counter (D0 - D2) are connected to the address pins of IC2 while the next three MSB's connect to the address pins on IC3 (we are going to temporarily forget about the seventh bit). Assuming that the counter starts counting at 000000, both IC2 and IC3's Z pins are connected to their Y0 pins. If these two Y0 lines are isolated from one another nothing happens; but, if they are shorted together (by a switch at the point at which they cross in the matrix, for instance) then a current flows from the Z pin of IC3 to the Z pin of IC2 through R4 which is tied to the ground. The resulting voltage rise across R4 appears on the line labeled STROBE as a logical 1, which we can interpret as an indication that a key is down.

When the clock cycles and the counter advances to 000001, it has no

effect on IC3, but IC2's Z pin is now connected to it's Y1 pin. If those points in the matrix are isolated - nothing; if they're connected, we get a 1 on the strobe line. As you can see, each clock cycle advances the counter, which will have the effect of looking at each cross point in the matrix, one at a time. A STROBE results if the cross points are connected.

At any instant in time, the six bit number appearing on the data line is the number of the key being examined - in binary, and the status of the STROBE line will tell us whether that key is up or down.

It will also be handy at times to have a line that goes low when a down key is found, so G4 is used as an inverter to provide the complement of STROBE - STROBE. (I'm tempted to say son-of-strobe, but actually NOT strobe.)

One subtle point about the 4051's that we overlooked above: the line from the clock also connects to the E pin of IC2. The effect of this is to allow a STROBE to occur only during negative half-cycles of the clock (immediately after the counter changes state) like this:

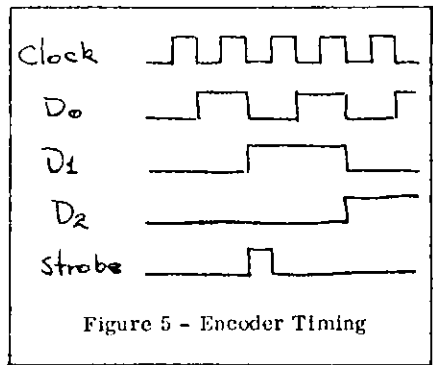


Figure 5 - Encoder Timing

which assumes that key 000010 is down. This is done for timing considerations.

Also, getting back to the counter again for a moment, we have a reset available; and while I can't think of a use for it right now, one may come up later. I bring it out on a line with a pull down resistor, R3, and label the line RESET. Raising this line to a 1 will reset the counter. Also, that seventh bit that we conveniently forgot, we can now bring out on a line labeled START. In computer application this line will serve as an indication that a scan is just starting or ending.

So, that's our all-purpose, super-gee-whiz keyboard encoder. In all of the drawings, I've shown it operating from a 5 volt supply because in computer applications we're going to be tapping power from the processor; but we're using CMOS logic here and the big reason is that it likes all different kinds of supply voltages - anywhere from 3 to 15 volts. If we retro-fit this stuff into a 4782 Road Keyboard (which as you might expect, I highly recommend) we can easily use the +9v. part of the supply that's already there to power both the encoder and the D/A.

The encoder can handle up to 64 switches (the number of cross-points in the matrix) and it will obviously work with a 5 octave keyboard. We really want to concentrate on a 37 note unit, though, since this is our standard.

No matter whose keyboards we are going to use, we are probably going to have to make some changes in the switch busses first. I'll show you on one of ours. If yours is different, I'm sure you can figure out something.

PAIA keyboards (and most others, too)

have two busses; one of which boils down to a single switch that is closed as long as any key is down. With analog S/H's, this is a signal to the circuitry to do its stuff. We don't need this anymore.

The second buss is really 37 switches, with one side of each switch tied to a common connection. We could represent it like this:

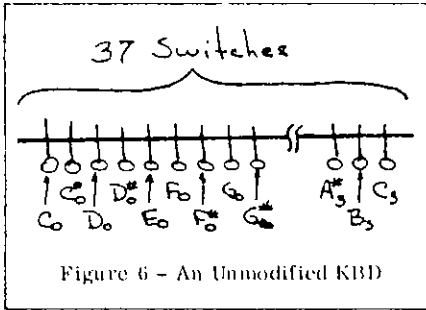


Figure 6 - An Unmodified KBD

The switch contacts that are not commoned would ordinarily go to the voltage divider board in an analog system.

We need to break these switches down into groups of 8 (giving us 4 such groups with a group of 5 keys left over) by cutting small sections (about 1/8 inch or so) out of the buss rod that runs the length of the keyboard. When you

do this, don't forget that you have the keyboard upside down. Be sure that the first cut is between the first G and G# on the keyboard. I ran into structural problems after cutting the buss rod; one section of it was supported at only a single point. An easy fix for this problem was to slip short sections of clear tubing (spagefil) over the adjacent ends of the cuts, providing both insulation of the buss section and mechanical rigidity. When you're finished, what you have could be represented by figure 7.

Now we buss together the individual key switches from each group by connecting together all of the first keys in each group, all the second keys in each group, etc. Notice that again to prevent sneak current paths which could generate "phantom" keys if multiple switches were closed, we've added a diode in series with each key. When we're done, we have what's shown in figure 8.

If we now redraw what we've got and superimpose it on the matrix, we have what's shown in figure 9.

You probably noticed that the first key does not begin at note 000000, but rather picks up from row 2 of the matrix; equivalent to making it key

number 010000 from the encoder's standpoint, and transposing the keyboard 16 semi-tones up-scale from the D/A's point of view.

IT DOESN'T MATTER WHERE THE FIRST KEY STARTS.

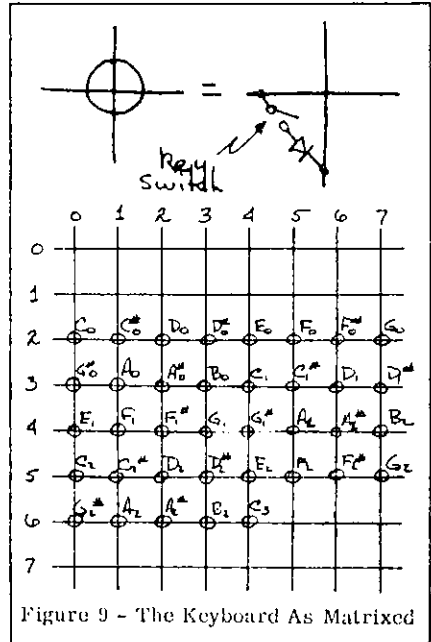


Figure 9 - The Keyboard As Matrixed

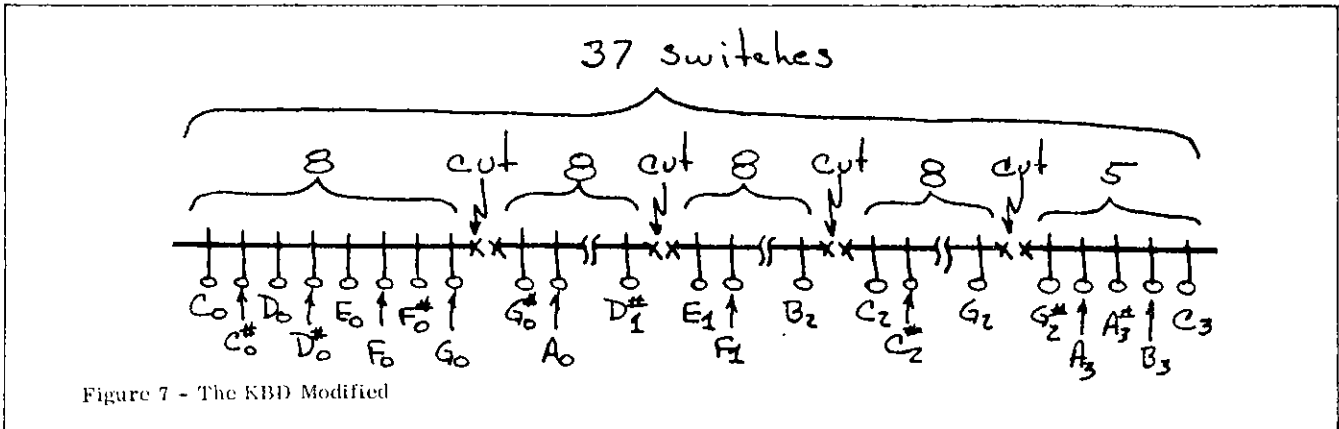


Figure 7 - The KBD Modified

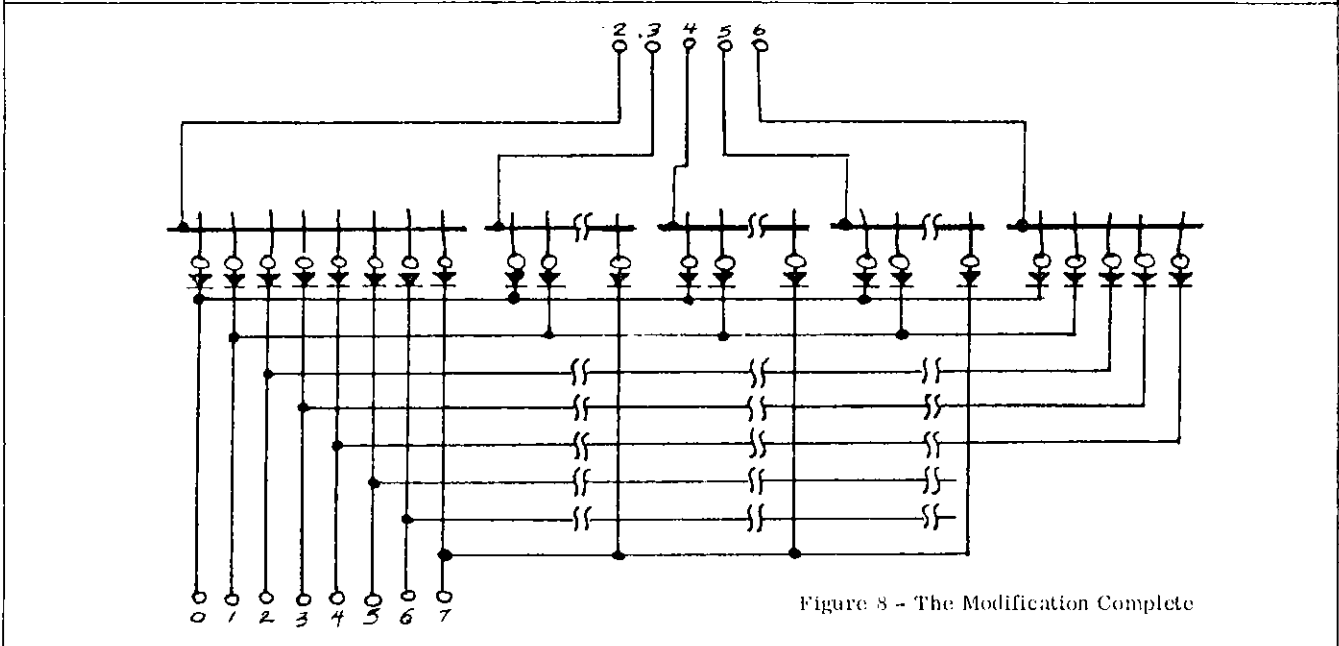


Figure 8 - The Modification Complete

Between the pitch knobs on our oscillators and the one on the D/A, we will be able to "put" the oscillator in any pitch range we want anyway.

There are a couple of good reasons for starting with key number 010000; First, I have a few computer things in mind for keys 000000 through 000111, and I want to hold them in reserve. Also, one of the things that our computer is eventually going to do for us is take care of transpositions into a new key signature, which will simply be a matter of adding to, or subtracting from, the note data the number of semi-tones by which we want to transpose. If my first key is 000000, I'm going to have a hard time transposing it down scale.

Now that I have the keyboard connected to the encoder, I'm ready to start doing things. Like replacing my old analog S/H with this shiny new digital model. There are lots of ways that I can do this. One is shown in figure 10.

Assuming that no keys are down, the encoder's STROBE line is at a 0 and STROBE is at a 1, making the RDY on the D/A high. The 8780's input latches are in a holding state and the activity on the data lines D₀-D₆ is invisible to the converter. This is fortunate, since the data lines are "counting" as the encoder continually looks at the keyboard.

Now, we push down a key. For the purpose of illustration let's say that it's the first key, number 010000. When the data lines next reach the state 010000, the encoder finds that the key is down, and because of that, the STROBE line goes high stopping the encoder clock, and the STROBE line goes low which takes the D/A's RDY line to a 0 putting the D/A's latches in a pass state. The new note data (010000) is strobed into the converter and a control voltage representing that key appears at the control voltage output of the D/A. The STROBE line from the encoder also connects to the D₆ input of the D/A, which appears at the D/A output panel as the first trigger flag (F1), so we have a trigger showing that a key is down. And this trigger is used the same way we would a trigger from the analog system.

As long as the key is down, the system is going to sit in this configuration. But, when I release the key, new things happen. Almost simultaneously STROBE goes low which removes the trigger flag D₆ (indicating that the key is now up) and allows the clock to start again (searching for the next key down). Simultaneously, STROBE goes high forcing the RDY line on the D/A high which puts the latched in a holding condition - and what they're

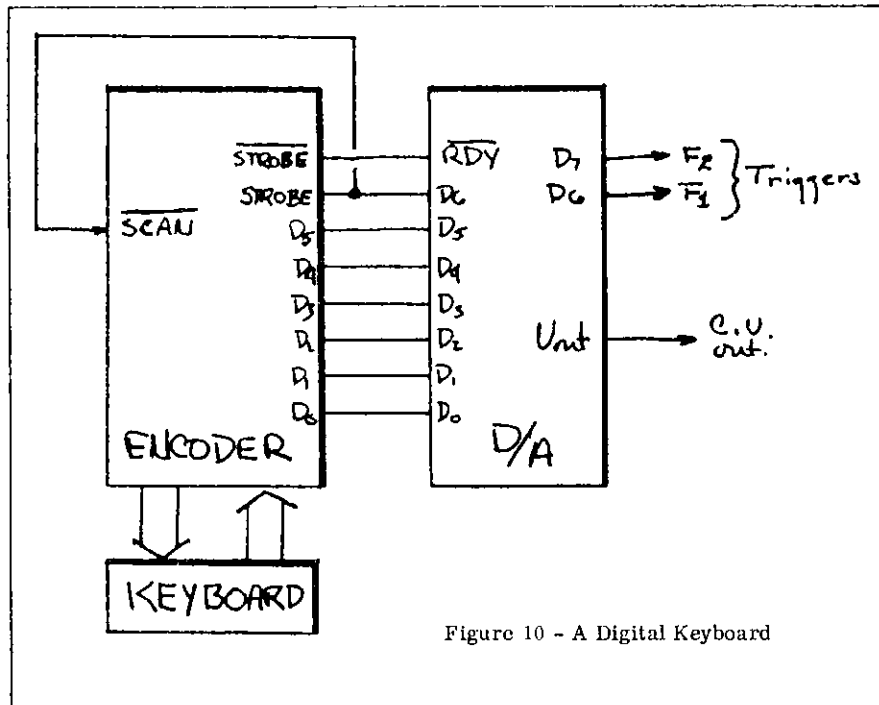


Figure 10 - A Digital Keyboard

holding is data on the last key that was down.

This is behaving exactly like the old analog system that we had, except, as I already mentioned, it doesn't drift. AND it gets rid of that annoying "in between" note that we had with the old keyboard if two notes were pressed at the same time (since the clock stops, the encoder can "see" only one down key at a time). AND, it doesn't have 37 adjustments to tune it; now there are none.

Let me show you something that this keyboard can do that our others can't.

Suppose that we remove the wire connection between the encoder's STROBE output and SCAN input. You will remember that this was the thing that caused the clock to stop when a key was found down. If we replace the wire with a capacitor, say about .22 mfd. or so, we have generated a little time delay in this loop. The clock will stop when a key is found down, but only temporarily - until the capacitor discharges - then it is going to go looking for the next key down. If, in the process of searching, the encoder finds another key down, it will strobe it into the latches, hold for the time delay, and then go searching again. With this arrangement, if two keys are held down, the output of the D/A will alternate between the two, and what we will hear is a trilling between these two notes. If three keys are held down, each note will be heard in turn and while this is not polytonic by any stretch of the imagin-

ation, it can certainly sound that way.

Can you imagine what the effect of pushing down a large number of keys will be? I call it the "orgasmatic glide" but everyone here thinks that's a terrible name.

Anyway, the arpeggiation gimmick is slick and if you wish it can be left in place and bypassed with a switch when not used as shown in figure 11.

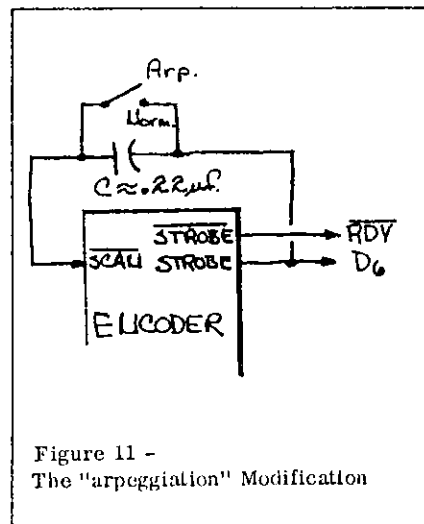


Figure 11 - The "arpeggiation" Modification

Here's another one.

You may have noticed that there is an input to the encoder that I hadn't mentioned; the one labeled (innocuously) RND. This line is normally held high by R5, but when pulled low momentarily it causes G5 and G6 to both change state which in turn activates the strobe line - even though there are no keys down.

The effect of this is that whenever we activate this line, whatever number happens to be on the data buss at that instant will be strobed into the D/A. Since the encoder clock is working very fast, there is no way to know in advance what the number on the data lines will be. As you've probably guessed, RND stands for RANDOM, since that is the effect of this input. It causes a random note to be strobed into the D/A.

If it occurs to you that there is a lot of activity around G5 and G6, what with R5, R6, C3 and C4 and the funny little jumper marked (*) being there, you're right. This circuit not only buffers the RND input line, it is also a slow clock. If we hold the RND down for more than just an instant, square waves begin appearing at the output of G6. And, naturally, for each cycle of the output of G6 a new random note is strobed into the D/A. With the values shown, the tempo of this clock is several cycles per second. That's a bunch, and that's where the (*) marked jumper in series with R5 comes in. By replacing this jumper with a pot (about 500K is a good value) we've picked up a control of the tempo of this circuit.

By adjusting this new tempo control we can effect not only the rate at which random notes are thrown out, but also the character of the notes (whether they appear to be really random, or run upscale, or downscale, or whatever).

To understand why the character of the notes is effected, imagine for a moment that the tempo of the RANDOM clock is exactly 1/64 that of the scanning clock. Under these conditions, the RANDOM clock is going to pull one note from the encoder for each complete encoder scan. Since 64 notes constitutes an entire scan, the RANDOM note that we pull will not be random at all. It will be the same note each time.

Suppose, now, that the RANDOM clock is running at exactly 1/65 the tempo of the scan clock. Now each time the RANDOM clock says "sample", the scanner will have gone through a complete cycle plus one note. Each succeeding sample will pull a note that is one semi-tone higher in pitch than the previous sample, and we will hear an ascending series of semi-tones that increments by one semi-tone for each event.

If the RANDOM clock is running at 1/63 the frequency of the scan clock we will have a similar situation except that the note pulled each time will be one semi-tone lower in pitch than the preceeding semi-tone.

Actually, for any practical situation, the RANDOM clock is going to be running several hundred or thousand times slower than the scanning clock; but the principle still applies. Small changes in the RANDOM clock rate will produce wide variations in the character and organization of the notes that are "randomly" pulled from the encoder.

Out of space and out of time, again. And so much left to do. It will have to wait for next time.

Speaking of next time - here are some things that we're going to do:

We're going to look at a memory add-on for the encoder, D/A combination that will allow you to do some terrific digital sequencer things. We're also going to look at an expansion system that will convert what we've done so far into a polytonic (phonic) keyboard. Also we'll have a story on a touch keyboard - the easy way, and will look at ways that this kind of thing can be tied into our encoder, D/A set-up.

And, I think, our computer will be ready. We've put a lot of time into configuring it for maximum usefulness either as a stand-alone

micro-processor trainer or for use with the music stuff. I believe that the time has been well spent. When you see all of the things that this system will do for you it's going to:

BLOW YOU AWAY

No kidding.

POSTSCRIPTS

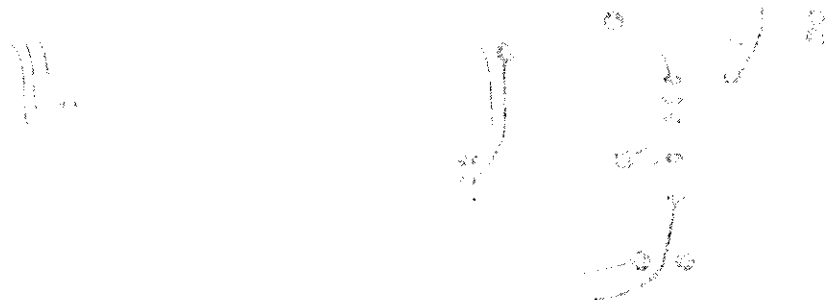
NOTE:

The PAIA Experimenter's Kit series is not intended for the novice builder. They are intended to provide the experimenter with a place to start on what will hopefully be a series of interesting and enlightening projects at the very lowest possible cost.

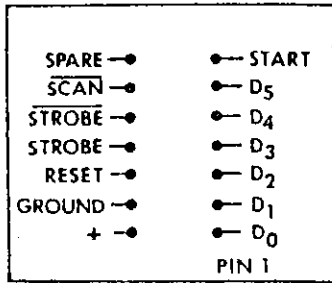
Because of this, parts that are considered to be either optional or easily obtainable from other sources (your "junk box" for instance) are not included. Also, circuit boards in this series are not normally printed with parts placement designations and assembly instructions are minimal, with most of the narrative type textual material concentrating on "how it works" and "how to make it do other things." If you feel that this approach does not serve your purposes you should return this item immediately for a refund.

Parts placement for the EK-3 circuit board is shown below.

Note that with the exception of the RND input, all input and output lines come together at the 14 pin DIP configuration between IC4 and IC5 on the circuit board. A DIP socket and connector may be used here if desired for easy connection and disconnection (a nice touch, but not highly recommended).



Below is an enlarged version of this I/O cluster which may be cut out and placed in the vicinity of the EK-3 for easy reference.



The "standard" connectors that we will be using for the complete kit version of these devices will be 25 pin "DB25" type sockets and plugs. If you decide to use these connectors, we

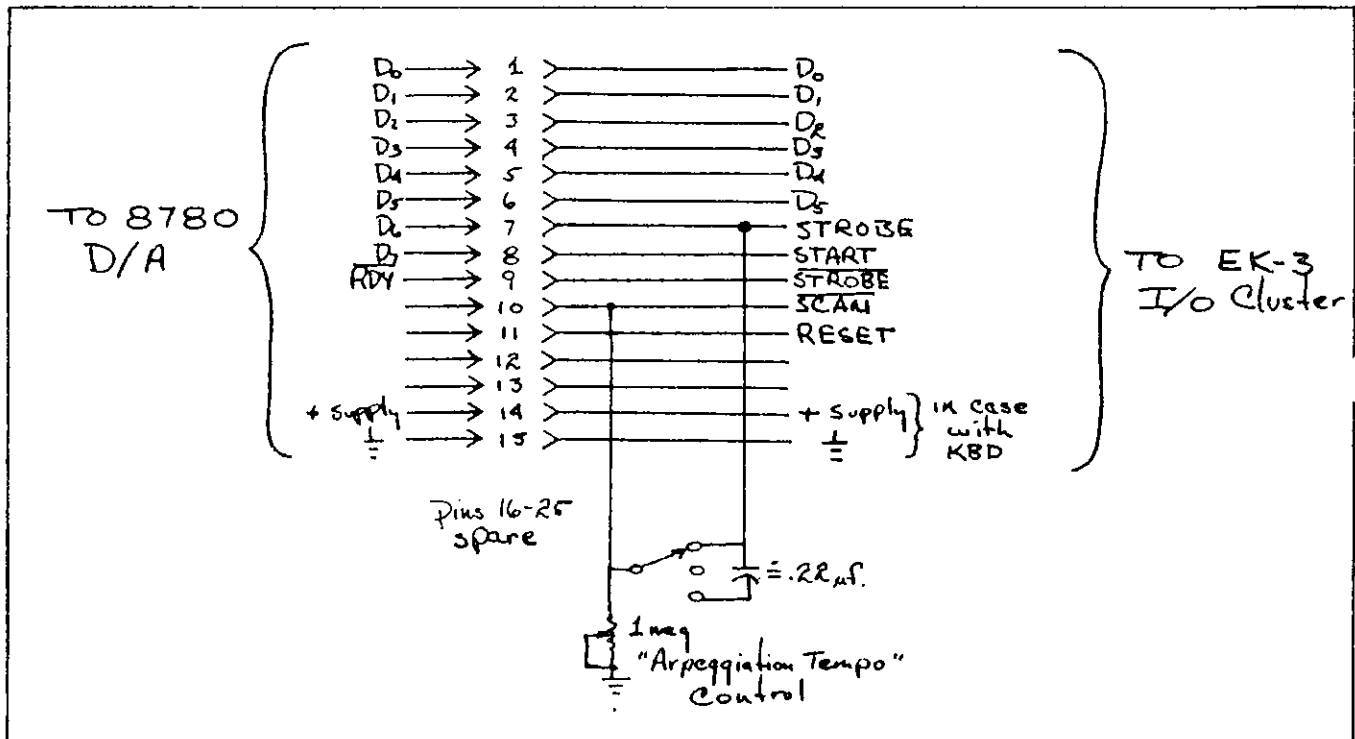
recommend that they be wired as listed below:

| PIN # | 8780 D/A (plug) | EK-3 encoder (socket) |
|-------|-------------------|-----------------------|
| 1 | D0 | D0 |
| 2 | D1 | D1 |
| 3 | D2 | D2 |
| 4 | D3 | D3 |
| 5 | D4 | D4 |
| 6 | D5 | D5 |
| 7 | D6 | STROBE |
| 8 | D7 | START |
| 9 | RDY | STROBE |
| 10 | N/C | SCAN |
| 11 | N/C | RESET |
| 12 | N/C | N/C |
| 13 | N/C | N/C |
| 14 | + supply | +supply |
| 15 | ($\frac{+}{-}$) | ($\frac{+}{-}$) |

Some forethought has gone into the configuration of these connectors with additional scheduled elements of the series in mind.

For example, if this scheme is followed, the arpeggiation gimmick described in the text would be added as shown in the figure below.

The three position switch can then select a mode of operation in which the clock stops when a down key is found (up position); a mode in which the clock does not stop when a down key is found, and this mode will be used in polytonic retro-fits (middle position); and the arpeggiation mode in which the clock stops momentarily for down keys (bottom position). As is indicated, a control of the arpeggiation rate (within limits) may be added with the 1 meg potentiometer shown.





Equally Tempered Digital to Analog Converter

By: John S. Simonton, Jr.

Many experts will tell you that in order to interface a computer to an electronic music synthesizer, you must use exponential response voltage controlled elements (oscillators, filters, amplifiers, etc.).

Here's why:

Computer control of synthesizers requires a Digital to Analog converter to change the numbers that the computer puts out into an analog control voltage that the modules can use.

By far the most common type of D/A (so common that many seem to think it's the only kind) is known as an "R/2R ladder". I don't want to get into the design details of this circuit. If you are interested, there is plenty of information available from text-books, manufacturers literature, etc. But we do need to examine a functional aspect of this circuit.

Any analog to digital converter works by accepting at its input a digital quantity (we will call this data) and generating at its output an analog

voltage that is a unique representation of that data. Most of the D/A's that I'm familiar with accept the data as binary digits - a bunch of 1's and 0's that appear simultaneously on a group of wires going into the converter.

In a R/2R ladder converter, a unique weighting is assigned to each bit in the data coming in. When the time comes for a conversion to be made, the circuitry adds together the weightings corresponding to the bits in the data that are in an "on" state (for our purposes, a 1; through not always) and ignores the weighting represented by the bits that are "off"-equivalent to adding in a zero.

If we assume that we are going to be using exponential response oscillators, the R/2R ladder converter works quite well. We can assign weightings to the bits that are integral multiples of 1/12 volt; the same incremental voltage change that keyboards designed to operate exponential oscillators produce, and when we do we come up with a series of weightings

which - progressing from the Least Significant Bit (LSB) to the Most Significant Bit (MSB) - Looks like this:

LSB
1/12, 2/12, 4/12, 8/12, 2ⁿ/12 MSB

Figure 1

Where n is, of course, the number of bits that the converter can accept as data.

Let's watch four bits "count" into this type of converter and observe the resulting output voltages.

| DATA | MEANS | OUTPUT |
|---------|---|------------------------------|
| 0 0 0 0 | 0+0+0+0 | 0 |
| 0 0 0 1 | 0+0+0+ ¹ / ₁₂ | ¹ / ₁₂ |
| 0 0 1 0 | 0+0+ ² / ₁₂ +0 | ² / ₁₂ |
| 0 0 1 1 | 0+0+ ² / ₁₂ + ¹ / ₁₂ | ³ / ₁₂ |
| ----- | | |
| ----- | | |
| 1 1 1 1 | ⁸ / ₁₂ + ⁴ / ₁₂ + ² / ₁₂ + ¹ / ₁₂ = ¹⁵ / ₁₂ | |

Table 1

If I had made the "word" (collection of 1's and 0's going into the converter) 6 or 8 bits long instead of just 4, the resulting series of output voltages would still increase 1/12 volt for every unit increment of the data and the only effect would be to increase the range of the output voltage.

Unfortunately, while the distinguishing feature of an exponential oscillator is that equal incremental voltage changes will cause it to generate a series of equally tempered pitches, this is not the case for linear response oscillators. A linear oscillator requires constantly increasing voltage increments to produce equally tempered semi-tones.

While this increasing voltage requirement doesn't make the application of R/2R converters to linear oscillators impossible, it certainly makes it cumbersome.

Cumbersome because we have to make the incremental change from the converter small enough to guarantee that there will be some pattern of 1's and 0's that defines a control voltage reasonably close to what we're really after.

Very small voltage increments - there are three things "wrong" with this:

1) We're going to need a "bigger" converter - one with greater resolution and consequently greater word size. Whereas 6 bits of data will provide a little more than 5 octaves of control voltage to an exponential oscillator; the same 5 octaves from a linear oscillator will require 12 data bits. Now, if that doesn't offend you by its notable lack of elegance, it's cost certainly should. A 12 bit D/A is going to set you back about \$100.00; then you've got to put it on a pc board, add controls - front panel, etc.

2) As if to add insult to injury, there will be lots of combinations of bits that represent the intervals between adjacent semi-tones, but notice that they are not equally tempered intervals and therefore next to useless even for micro-tonal tunings. We're paying out our hard earned bucks for words that we're never going to use, but must have to fill up the "cracks".

3) We've turned the determination of what data to output from a relatively simple matter of counting the keys and using that as the data into a process that at best is going to require a lookup table (where the machine says "Aha - key number 12, that's note 000110010100001") or some such similar computer calisthenics. Not particularly complicated, perhaps, but why bother with it if we don't have to.

And that, friends, is the point of all this. We don't have to. For the simple reason that an R/2R ladder

converter is not the only kind that we have to work with. There are other kinds. One of the other kinds is called a Multiplying D/A (or just MD/A, I guess).

While the most important operational feature of the R/2R ladder converters was that it added things to arrive at the output, the dominant feature of an MD/A is that (you're ahead of me, right?)

IT MULTIPLIES.

Far out.

If you're up on your basic music theory, a responsive chord (if you'll pardon the expression) should be struck here. The determination of the frequency of the pitches in equally tempered tunings is itself a multiplication process. The frequency of each semi-tone in the series is greater than the frequency of the preceding semi-tone by a factor of $2^{1/12}$ - the infamous "twelfth root of two" ($2^{1/12} = 1.059$). Intuitively, it would seem that this type of D/A would be more appropriate for our purposes.

In fact, this is true. We assign weightings to the bits (starting with the LSB) according to this series:

| | |
|-------------|------------|
| LSB | MSB |
| $2^{-1/12}$ | $2^{n/12}$ |

Figure 2

Where again, n is the number of bits of data that the converter will accept.

Now, we count into this converter the same way that we did in the R/2R ladder type. Remember that bits that are "off" here are not included in the total (only now this is equivalent to multiplying by 1) and that the product that results from the condition of the data will be multiplied by some internal reference voltage.

| DATA | OUTPUT |
|---------|---|
| 0 0 0 0 | $1 \cdot 1 \cdot 1 \cdot 1 \cdot V_{ref} = V_{ref}$ |
| 0 0 0 1 | $1 \cdot 1 \cdot 1 \cdot 2^{1/12} \cdot V_{ref} = 2^{1/12} V_{ref}$ |
| 0 0 1 0 | $1 \cdot 1 \cdot 2^{2/12} \cdot 1 \cdot V_{ref} = 2^{2/12} V_{ref}$ |
| 0 0 1 1 | $1 \cdot 1 \cdot 2^{2/12} \cdot 2^{1/12} \cdot V_{ref} = 2^{3/12} V_{ref}$ |
| ----- | |
| 1 1 1 1 | $2^{8/12} \cdot 2^{4/12} \cdot 2^{2/12} \cdot 2^{1/12} \cdot V_{ref} = 2^{15/12} V_{ref}$ |

Table 2 *

* Multiplying a base number raised to various powers (exponents) is accomplished by adding the exponents. That's how a slide rule works - remember slide rules?

You may recognize this as an equally tempered series (if not you'll just have to take my word; it is). All we have to do now is design a circuit that does this.

Let's do that.

Here's a simple unity gain buffer amplifier:

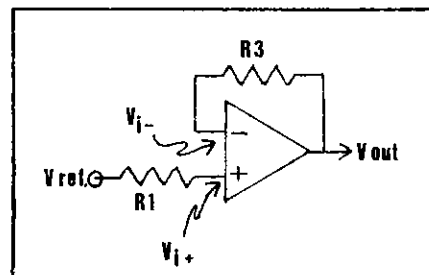


Figure 3

You may not be used to seeing it in this form because ordinarily the resistances that are shown would be replaced with direct connections. But having the resistances there doesn't matter simply because for any practical case, they are going to be much smaller than the equivalent resistance from either of the operational amplifier's inputs to ground. I should mention here that for any linear operational amplifier circuit the voltages at the inverting and non-inverting inputs are equal ($V_{i+} = V_{i-}$; this is the key to op-amp design, but that's another story). Of the circuit in figure 3 we can say:

(a) $V_{out} = V_{ref}$.

An excellent beginning. Here's another circuit:

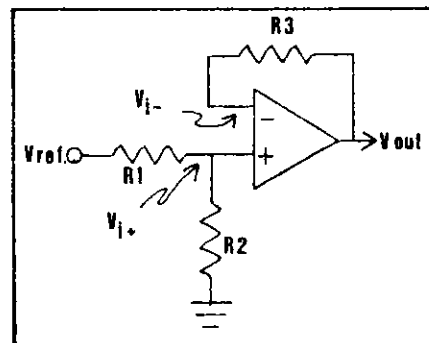


Figure 4

Adding R2 to the circuit has produced a voltage divider at the + input of the op-amp and because of this we can say:

(b) $V_{out} = \left(\frac{R2}{R2+R1} \right) V_{ref}$.

Fantastic. Now we change the circuit again so that it looks like this:

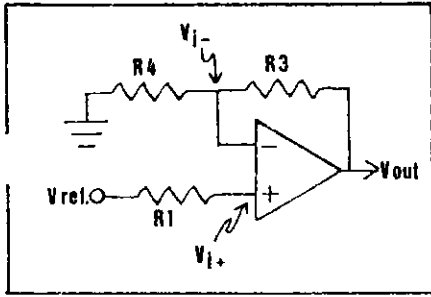


Figure 5

and instead of a voltage divider at the + input we now have one at the - input. This means that:

$$(c) \quad V_{out} = \left(\frac{R3+R4}{R4} \right) V_{ref}.$$

All together now:

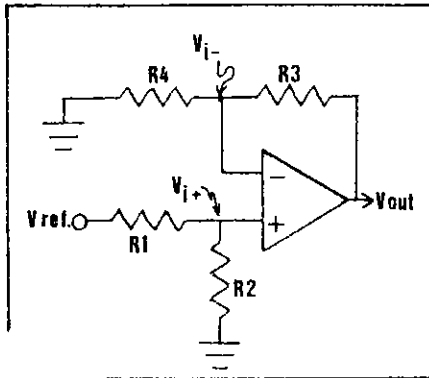


Figure 6

And for this configuration:

$$(d) \quad V_{out} = \left(\frac{R2}{R1+R2} \right) \left(\frac{R3+R4}{R4} \right) V_{ref}.$$

Do the four equations from (a) - (d) look familiar? No? Look back at Table 2. Now do they look familiar? Still no? Then let's say:

$$(e) \quad \frac{R2}{R1+R2} = 2^{1/12}; \quad \frac{R3+R4}{R4} = 2^{2/12}$$

and then by making these substitutions and putting the equations together:

$$(a) \quad V_{out} = V_{ref}. \quad \textcircled{A}$$

$$(b) \quad V_{out} = 2^{1/12} \cdot V_{ref}.$$

$$(c) \quad V_{out} = 2^{2/12} \cdot V_{ref}.$$

$$V_{out} = 2^{1/12} \cdot 2^{2/12} \cdot V_{ref} = 2^{3/12} V_{ref}.$$

Now you must certainly recognize them - it's the same series as the first four entries in Table 2. Putting the resistors

R2 and R4 into the circuit and removing them is simply a matter of putting switches (either mechanical or electronic) in series with them and when we do the whole circuit looks like this:

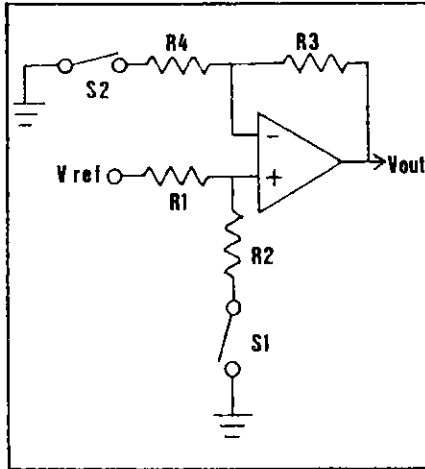


Figure 7

The switches S1 and S2 here are, respectively, the Least Significant and Most Significant data inputs to the converter; and I will avoid the obvious comment about this being a 2 bit D/A.

Oh, but there's one thing that I forgot to tell you:

$$(f) \quad 2^{1/12} \neq \frac{R2}{R1+R2}$$

Why? Because $2^{1/12}$ is a number greater than 1 and the only way that the ratio of a number to itself plus something else can be greater than 1 is if the something else is negative - which in our case, it's not (yes, there is such a thing as negative resistance, but the concept is not applicable here).

Happily, we have an alternative to negative resistance and that is to make:

$$(g) \quad \frac{R2}{R1+R2} = 2^{-1/12}$$

Making the exponent negative is equivalent to taking the reciprocal of the number.

At this point I'm afraid that in the interest of brevity I must make a gigantic leap and say that -- because we're using the reciprocal of the weighting, we must also complement the bit representing that weighting. In the instruction manual for this module, we will cover why. But there's not enough space to do it here. And, in any case, any of you who really want to can figure it out for yourselves. It's easy, honest.

Expanding this D/A out to handle greater word lengths is simply a matter

of cascading several of these sections. When we do this and replace the mechanical switches that we had earlier with 4066 type Quad Bi-lateral CMOS switches we come up with a thing that looks like the circuit shown in figure 8.

Notice that the complemented bits that we require are indicated by the overbar on like $\overline{D_0}$ for instance. This is read "not D₀" and by custom indicates that the low (0) state is considered to be "on".

You are probably also wondering about those R_a's, R_b's, etc. The values of these resistors are determined by solving equations (a) through (d) and they produce some strange values that need to be exact. 5%'ers won't get it here. In order to meet the necessary precision and stability requirements, we've had "one of the nation's leading resistor manufacturers" (at least that's what they say) make up some custom Cermet resistor networks. They look about like any 16 pin DIP IC (except that they're a beautiful robin's egg blue), but inside are resistors instead of other stuff. Once manufactured, they're trimmed by LASER to be exactly the right ratios (Laser, yet - how about that!).

I really don't expect that to impress you too much, but this should:

THERE ARE NO ADJUSTMENTS TO THIS MODULE

You just put it together and it "plays" (which is the computer people's phrase for works).

Do you realize that this gets rid of all those trimmers from our old '8 keyboard - it even gets rid of the zero pot. I really like it.

But we're really not through yet, we need to completely dress the design by adding input latches (so that the input information can be stored), and some kind of indicators so that we will know what's going on (LED's - they wink, they blink, they twinkle like stars in the night; anybody can look at this thing and know that it's got something to do with computers). This part of the circuit is shown in figure 9.

The 4042's are the latches and one of their features is that they have both Q and \overline{Q} (the complement of Q) outputs - since we needed some complemented data bits, this is nice. Q9 - Q14 are level converters. We need to have the "on" resistance of the 4066 switches in the converter circuitry working at as high a supply voltage as possible in order to achieve predictable low "on" resistances and this means that they operate from the +9v. synthesizer supply rather than the +5v. logic supply.

That's the design. Let's take a few minutes to review what we've got here.

We've got a new synthesizer module that does at least one thing that many people thought couldn't be done; a 6 bit Digital to Analog converter that will provide slightly more than 5 octaves of equally tempered control voltage to linear response voltage controlled synthesizer elements.

The front panel PITCH control allows the module's output to be chromatically transposed over another octave, so the total range of output voltage available is a little more than 6 octaves (compared to typically 4 octaves for a #4782 keyboard).

We have two trigger flags available, either of which can be set or re-set independently (very handy). As we will

see in a future issue, these flags can also be used to select micro-tonal intervals.

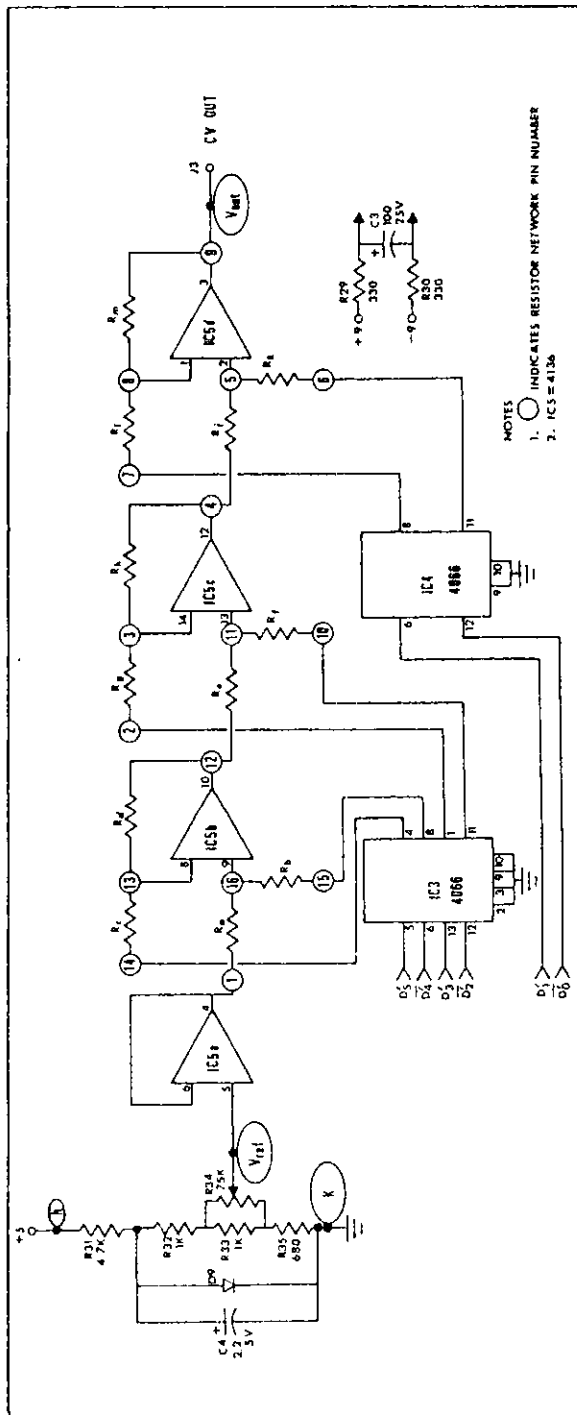
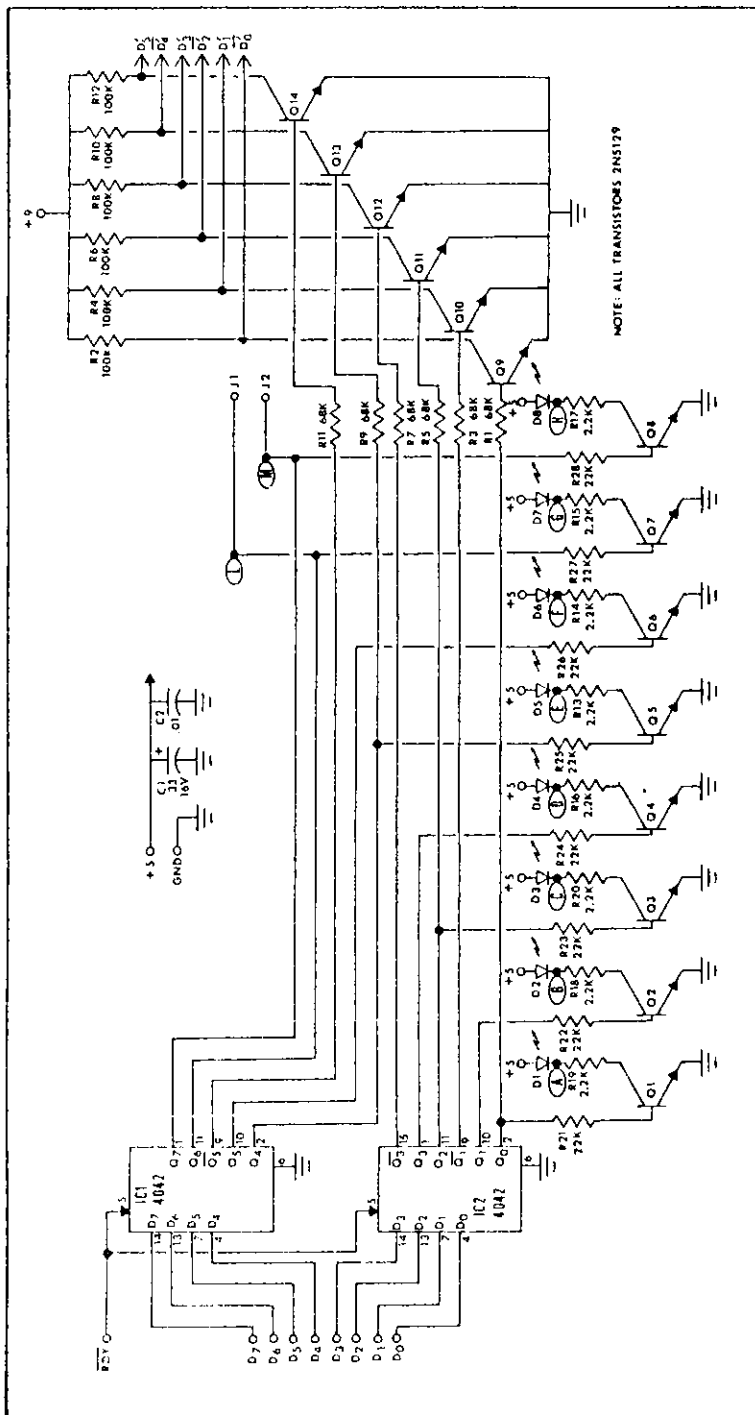
The status of the 8 bits of data coming into the module is displayed on the front panel LED's, six of which indicate the note that the module is producing and two of which indicate the status of the trigger flags.

To make the module easy to interface to anyone's computer (or simply keyboard encoders - see LAB NOTES) we have an input terminal marked RDY (not ready). When this terminal is grounded, the latches that are provided on the data lines are in a "pass" condition and any changes of the data on the data input lines will be reflected as

changes in the module's output voltage. When the RDY line is taken to a high logic state, the last data that appeared on the input lines is stored in the latches and further changes on the data bus will not produce any change in the output voltage (this is about like the action of the SAMPLE inputs of clockable sample and holds).

The road to applying the processing and control power of the computer to electronic music synthesis is not a short one - but it is certainly a trip worth taking. The Equally Tempered D/A is only a first step.

As first steps go, though, this is a good one - like walking in seven league boots.



LAB NOTES

IN PURSUIT OF THE WILD QuASH

by John S. Simonton, Jr.

Now that we have a way to interface our synthesizers to computers - the 8780 D/A - we can begin thinking of ways to independently control large numbers of musical elements simultaneously. Lots of VCOs, lots of VCFs.

The first time that you think of this your reaction may be something like:

WOW! - ALL THOSE D/As.

Multiple D/As (one for each control "channel") would be a possible way to go. An expensive way - at \$35.00 each, controlling just 4 VCOs means almost \$150 worth of just D/As.

There's a much cheaper way.

You may find this a little circuitous, after working so hard at our digital interface, but we're going back to analog Sample and Hold circuits.

Now wait, don't panic. These S/H's are nothing like the ones that we're accustomed to. They don't have to hold voltage steady for a long period of time - only a few milli-seconds. Long before even that short time has passed we will have used the computer to come back and re-write the correct voltage into the circuit. Computer re-freshed S/Hs.

Magic!

When you're designing a S/H to be good for only a fractional part of a second it gets really easy. Like this:

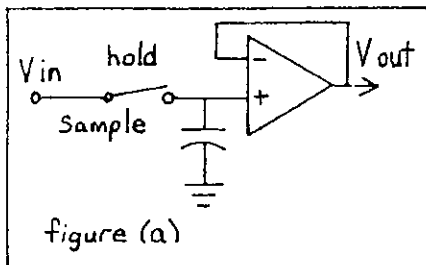


figure (a)

I'm sure that we've all seen this kind of thing before. It's an op-amp used as a unity gain voltage follower.

When it comes time to take a sample, the switch closes causing the capacitor to charge up to the input voltage. The output of the voltage follower "follows" this voltage (what else?), and when the switch opens again, the capacitor "remembers" the voltage.

One of the characteristics of this circuit is that the "+" input represents

a very high input impedance to any load that it sees. A relatively small capacitor can accurately hold a voltage for almost a second.

Now, we're not going to use a mechanical switch here. Last time, we looked at the 4051 multiplexer and decided that we would be using it a lot. And we are, just not this time.

This time, we're going to use a very close relative of the 4051 - the 4052 (I defy you to get any closer than that). The 4052 looks like this:

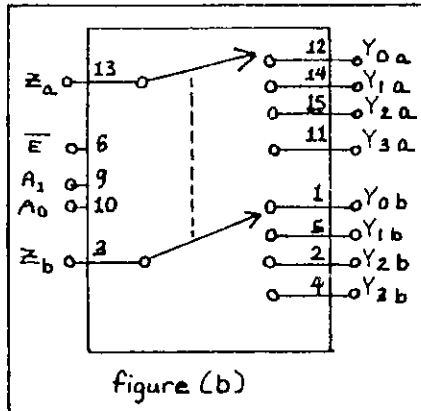


figure (b)

and whereas the 4051 was an electronic equivalent of a Single Pole Eight Throw switch, the 4052 is like a Double Pole

Four Throw one.

Which pairs of switches are to close is specified by the two address lines (A_0 & A_1). The switches actually close when the \bar{E} pin goes to ground.

Using 1/2 of one of these devices we can come up with a Quad Addressable Sample and Hold (QuASH?) that looks like figure C, and it works about the way that it looks. An address applied to the A_0 and A_1 pins sets up one of the four switches and when the \bar{E} pin is taken to ground that switch closes connecting the output of the D/A to the selected S/H. Simple.

That takes care of our control voltage output - but there are still other things to think about. For instance, we need a trigger flag (gate signal) to go along with each of the control voltages to take care of things like triggering envelope generators. *(1)

An easy way to handle this is to use the other 1/2 of the 4052 to route one of the two trigger flags available from the D/A to an output corresponding to the control voltage output. And since we're time sharing the D/A we also need some way to hold the status of that flag during the times that other control

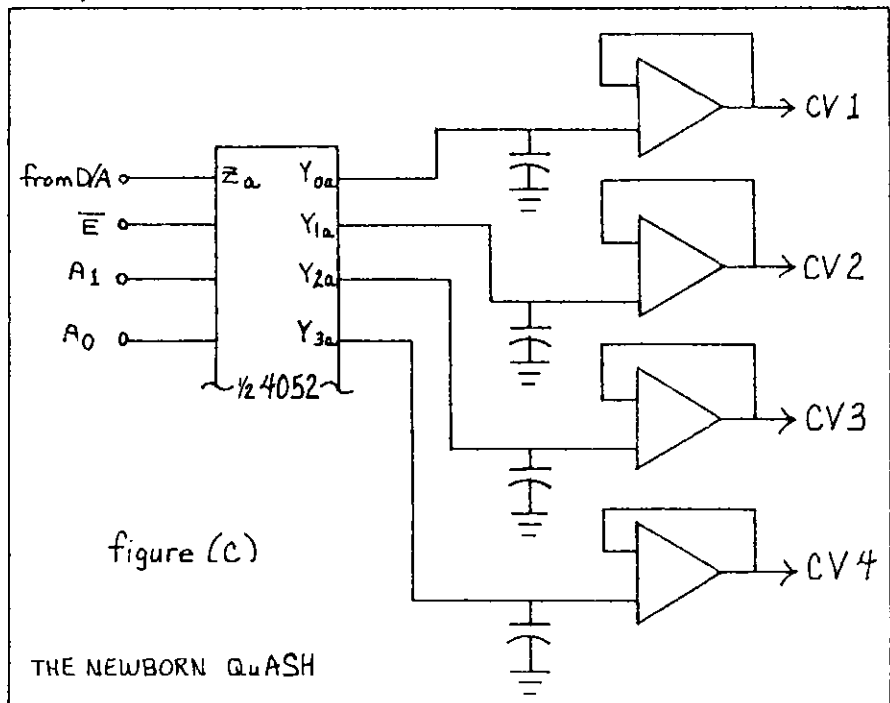


figure (c)

THE NEWBORN QuASH

data and then wait (or do something else) while these analog circuits get to where they're supposed to be. *(2)

Notice that the Q_0 and Q_1 outputs of the latch - corresponding to the two most significant address bits - go directly to the 4052's where they serve to select one of the four outputs.

Notice also that Q_2 and its complement $\overline{Q_2}$ as well as Q_3 and $\overline{Q_3}$ from the 4042 come out to pads on the circuit board. By jumpering these outputs to the inputs of the NOR gate G1 we can determine which group of addresses the card we're working with represents.

For example, if we connect the inputs of G1 to $\overline{Q_2}$ and $\overline{Q_3}$ then this block of four S/H's occupies the addresses 00XX in binary where XX represents the bits that select one of the four S/H. Address 0000 corresponds to the first S/H, 0001 to the second, and so on. By

connecting the inputs of G1 to Q_2 and Q_3 , the S/H's occupy the address 01XX. The first S/H is 0100, the second 0101, and like that. This scheme allows us to easily use up to four of these expanders (16 outputs) in a system without needing to do anything but set the jumpers properly.

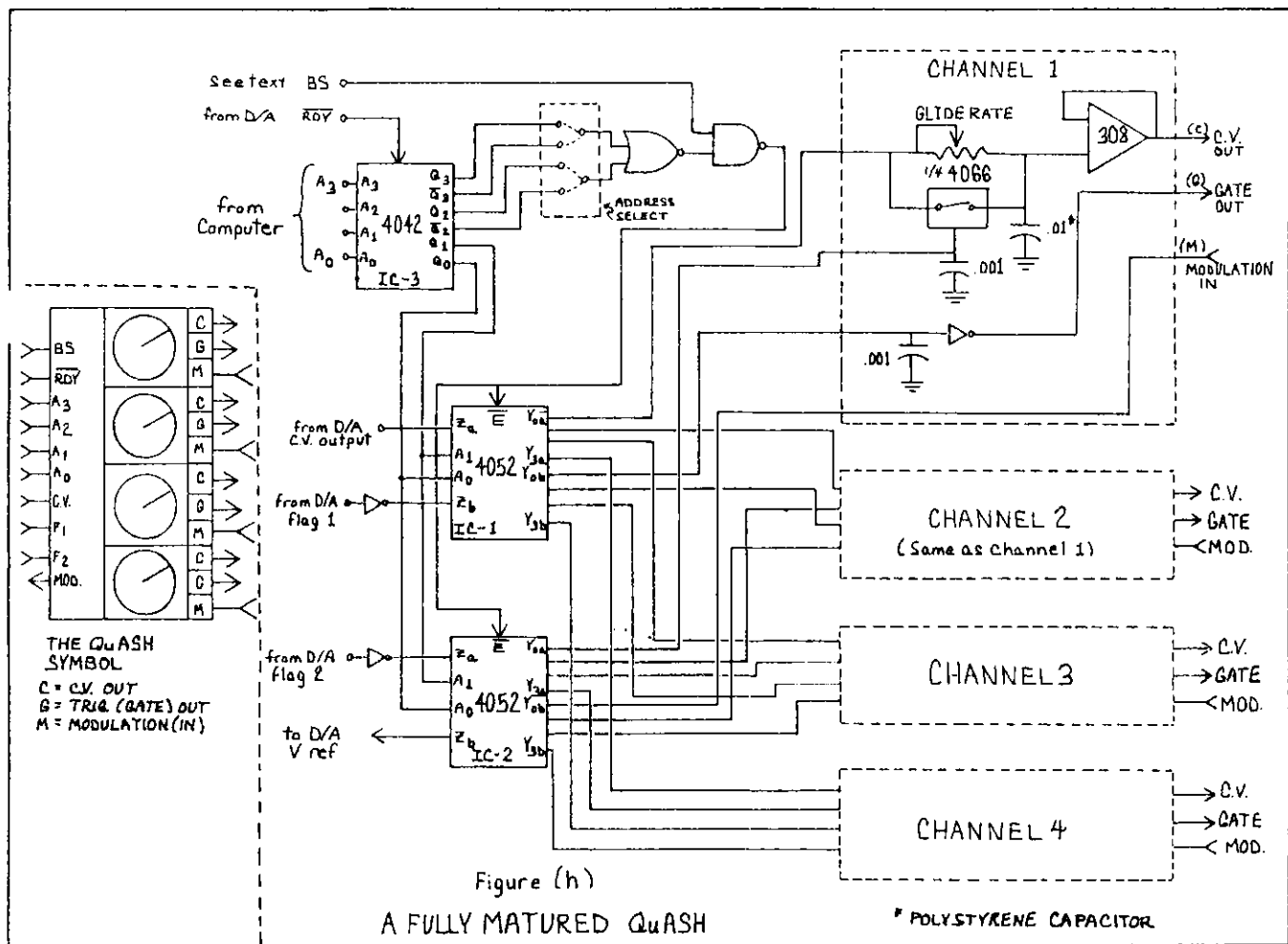
You will notice that there is another line coming out of this decoding circuit which is labeled "BS". This is not my opinion of this whole mess, it's a means by which we may expand the system beyond even four expander modules - BS stands for "Bank Select" and as long as this line is held at a logical "1" level the system operates as described to this point.

But, when the BS line is pulled low one input of the NAND gate G2 is not fulfilled resulting in its output being high which in turn holds the 4052's enabling

input (\overline{E}) high - which means that none of the switches in the multiplexer will close (even if addressed otherwise) and none of the S/H's will be selected.

External decoding circuitry is required to drive the BS input, naturally, but we would begin to need external circuitry at about this point anyway to buffer address lines. The decoding required here will be covered in the instruction manual for this kit.

When we tie all of these bits and pieces together, we come up with a thing that looks like figure H, our complete QuASH. And in the interest of saving space and time, we will from this point forward represent it with the symbol shown (at least until we can come up with something more abbreviated). The knobs in the output "boxes", by the way, represent the glide rate controls associated with each output channel.



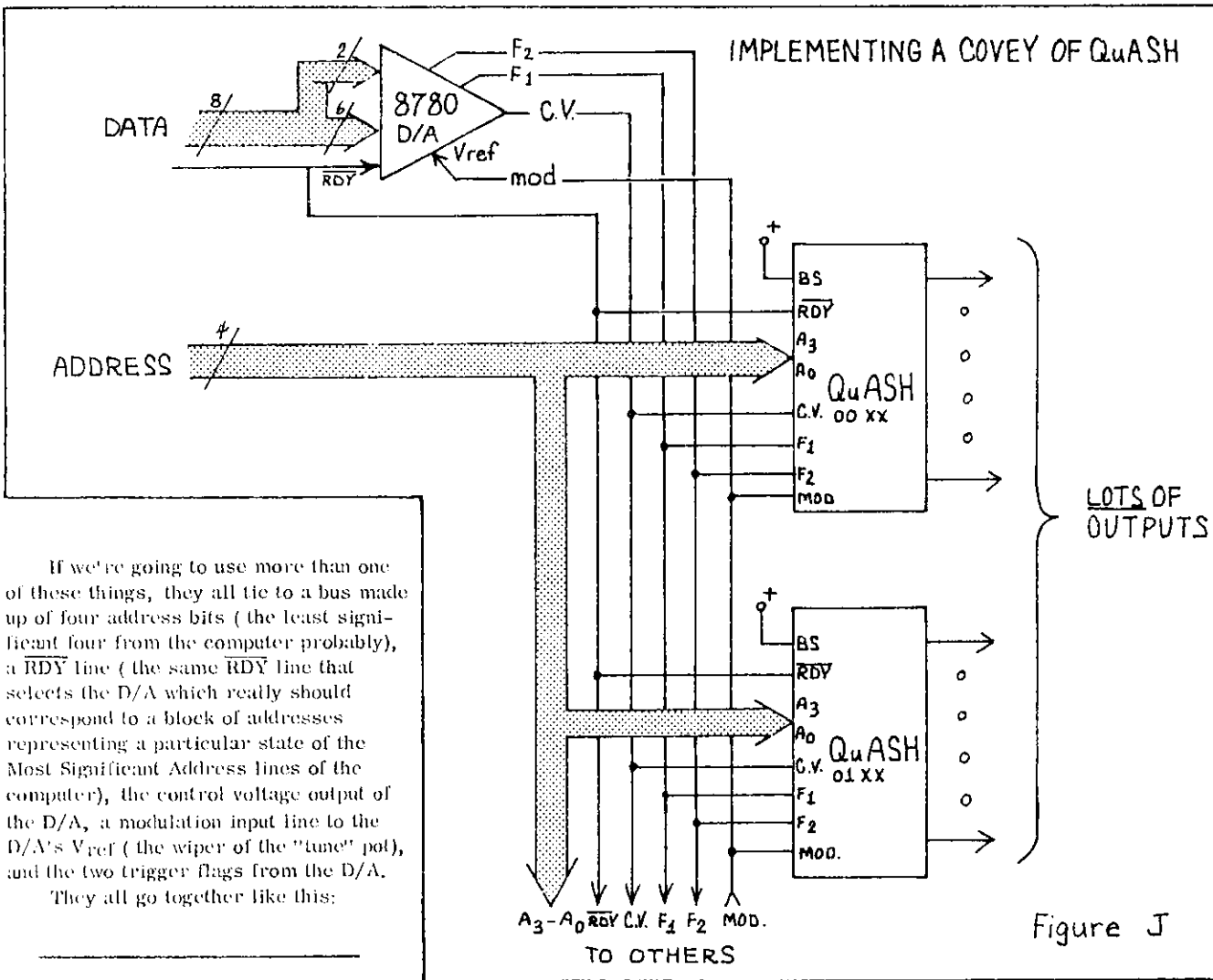
NOTES:

*(1) Those of you who have been thinking about this stuff for a while will, of course, recognize the imminent demise of the ADSR. Providing Attack, Decay, Sustain and Release parameters is one of the easier tasks to turn over to the computer entirely. On the other

hand, I've played with this some and can testify that varying the position of a knob is handier - in this case - than changing parameters in the memory of the machine. Some Hardware ADSRs mixed with some Software ADSRs seems a good compromise.

*(2) This off-hand statement is not meant to imply a wait in human terms (major fractions of a second), but rather a wait in machine terms - micro-seconds. You don't have to wait for a GLIDE to finish (for instance) before doing something else.

IMPLEMENTING A COVEY OF QuASH



If we're going to use more than one of these things, they all tie to a bus made up of four address bits (the least significant four from the computer probably), a RDY line (the same RDY line that selects the D/A which really should correspond to a block of addresses representing a particular state of the Most Significant Address lines of the computer), the control voltage output of the D/A, a modulation input line to the D/A's Vref (the wiper of the "tune" pot), and the two trigger flags from the D/A. They all go together like this:

Figure J

THE POLYPHONIC SYNTHESIZER

By
John S. Simonton, Jr.

LAB NOTES

We've come a long way over the last year in terms of developing a series of digitally interfaced modules that will allow computer control of music synthesizers. I suppose that the time has come to look at tying them all together, with the computer, and begin doing interesting things.

I had wanted to start with "the ultimate sequencer programs" but am not completely happy with them yet. They still need a little polishing.

Instead, we'll start with what should be another popular system:

THE POLYPHONIC SYNTHESIZER
Which is a much simpler job than the ultimate sequencer.

I would like to go through the system showing specific ways to do things for a variety of manufacturers equipment but that just isn't practical. Instead, we'll look at a completely PAIA based system and assume that if you are using different equipment you are familiar enough with it to make whatever changes are necessary.

Oh, one more thing before we begin,

be sure that you understand that there are a wide variety of ways to do polyphonic synthesizers. This is only one of them. I hope that the algorithm used here works for you. It's one of many, some with sort of special quirks that make them useful in certain situations but difficult to work with generally - This seems to be good general purpose way. Ready? We have lots to do and little space and time; here we go.

THE HARDWARE

Most of the hardware that we'll be using has been described here over the last year (or so). For the controller portion of this system we'll need:

- 1) AN ENCODED KEYBOARD
8782 or EK-3 retro-fitted equivalent
- 2) A COMPUTER

An 8700 in it's minimum configuration will run the programs that we'll list. A cassette interface system is useful to the point of being almost mandatory. We'll show some new panels and stuff to make it all pretty.

- 3) DIGITAL/ANALOG CONVERTER AND SAMPLE AND HOLDS the 8780/8781 system.

And, of course, we'll also need as much synthesizer as we think is necessary.

With all of the items listed, various wiring schedules have been mentioned for doing various non-computer things. We now need to establish some standards for this new use, a computer based polyphonic system.

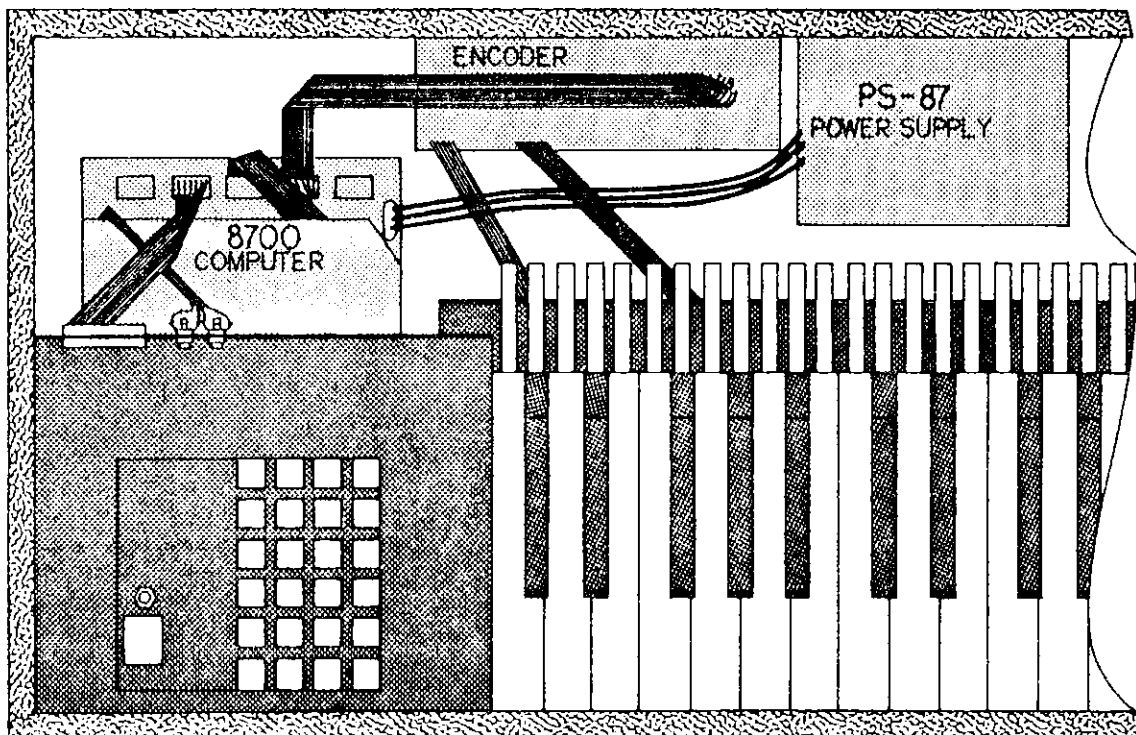
If we choose wisely, we should come up with a standard that has plenty of room for future growth. Some consideration has gone into the system which follows and I believe that it will serve our needs for some time to come.

Many of you will already have much of this wiring done, as much of it is simply an extension of what we've done before. Check carefully to be sure your wiring is to this new standard.

THE KEYBOARD

Let's go ahead and configure this system from the beginning so that the computer fits in the synthesizer cases that we've been using. All of the parts will fit in the case like this:

figure 1. computer/synthesizer sub-module placement.



PAIA 8700 COMPUTER, POWER SUPPLY AND KEYBOARD ENCODER
RETRO-FIT TO 4700 OR 8700 SERIES KEYBOARD.

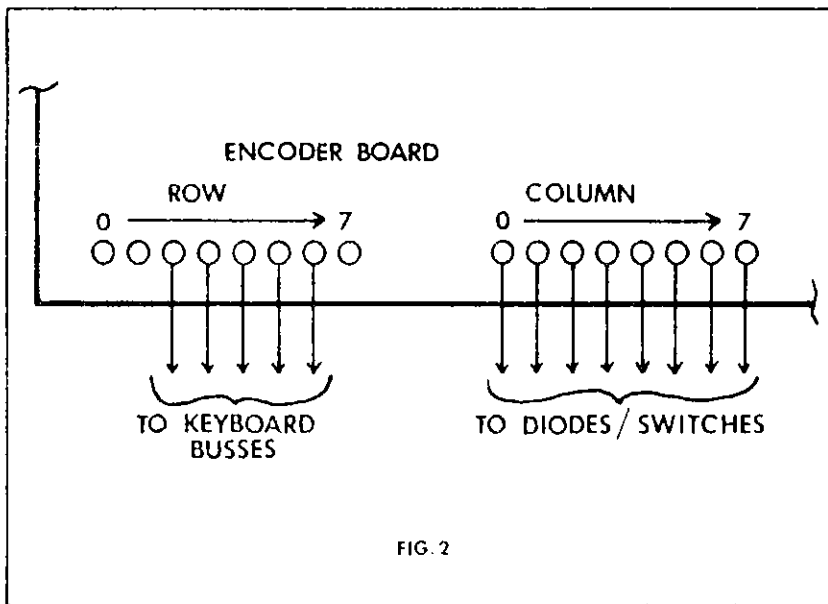


FIG. 2

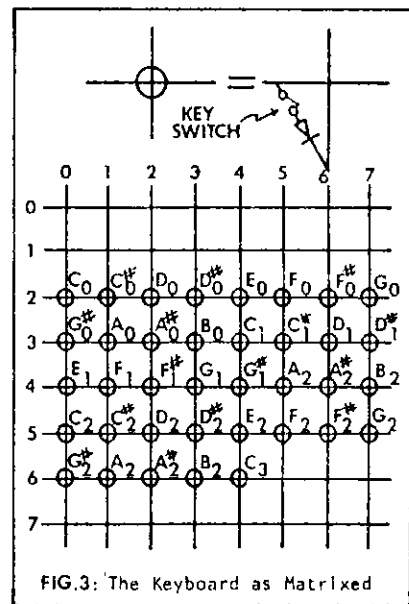


FIG.3: The Keyboard as Matrixed

At this stage you may have more dis-assembly to do than assembly. Particularly, the old control panel of the keyboard is removed to make room for the computer and any unregulated supply that was powering your keyboard encoder is replaced with a PS-87 which supplies all digital power for the entire system. This is going to give you a few parts for your "bench stock", the old power supply components and a couple of push-buttons, but some of the parts we will be re-using. Don't throw anything away.

KEYBOARD TO ENCODER CONNECTIONS

Maximum usability of the system would seem at first to depend on where the AGO keyboard switches appear in the key matrix. We want them in the middle so that we have as much room to transpose down in pitch as we do for up-scale transpositions. Some 8782 instructions had the keyboard placed 8 switch positions below where it should be for this ideal. The "column" connections are fine, but the "row" connections on these keyboards will need to be "slid up one" so that they conform to the configuration as shown in figure 2.

This will place the keyboard more or less in the middle of the matrix as shown in figure 3. This is really a fine point, and the system will work OK in most applications almost no matter where in the matrix the keys are, but go ahead and change now so that you won't be limited in the future.

ENCODER MODIFICATIONS

We don't need any of the "trick" things that we used when we didn't have a computer (the orgasmatronic glide circuit, etc.), just the bare-bones encoder. You may remove all push-buttons slide switches, pots etc.; most of these will come out when you remove the old front panel.

ENCODER TO COMPUTER

If your system previously had a

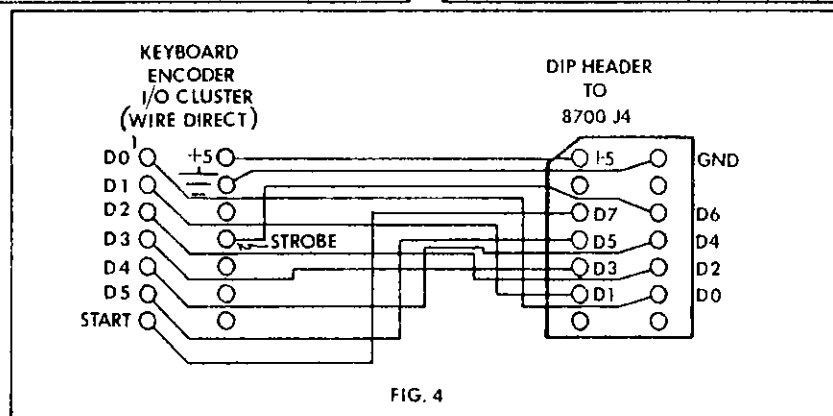


FIG. 4

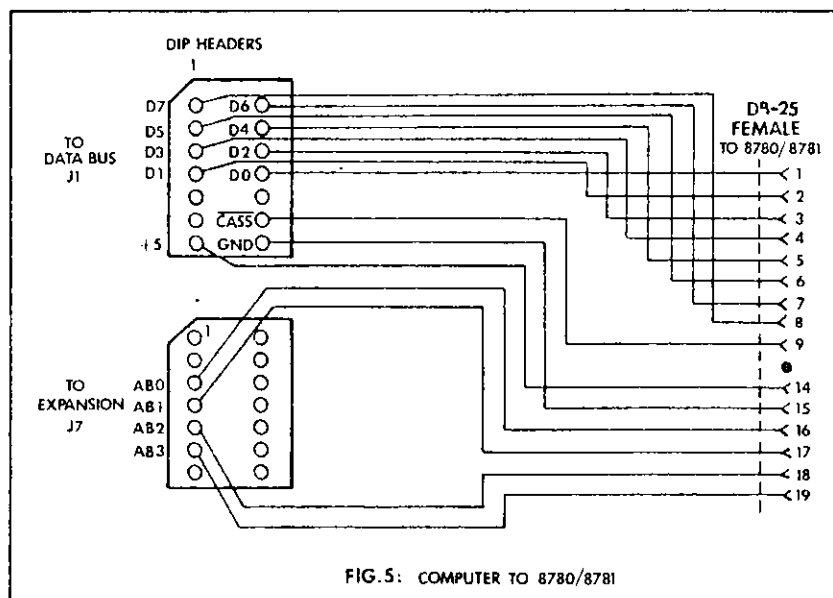


FIG.5: COMPUTER TO 8780/8781

DB-25 female connector tied to the output of the encoder, desolder it (carefully - whistling may make the job seem easier). In place of the DB-25 connector, we now need to terminate the output of the encoder in a DIP header that will mate with the INPUT PORT #1 (J4) connector on the rear edge of the 8700 computer board. These connections should be made as in figure 4.

These connections should also be

made carefully and the DIP header pins well heat-sunked to prevent melting the plastic header. NOTE that while many of the non-computer applications used the STROBE line to trigger the D/A, here we ignore this line and instead use the STROBE as the seventh data bit (D6) of the interface.

Similarly, the encoder's START line becomes the 8th data bit (D7).

Also, you will notice that power to

the encoder is picked up through this connection from the 8700 itself.

COMPUTER TO SYNTHESIZER HEAD

So that our resulting system can be easily broken down into two separate units (computer/keyboard and synthesizer head), this is the place to use the DB-25 connector that was salvaged from the old keyboard front panel.

Connections should be made between the female DB-25 connector and a pair of DIP headers like those in figure 5.

NOTE that the first header (P2) provides data lines and the CASS select signal (our 8780/8781 shares this output structure) while the second connector (P3) provides the address lines required by the QuASII.

8780/8781 WIRING

The male DB-25 connector that terminates the cable to the 8781 is wired in what is essentially an expanded version of our previous standard so that here you are faced more with adding wires than re-arranging them.

Connect these elements together as in figure 6.

This wiring schedule is examined in detail in the 8781 QuASII assembly manual. An important thing to notice here is the way the grounds are handled. Note that the ground pin on the rear of the 8780 board serves as the central ground for both analog (synthesizer) and digital power distribution. This grounding scheme is important to prevent ground loop problems and should be followed exactly. This entire 8780/8781 assembly should be mounted in the synthesizer head cabinet.

FINAL ASSEMBLY

Finally, make arrangements for physically mounting the computer in the keyboard case by first mounting the computer to a suitable front panel as shown. (See figure 7)*

And don't forget to provide a socket at the 8700's expansion connector (J7) or to mate P3 with this socket before assembling the computer/front panel. If the cassette interface is being used, terminate the input and output lines in miniature phono jacks as shown in figure 8.

Plug all the connectors together and you should be ready to load a program.

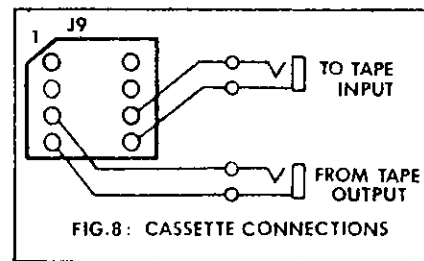


FIG. 8: CASSETTE CONNECTIONS

THE PROGRAM

The polyphonic program that we'll be using is called simply:

POLY 1.0

This program supports up to 8 output channels the way that it is written and can be easily modified to provide for more.

POLY 1.0 allocates synthesizer resources to keyboard requirements using this algorithm:

- 1) Output all notes appearing in the output buffer area (NTABLE) after adding the corresponding transposing figure from TTABLE. Go to 2.
- 2) Wait for keyboard scan to start and place a list of all keys currently being held down in the input buffer area (KTABLE). When buffer full or scan complete go to 3.

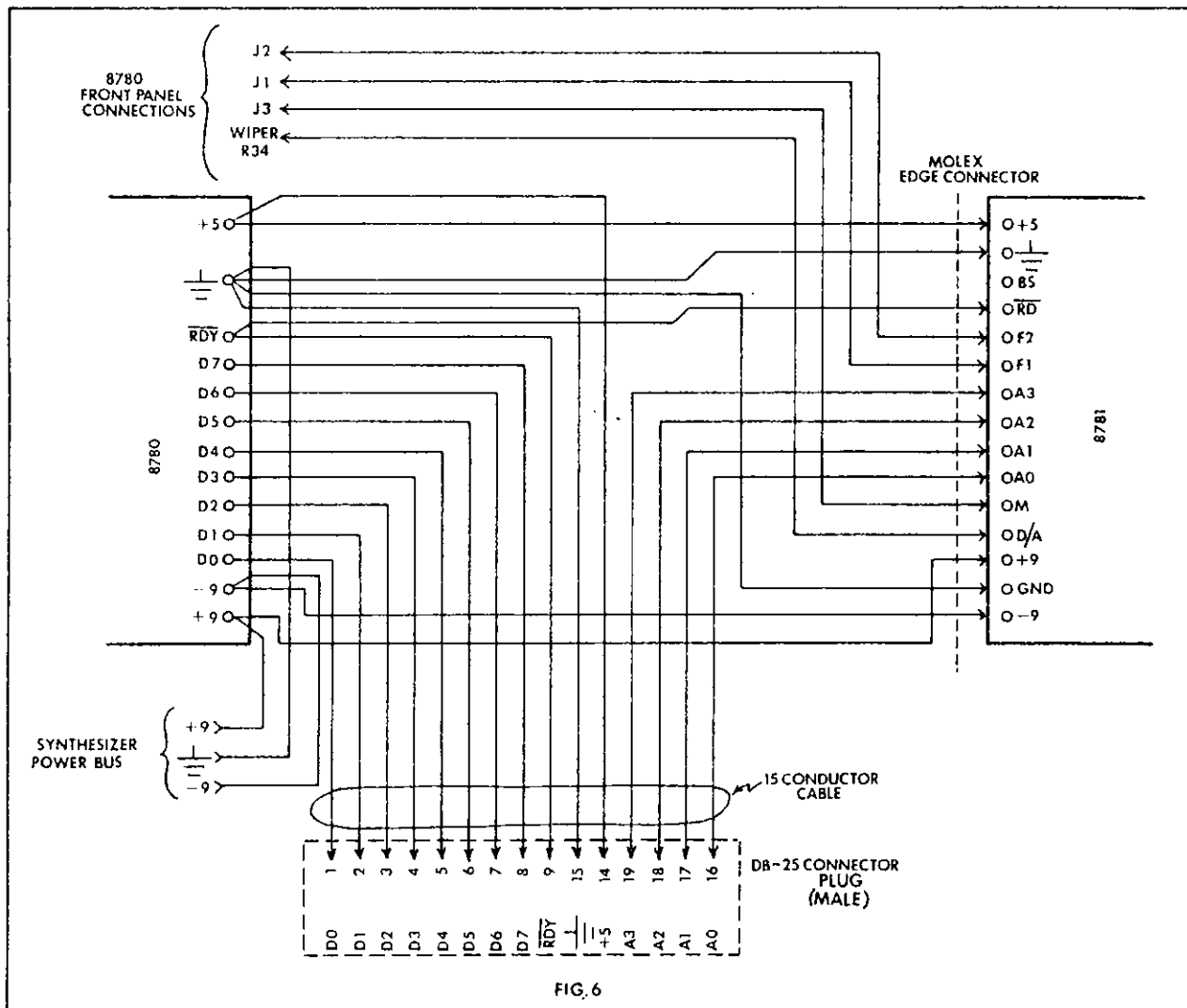
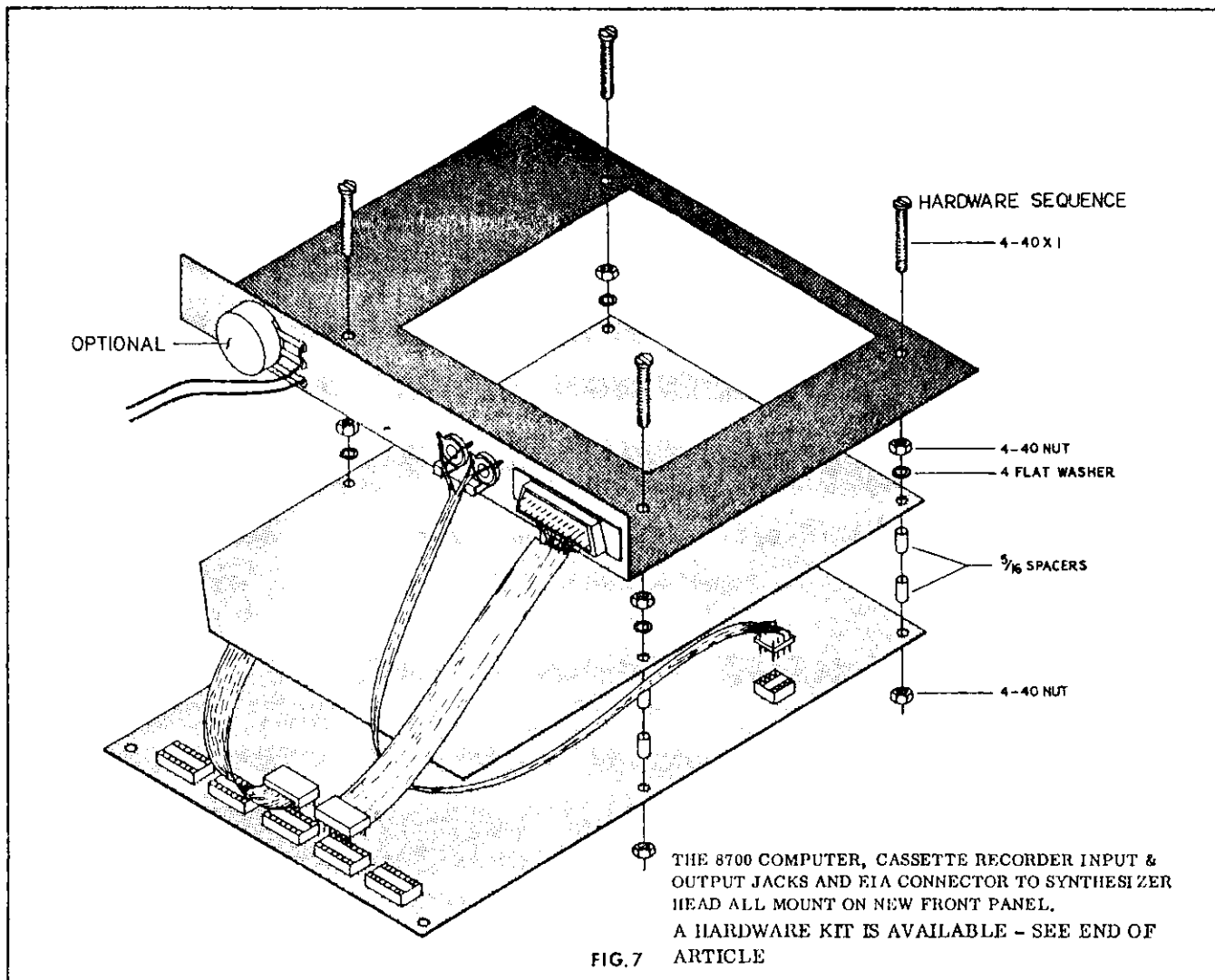


FIG. 6



3) Clear the trigger flags (D6) of all notes in NTABLE (the output buffer).
 4) Compare each entry in the input buffer (KTABLE) to each entry in the output buffer (NTABLE). If they are the same, set the trigger bit of the NTABLE entry and eliminate (zero) the entry from KTABLE. If all available outputs are used, or if all keys down find a home go to 1.
 5) Place the remaining input buffer entries in output buffer locations which do not currently correspond to a down key (those in which D6 is cleared). When all input data has been placed or all channels available have been used go to 1.

There are a number of subtle implications here and unfortunately not enough space to cover them all.

A couple of really important ones are that if we think of "new" notes as ones corresponding to keys that were just pressed, this method tries to place those new notes in output channels which at some point in the past were already producing those notes.

This prevents a string of identical

eight notes (for example) from being assigned to different outputs each time they're used. Notes, once assigned, tend to stay assigned regardless of other keyboard activity - they don't move around in a totally unpredictable fashion as with some analog multi-note keyboards.

It also means that once the number of output channels available is "used up" by down keys that need to be placed, all other keys that are down are simply ignored (this is exactly what you want).

One important aspect of the above is that the program must "know" how many output channels are available to it, otherwise there is the possibility that notes may be assigned to non-existent channels (ones that have no corresponding hardware, not too bad in itself) and further (the really bad part) future activations of the note will be assigned again to those non-existent outputs - producing "dead" synthesizer keys that seem not to be doing anything.

Memory location \$00EA contains the number of synthesizer channels available, more on this shortly.

THE PROGRAM

Shown on the next page is a disassembled listing of POLY 1.0.

Because, again, of space limitation we cannot re-print a fully documented version of POLY 1.0. It is supplied with the assembly and using manuals for the 8781 QuASH.

POLY 1.0 is also available in 8700 compatible cassette-tape form for \$4.00.

LOADING AND INITIALIZING POLY 1.0

If you have a cassette interface on your 8700 and the POLY 1.0 tape, loading is simply a matter of connecting your tape recorder to the cassette input connectors on the 8700 and loading the tape using the following entry sequence:

0-0-0-0-0-F-F-0-0-1-1-TAPE

If you don't have the CS-87 option, you must enter the code manually from the 8700 keyboard.

POLY 1.0

By John S. Simonton, Jr.
 © 1978 by PAIA Electronics, Inc.
 All rights reserved.

The cassette version of this program loads all of page zero of memory (its total requirement) and in the process initializes a couple of things that you will need to care for manually if the cassette is not available. When entering manually, be sure to set the number of outputs to correspond to the number you have available. For example, assuming that you have a system with a single QuASH, the number of channels available should be set to 4 using the following computer keyboard sequence:

```
RESET-0-0-E-A-DISP
0-4-ENTER
```

The tape version initializes the number of outputs at the most likely number of 4. If you want to use less channels (because of lack of modules, say) or have a system with more, do it as was shown above.

When entering the program manually, make sure the decimal mode flag in the status register is cleared by using this sequence:

```
RESET-0-0-F-F-DISP
0-0-ENTER
```

This is automatically taken care of when the tape version is loaded.

USING POLY 1.0

With everything connected, loaded and initialized, we're ready to begin making music. Go to the beginning of the program and begin running it.

```
RESET-0-0-0-6-RUN
```

If everything is working properly, we will see the 8700 displays counting quickly, incrementing by one for each scan of the keyboard. All of the QuASH outputs should be at a very low output voltage (the program initializes them as zero) and the trigger flags for each channel should be cleared.

As we press synthesizer keys, QuASH channels should "come alive" and produce control voltages corresponding to them. The trigger flags should be set if the key corresponding to the channel is currently down and clear when the key is released.

TWO MORE FEATURES OF POLY 1.0

While POLY is running, touching any of the keys from 0-3 on the 8700 keyboard (the first row of keys) causes the system to clear all QuASH channels to zero and wait for new data to be assigned. You'll figure out what this is good for as you become familiar with the system.

Maybe more importantly, touching any of the keys 4-7 (the second row

| | | | | | | | |
|-----|----------|-----|----------|-----|----------|-----|--------|
| 06- | A9 00 | LDR | #100 | 69- | A6 E9 | LDR | #E9 |
| 08- | A2 18 | LDR | #18 | 6B- | B4 CF | LDR | #CF, X |
| 0A- | 95 CF | STA | #CF, X | 6D- | F0 10 | BEQ | #0000 |
| 0C- | 0A | CLX | | 6F- | 02 09 | LDR | #09 |
| 0E- | 00 FB | BNE | #00FA | 71- | 0A | DEX | |
| 0F- | A2 08 | LDR | #08 | 72- | F0 F1 | BEQ | #0065 |
| 11- | 05 D7 | LDR | #D7, X | 74- | 98 | TYA | |
| 13- | 18 | CLC | | 75- | 55 D7 | EDR | #D7, X |
| 14- | 75 DF | ADC | #DF, X | 77- | 0A | ASL | |
| 16- | 80 00 09 | STA | #0000 | 78- | 0A | ASL | |
| 19- | 90 F7 09 | STA | #00F7, X | 79- | 00 F6 | BNE | #0071 |
| 1C- | A0 04 | LDR | #04 | 7B- | 98 | TYA | |
| 1E- | 88 | DEY | | 7C- | 15 D7 | ORR | #D7, X |
| 1F- | 00 FD | BNE | #00FE | 7E- | 95 D7 | STA | #D7, X |
| 21- | 0A | DEX | | 80- | 06 E8 | DEC | #E8 |
| 22- | 00 ED | BNE | #00EE | 82- | F0 31 | BEQ | #0085 |
| 24- | A2 03 | LDR | #03 | 84- | A6 E9 | LDR | #E9 |
| 26- | A9 00 | LDR | #00 | 86- | A9 00 | LDR | #00 |
| 28- | 95 CF | STA | #CF, X | 88- | 95 CF | STA | #CF, X |
| 2A- | 0A | DEX | | 8A- | F0 09 | BEQ | #0065 |
| 2C- | 00 FD | BNE | #00FD | 8C- | A9 00 | LDR | #00 |
| 2D- | A2 08 | LDR | #08 | 8E- | 02 09 | LDR | #09 |
| 2F- | 20 10 08 | BIT | #0010 | 90- | 0A | DEX | |
| 32- | 30 FD | BMI | #002F | 92- | F0 22 | BEQ | #0085 |
| 34- | 20 10 08 | BIT | #0010 | 94- | 04 CF | LDR | #CF, X |
| 37- | 30 0F | BMI | #000F | 96- | F0 F9 | BEQ | #0090 |
| 39- | 50 F9 | BVC | #00F9 | 98- | 95 CF | STA | #CF, X |
| 3B- | A0 10 00 | LDR | #0010 | 9A- | A2 09 | LDR | #09 |
| 3E- | 95 CF | STA | #CF, X | 9C- | 0A | DEX | |
| 40- | 00 10 08 | CMR | #0010 | 9E- | F0 17 | BEQ | #0085 |
| 43- | F0 FD | BEQ | #00FD | A0- | A9 40 | LDR | #40 |
| 45- | 0A | DEX | | A2- | 35 D7 | AND | #D7, X |
| 4C- | 00 EC | BNE | #0034 | A4- | 00 F7 | BNE | #0090 |
| 4E- | E6 E8 | INC | #E8 | A6- | A9 00 | LDR | #00 |
| 4F- | A5 E8 | LDR | #E8 | A8- | 35 D7 | AND | #D7, X |
| 4C- | 8D 20 08 | STA | #0020 | AA- | 95 D7 | STA | #D7, X |
| 4F- | EA | NOP | | AC- | 98 | TYA | |
| 50- | EA | NOP | | AE- | 15 D7 | ORR | #D7, X |
| 51- | EA | NOP | | B0- | 95 D7 | STA | #D7, X |
| 52- | A5 0A | LDR | #0A | BA- | 06 E8 | DEC | #E8 |
| 54- | 95 E8 | STA | #E8 | BC- | F0 02 | BEQ | #0085 |
| 56- | A2 08 | LDR | #08 | B5- | 00 D7 | BNE | #0000 |
| 58- | A9 0F | LDR | #0F | B7- | 20 00 FF | JSR | #FF00 |
| 5A- | 35 D7 | AND | #D7, X | B9- | 09 04 | CMR | #04 |
| 5C- | 95 D7 | STA | #D7, X | BB- | E0 03 | BCC | #000F |
| 5E- | 0A | DEX | | BD- | 4C 06 00 | JMP | #0006 |
| 5F- | 00 F7 | BNE | #00F7 | BF- | 09 03 | CMR | #03 |
| 61- | A9 09 | LDR | #09 | C1- | 00 05 | BCC | #0008 |
| 63- | 85 E9 | STA | #E9 | C3- | A9 2E | LDR | #2E |
| 65- | 06 E9 | DEC | #E9 | C5- | 4C 08 00 | JMP | #0008 |
| 67- | F0 23 | BEQ | #0023 | C7- | 4C 0F 00 | JMP | #000F |

on the 8700) provides a tuning function and causes all QuASH channels to produce the same note with the trigger flags set, allowing all oscillators to be set to the same pitch. The note produced corresponds to the 2nd C on a standard configuration 3 octave keyboard. THE CHANNELS MUST BE CLEARED AFTER TUNING by touching the first row of 8700 keys.

THE SYNTHESIZER

There are an almost unlimited number of ways to use the multiple control voltage produced by the QuASH and POLY 1.0.

You may want to use multiple VCO's mixed into a single voicing circuit, (See figure 9), or what amounts to a complete synthesizer for each control channel or anything in between, (See figure 10).

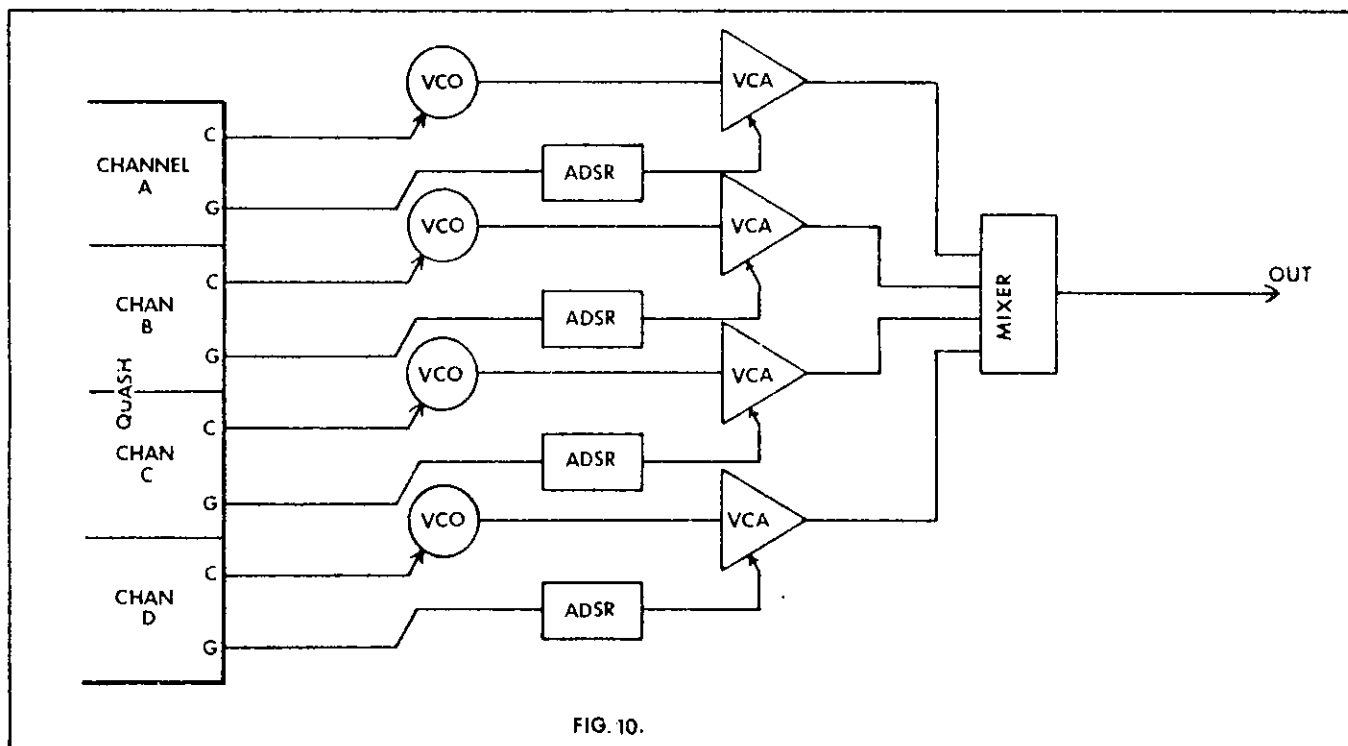
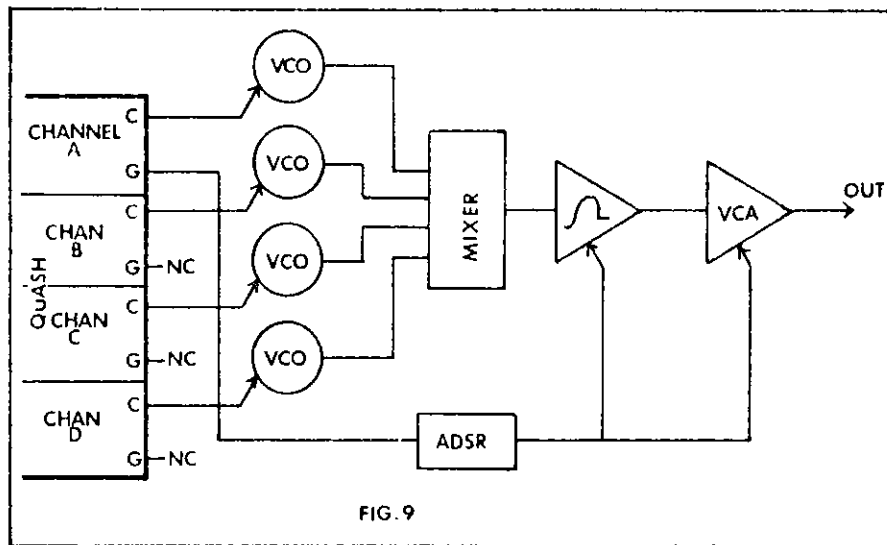
A word of advice: in your beginning stages of learning to use this system, you should try to stick to configuration in which all of the channels are producing the same "type" of sound - as close to identical as possible. As your skills progress and you develop a feel for how POLY 1.0 is going to massage data you can work up to using some output channels to set VCO pitches while

others control filter parameters (just an example - the number of possible combinations is extraordinarily large).

POLY 2.0 is under development and features the use of some QuASH channels as software controlled envelope generators, reducing the need for lots of these hardware modules.

POLY 3.0 provides for computer storage of sequences of chords or notes.

ONLY POLY 1.0 IS AVAILABLE NOW. The others are still a couple of months away. I mention them only because I want to make sure that we all understand that the nature of this new musical tool is a function of the program that is running and not so much of the hardware that it uses.



LAB NOTES: MUS 1

by John S. Simonton, jr.

With the new miracle ingredient - stg

With the exception of the bare-bones listing of POLY 1.0 that ran in the last issue, we haven't looked at any software—mainly because there was little to examine.

But MUS1 was just recently finalized, so that situation is beginning to change.

MUS1, for the benefit of those of you who haven't been waiting for it for the last six months, is what many would call "system firmware"—and since that has the sort of technical ring to it that tends to make things interesting, we'll call it that, too.

In almost any computer application there are some programs which, for one reason or another, are best handled as firmware—a name that these days means not software (which must be loaded from some storage media external to the computer) and not hardware (a permanently wired collection of gates, etc. which cause a specific, set sequence of actions to take place) but something betwixt and between; most usually, software that is contained in a PROM somewhere.

The most obvious firmware is a non-program such as PIEBUG. Since program is the thing that allows for the entry of data and instructions into the memory of the computer in the first place (as well as usually providing whatever de-bugging and editing features the designer thought were important and/or had room for), it is at least inconvenient to have to load it every time it is needed. Much better to have it in a dedicated PROM where it is always available for immediate use.

The firmware of MUS1 is roughly analogous. These are universally useful routines that, with rare exceptions, will be used with everything we do musically. It's a waste of time and resources to have to load them to RAM from tape (or worse yet, manually) every time they're needed. A PROM is their happiest home. In our 8700 Computer/Controller, MUS 1 is a 1702A PROM that occupies the address range \$D00-\$DFF (IC-17).

Examples? OK, the keyboard reading routine (LOOK). It isn't particularly long or complicated (a little over 30 bytes) but we're going to need it every time we turn on the system— even if it isn't used to read the keyboard, it's the thing that our protocols dictate will be tempo-determining element in the item (based on the clock rate of the encoder). At some future date the occasion may arise when we can examine this in detail. Today, it's not the point.

The QuAsh drivers (called NOTE)—same thing—we're going to need them for almost everything we do. Why bother to load them?

In addition to these two routines, MUS1 also contains:

INIT: an initialization routine that takes care of setting various variables and buffer areas to a known, acceptable state (as opposed to the random numbers they will contain when power is first applied.)

POLY: essentially the polyphonic (I still prefer polytonic) allocation algorithm from POLY1.0, except refined somewhat to take less memory space.

TRGN: The new miracle ingredient—Software Transient Generators (STG). A routine that will serve as a software substitute for ADSRs.

OPTN: A very simple option selecting program that allows the remaining firmware of MUS1 to be tied together into a 16 voice polyphonic synthesizer with or without software transient generators—without having to lead any additional software (though several parameters will need to be initialized manually).

All of this is pretty straight-ahead code that should be understandable from the documented listing that appears at the end of this article— you may need to refer back to previous articles in this series for background information; "In Pursuit of the Wild QuAsh" (reference Polyphony, July '77) and "What the Computer Does" (reference Polyphony 4/76) would be particularly useful ones.

Two exceptions, NOTE and TRGN, need some additional explanation — they introduce some new ideas.

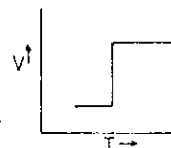
In an embryonic form, NOTE was a part of POLY1.0. It is the responsibility of this routine to take individual entries from the output buffer area (NTB1), add to it the corresponding entry from the Transpose buffer area (TTBL) and output the results to the QuAsh channels. Some aspects of the significance of the addition that takes place will be seen when we look at TRGN— for now, it will suffice to say that this will be an extraordinarily handy convention in a number of cases.

A more important function of NOTE is to make sure that what comes out of the QuAsh channels has no annoying glitches that may be artifacts of the D/A and multiplexing process. In an earlier story, we looked at one of the annoyances — the fact that our 8780 D/A, though quick, takes a finite amount of time to

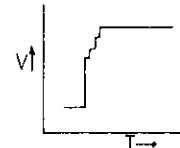
change from one value to the next and if appropriate settling time is not allowed between writes to the QuAsh channels we will be able to hear the changes as a slight "buzz" in each of the channels. The solution here is to output the data first to a "dummy channel" that is occupied solely by the D/A, with no corresponding QuAsh, followed immediately by a write of the identical information to an output which does correspond to a QuAsh channel. The first write allows the D/A to settle while the second strobes the settled output into the appropriate QuAsh channel.

And here we come face to face with the next problem; the QuAsh really need some settling time since they are at their heart nothing more than an RC circuit.

As long as we are thinking in terms of small systems (8 output channels or less) this is not a big problem since it can be dealt with simply by delaying after writing to one QuAsh but before setting up the next. If the delay is not long enough, we will hear changes from one value to the next not as an instantaneous change, but rather as a series of steps from the initial value to the final one:



We want this



But if the QuASH settling delay is not long enough, will get this.

figure a

In larger systems, this constant delay approach is not a practical solution because there is not enough time during alternate "dummy" scans of the keyboard (the time which our conventions allow for processing, output driving, allocations, etc.) to allow all of the output channels the luxury of a delay. The time comes for the keyboard to be read again (or other things to happen) and the processor is still busy waiting for all of those QuAsh to get to the right value.

The key to the solution of this problem is to notice that there is really only one set of circumstances under which the long QuAsh-settling delay is required, and that's when the output of one of these channels must change from one value to another (which happens only a small percentage of the time) and then, only when the glide of the channel is turned off. (If

the glide is on, its integrating action will smooth out the steps; and, in fact, a short write time is preferable here since it will serve to increase the time required for the glide.)

The actual solution is what I feel we should call "DYNAMIC QuAsh DRIVERS"—a small block of programming, more or less in the middle of NOTE.

This part of the program first checks to see if the glide control bit (the most significant bit of the data just written to D/A and S/Hs) was turned on or not. If we are in "glide mode," no delay is required so the program immediately goes to see if there are any channels left to write; if there are, it services them.

If the glide is not on, we have a candidate for dynamic operation so the dynamic mode switch is checked (more later) and if this option is selected the current data is compared to the data that was previously written to this channel (requires a new table that we've generated called "LAST") and if they're different (a change), the program goes into the delay that allows the output of the QuAsh to instantaneously (apparently) step from its previous value to the next one. The new value is saved in LAST (for use next time) and if there are more channels to do— it does them.

SOFTWARE TRANSIENT GENERATORS

Here we begin, for the first time, to replace some of the elements that constitute traditional synthesizer hardware with software that performs the same function (hopefully as well, or better) with less costly hardware. STGs are a good place to start because they're not super difficult to implement.

Just like their hardware equivalents, STGs respond to a note which has just been triggered (pressed on the keyboard) by producing a voltage that rises at a controlled Attack rate. After reaching some peak value, the voltage then drops at a Decay rate until it reaches a pre-set Sustain level where it stays as long as the note remains triggered. When the key is released, the voltage drops to its lowest level at the Release rate.

Computing the number which represents the current value of the transient is only slightly more complicated than adding, subtracting and comparing.

Unlike an ADSR, an STG has no knobs to set, in their place you enter numbers setting Attack rate, etc. into the computer.

Perhaps the biggest problem having to do with STGs is deciding where they should come out. Oh, the QuAsh channels, obviously; but which ones? Of the numerous

possibilities, we've selected the convention of having pitch setting voltages (those that correspond to notes) and transient voltages come from alternate QuAsh channels, primarily because this will work nicely with some stuff under development (or consideration, at least), without making obsolete all of the hardware that we've accumulated up to now.

This implies two distinct modes of operation; the first in which the STGs are not asserted and POLY assigns notes to sequential QuAsh channels; and, the second mode (STGs on), in which notes are assigned by POLY* to the odd number QuAsh channels (first, third, etc.) while transients are produced at the even number outputs (second, fourth, etc.).

The note produced at the first QuAsh output has a corresponding transient happening at the second output, and so on. Just as if the trigger from the first channel were patched to the input of an ADSR whose output was somehow tied to the output of the second QuAsh channel.

This would seem a good place to mention (in case it's not already obvious) that in this implementation all of the STGs produce the same kind of transient, and for the kinds of things that we're doing now, this is how it should be. It may also be worth mentioning that while the transients are all the same, they are totally independent where following the triggered and released states of their respective note channels is concerned.

There are also some internal details which muddy the STG waters. For instance, a key that is currently down may require a transient function that is either in the Attack cycle (increasing) or Decay/Sustain cycle (decreasing or holding) depending on its past history (had it already peaked?). Somewhere we need to save information on which cycle the transient is actually in.

Another, somewhat interrelated, problem concerns the smoothing of the transient waveform. Under most conditions, the glide of the QuAsh channels that are being used as transient outputs should be turned on so that a smoothly increasing or decreasing function is produced. But, the glide can't always be on because that would limit the maximum attack rate.

Without having the space to cover it entirely, I can only state that the solution to both of these difficulties lies in the use of the Transpose table and remembering that the data stored in TTBL entries is added to the output parameter in NTBL (where we're storing the actual current

* Note that POLY checks to see if the STGs are turned on as it assigns notes to outputs.

value of the transient) before the output operation takes place. Note also that while the data in NTBL is manipulated extensively by POLY and TRGN (as the, calculate, allocate, - regurgitate?) TTBL is untouched by computer hands, and this makes it an ideal place to save control type functions. Not only transpositions, but a place that glide and trigger bits and such can be permanently set.

These locations are so handy for this application that in TRGN they have been re-named CWRD (Control-Words... but do not be confused, this is still our old friend TTBL and has no relationship at all to the System Control Word-CTRL) and it is here that we keep track of the A/D/S state of each of the transient channels.

Also, to help me keep things straight in my own mind, the NTBL bytes that are used to store the current value of the transient have been re-named PARM (parameter); but, again, this is the same physical area as NTBL.

NOW, HOW DO WE USE ALL THIS?

Perhaps the best way to begin an essay on how to use MUS1 is to state one of the functions that it was devised to perform

As you are no doubt beginning to realize, we've carefully developed a system that will have applications far beyond what we've discussed to this point. It's complex; and while the complexity implies unmatched versatility, it undeniably has its intimidating aspects.

At one level of use, MUS1 should reduce this intimidation by giving the user an instrument with a specific (though within certain limits alterable) personality the instant that it's turned on, without having to hassle around with loading any additional programs (success) or variables (well...)

Also, these program modules should be written so that they easily interface with future expansions of the system, either hardware or software, so that, when needed, they can be accessed by programs offering distinctly different personalities (success here maybe— only time will really tell).

While we've reduced the intimidation, we've not eliminated it entirely because even when using MUS1 as a stand-alone personality there are some variables which must be initialized before you begin to play— some information that the system must have in order to operate properly. This data could be part of the PROM, but not without significantly compromising versatility.

For instance, we've mentioned in passing a couple of times the System Control Word-CTRL. This is a single word in the com-

puter's RAM memory at location \$0E8.

It is most helpful to visualize CTRL a collection of eight "switches", each bit representing one switch. To MUS1, only two of these switches have any significance- D7, which turns the STGs on and off, and D6, which enables or disables the dynamic mode option. The rest are reserved.

Every time you power up the system, CTRL must be set so that the desired options are selected- there is no default setting that is part of MUS1. If you want dynamic mode (which you should, for now) then bit 7 should be turned on. If you want STGs, bit 8 must be set.

The 4 possible combinations of these 2 bits then have the following significance:

| binary | hex | action |
|----------|------|----------------------------|
| 00000000 | \$00 | STGs off; dynamic mode off |
| 01000000 | \$40 | STGs off; dynamic mode on |
| 10000000 | \$80 | STGs on; dynamic mode off |
| 11000000 | \$C0 | STGs on; dynamic mode on |

CTRL is not the only variable which must be initialized manually. There's also:

EXTERNALLY INITIALIZED VARIABLES

LOC. LABEL USE

\$E8 CTRL SYSTEM CONTROL WORD
D7 SET TURNS ON
TRANSIENT GENERATORS
D6 SETS DYNAMIC MODE

\$E9 ODLY SETS OUTPUT DELAY.
IN DYNAMIC MODE, 120
RECOMMENDED.

\$EA OUTS NUMBER OF HARDWARE
SUPPORTED CONTROL
CHANNELS AVAILABLE

-- AND TRANSIENT PARAMETERS --

\$EA ATC ATTACK RATE
\$EB DCY DECAY RATE
\$EC SUST SUSTAIN LEVEL
\$ED RELS RELEASE RATE
\$EE DUR DUR VALUE (SEE TEXT)

RATES \$01 (SLOW)
\$3F (FAST)
LEVEL \$01 (MINIMUM)
\$3F (MAXIMUM)

Most of these are easily understood or have been examined in the past, so we won't go into any great detail. A few points are worth mentioning, however.

ODLY- this is a number that represents the delay that the QuAsh drivers will use, when required. For normal use, a value in the range of \$20-\$30 is most appropriate.

OUTS- this variable tells the POLY subroutine how many output channels it has to work with, so that notes don't get lost; we talked about this last time. Now we need to notice that when the STGs are asserted we should think of the QuAsh channel that is producing the transient as simply an extension of the channel producing the note. In other words, the two QuAsh channels constitute a single "hardware supported" channel. A single QuAsh represents two such channels.

ATCK/DCY/SUST/RELS- When the

transient generators are turned on, we also need to enter the attack, delay, sustain and release parameters that we want produced. These four entries should need little explanation other than the examples which follow shortly; their range is from \$01-\$3F, with \$01 representing the lowest rate or level and \$3F the highest.

PEAK- this fifth transient parameter needs a little extra attention. PEAK has only one use; it determines whether the transient produced is going to be percussive (quickest possible attack and full ADSR segments) or non-percussive. In the non-percussive mode, the glide is on for all segments of the transient and the Decay and Sustain states of the transient are eliminated entirely.

In fact, there is only one bit in the word PEAK that is changed to select one of these two options- the most significant bit. The remaining seven bits should (for now- until you have a real feel for what's happening) be set to \$3F (00111111 in binary). If the most significant bit of this word is cleared, you're in percussive mode. If the bit is set (so that PEAK contains \$BF - 10111111 in binary) you are in non-percussive mode.

The differences between the two are great. Assume for a moment that we have set the ADSR parameters at \$3F/\$04/\$20/\$01 respectively (fastest attack/moderated decay/medium sustain/slowest release) and that we are only going to change the PEAK parameter. If PEAK contains \$3F (percussive mode), a 'scope display of the transient will look something like this:

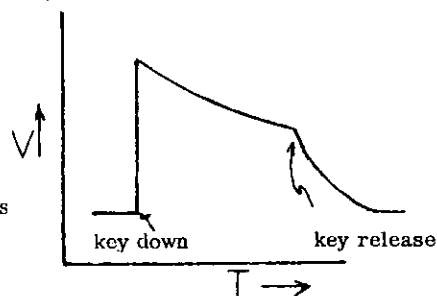


figure b

Setting PEAK to \$BF (non-percussive) produces this result:

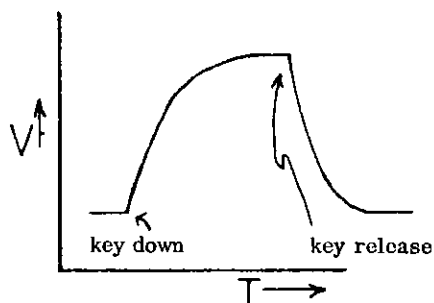


figure c

Because the glide is now on during the entire attack cycle and the Decay and Sustain portion of the transients are eliminated. Straightforward stuff, really.

We need to cover an example of system set-up before we wind up, but first must notice that the effect of having the PEAK parameter are far more far-reaching than we've been able to cover in detail. A quick example:

ADSR parameters set to \$10/\$04/\$20/\$01 and PEAK containing \$3F will produce this kind of transient:

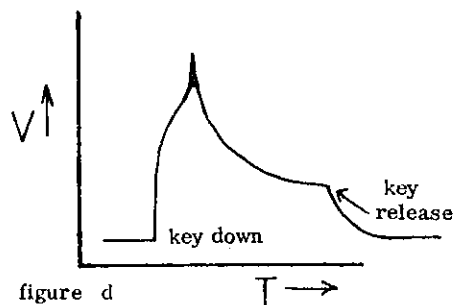


figure d

which, when heard, starts out with a non-percussive kind of "swell" with a percussive "plop" added at the last instant before the transition to the Decay and Sustain cycles. This would seem to be a unique and useful transient that isn't produced by traditional ADSRs.

Along the same lines, the TSGs can be considered to be "better" than our hardware ADSRs in that they need not finish the Attack cycle before transitioning to the Release state. If a key is released before its transient has gone all the way to PEAK, the transient immediately switches to the release state. This is frequently called "muting" and it offers the possibility of effective control of expression directly from the AGO keyboard.

A SUMMARY, OF SORTS

So, we've gotten our hands on a MUS1 PROM and are ready to start doing things. What has to be done first? Really very little.

First, the System Control Word, Output Delay and number of hardware channels available must be set. For example:

| keystrokes | explanation |
|------------|-------------------------------------|
| 0-E-8-DISP | sets monitor pointer to \$E8-CTRL |
| C-0-ENT | sets \$E8-asserts STGs dynamic mode |
| 3-0-ENT | sets ODLY value |
| 0-2-ENT | sets output channels at 2 |

these entries define the personality of the Instrument as a 2 voice polyphonic synthesizer (notes from channels A & C) with software transient generators (which appear at QuAsh channels B & D).

Next, we must set the transient parameters to the desired values:

| keystrokes | explanation |
|------------|------------------------------------|
| 0-B-A-DISP | sets monitor pointer to \$BA- ATCK |
| 3-F-ENT | sets shortest attack |
| 0-4-ENT | sets moderate decay |
| 2-0-ENT | sets moderate sustain |
| 0-1-ENT | sets slowest release |
| 3-F-ENT | percussive mode |

and you may recognize these parameters as being those that we examined in the illustration earlier.

Finally, we simply begin running the program:

| keystrokes | explanation |
|------------|---|
| D-0-0-DISP | sets monitor pointer to beginning of OPTN |
| RUN | presto- the program runs |

A typical patching configuration that would be consistent with these entries would look something like this:

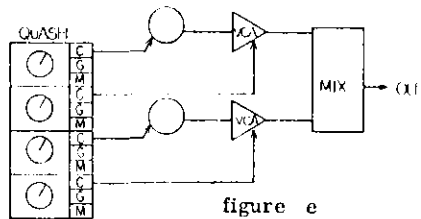


figure e

Oh, yes- I almost forgot. OPTN, like POLY 1.0, uses the 8700 keyboard to control two important functions. While OPTN is running, touching key 0 of the control keyboard will cause the entire system to be re-initialized. Not the entries that we made manually- those remain unchanged, but all the notes and transients go immediately to zero level.

Similarly, touching key #1 produces a tuning function that makes the synthesizer respond as if all the channels were seeing the second C on a three octave keyboard held down. The transients go, the notes play, etc. After tuning, be sure

to re-initialize the system by touching control key #0.

I prefaced one of the earlier paragraphs with "at one level of use." In all of the preceding words, that's all that we've examined- one level of use (the simplest and most obvious level, at that.) I've also referred in the past to "software modules" which can be strung together in different ways (just as can hardware modules) to produce different effects and personalities. MUS1 is the first set of these modules.

With more regret than you can imagine, I haven't the space here to go into all of the implications of this (even if I knew them all, which I'm sure I don't).

Providing you're more than just casually interested, you should spend some time trying to understand how MUS1 works internally (there are numerous different entry points to the routines that we haven't covered - for instance). I believe that the time investment will be wisely made.

```

*****
+
+          MUS1
+
+   BY JOHN SIMONON
+   (C) 1978 PAHA ELECTRONICS, INC
+   ALL RIGHTS RESERVED
+
+
+   SYNTHESIZER SUBROUTINES
+   ** AND **
+   MULTIPLE OPTION POLYPHONIC
+   ALLOCATION PROGRAM WITH
+   SOFTWARE TRANSIENT
+   GENERATION
+
+*****

POLYPHONIC SYNTHESIZER / OPTION
SELECTION

*****

INIT .DL 0021
POLY .DL 0071
TRGN .DL 00C3
NOTE .DL 002B
DECD .DL 0F00
FILL .DL 0052
DISP .DL 0820
CLK .DL 00BF

OPTION TIES MUS1 FIRMWARE
TOGETHER INTO A POLYPHONIC SYNTH
WITH OR WITHOUT TRANSIENT GENER-
TION. W/NO DYNAMIC QUASH DRIVERS

ALSO USES PIEBUG DECODE AND
ASSIGNS KEY #0 AS SYSTEM CLEAR
AND #1 AS TUNE ~ EQUIVALENT TO
ALL CHANNELS 2ND "C" ON KBD DOWN

0000- 20 21 00 OPTN JSR INIT      :ZERO ALL BUFFS
0003- 20 71 00 LOOP JSR POLY     :ALLOCATE CHANS
0006- 20 C3 00 JSR TRGN         :NEW TRANSIENTS
0009- 20 2B 00 JSR NOTE        :OUTPUT-READ AGO
000C- A5 BF LDA *CLK           :GET CLOCK VALUE
000E- 80 20 08 STA DISP       :RAZZ-NA-TRZZ
0011- 20 00 0F JSR DECD        :CHECK COMMANDS
0014- C9 01 CMP 01            :0? 1? >1?
0016- 30 E8 BMI OPTN          :0: CLEAR ALL
0018- 00 E9 BNE LOOP         :>1: KEEP ON
001A- A0 5C LDY 5C           :1: TUNE 2ND C
001C- 20 52 00 JSR FILL       :KEYS ALL DOWN
001F- F0 E2 BEQ LOOP         :BRANCH ALWAYS

INIT
INITIALIZATION ROUTINE
*****

CTRL .DL 00E8
TBEG .DL 00BF

INIT CLEARS INPUT BUFFER (KTBL)
OUTPUT BUFFER (NTBL) AND TRANS-
POSE BUFFER/CONTROL WORDS (TTBL)
HEXADECIMAL MODE IS SELECTED

ENTER AT INT0 TO FILL TABLES
WITH CHARACTER FROM ACCUMULATOR

0021- A9 00 INIT LDA 00        :PREPARE TO ZERO
0023- A2 28 INT0 LDX 28       :SET POINT/COUNT
0025- 08 CLD                 :SET HEX MODE
0026- 95 BF INT1 STA *TBEG,X  :ZERO BUFFER
0028- CA DEX                 :POINT TO NEXT
0029- 00 FB BNE INT1        :SOME LEFT -LOOP

NOTEOUT/LOOK
16 CHANNEL QUASH DRIVERS AND AGO
KEYBOARD READING ROUTINE

*****

CTRL .DL 00E8
ODLY .DL 00E9
KTBL .DL 00DF
NTBL .DL 00CF
TTBL .DL 00BF :ALSO CLK
LAST .DL 00A9
S/H .DL 00AF
D/H .DL 00A0
RBD .DL 0010

*** NOTEOUT ***
DYNAMIC QUASH DRIVERS
GETS NOTES TO BE PLAYED FROM THE
OUTPUT BUFFER (NTBL) AND ADDS
TRANSPOSING VALUE FROM TRANSPOSE
BUFFER (TTBL). OUTPUTS RESULT

002B- A2 10 NOTE LDX 10      :SET POINTER
002D- 85 CF NO0 LDA *NTBL,X  :GET NOTE
002F- 18 CLC                 :PREPARE AND
0030- 75 BF ADC *TTBL,X     :ADD TRANSPOSE
0032- 80 00 09 STA D/H      :LET D/H SETTLE
0035- 90 EF 09 STA S/H,X    :WRITE TO S/H

NOW THE DYNAMIC PART: IF GLIDE
IS ON, DELAY IS SKIPPED. IF NOTE
IS SAME AS LAST PLAYED (IGNORING
CONTROL BITS D6 & D7) DELAY IS
SKIPPED. IF NOT IN DYNAMIC MODE
AND NO GLIDE, DELAY ALWAYS TAKEN

0038- 30 0F BMI NO2         :GLIDE? NO DELAY
003A- 09 80 ORA 80         :IGNORE FLAGS
003C- 24 E8 BIT *CTRL       :DYNAMIC MODE ?
003E- 50 04 BVC DELAY      :NO, JMP TO DELAY
0040- 05 A9 CMP *LAST,X    :COMPARE TO LAST
0042- F0 05 BEQ NO2        :SAME: SKIP DELAY
0044- A4 E9 DELAY LDY *ODLY  :GET DELAY VALUE
0046- 88 NO1 DEY           :DECREMENT DELAY
0047- 00 FD BNE NO1        :LOOP TIL DONE
0049- 95 A9 NO2 STA *LAST,X :FOR NEXT TIME

```

```

004E- CA           DEX           :POINT TO NEXT
004C- D0 DF           BNE NO0           :SOME LEFT -LOOP

:
:LOOK WAITS FOR THE BEGINNING OF
:AN "ACTIVE" SCAN-BEGINNS PUTTING
:THE NUMBERS OF KEYS DOWN IN SE-
:QUENTIAL IN-BUFF WORDS. WHEN
:SCAN DONE REMAINING IN-BUFF IS
:ZERO'0.

004E- E6 BF       LOOK INC *TTBL     INCREMENT CLOCK
0050- A0 00           LDY 00           :PREPARE FOR CLR
0052- A2 08       FILL LDX 08           :SET UP POINTER
0054- A0 10 08    LK2  LDA KBD           :WAIT FOR
0057- 20 FB           BHI LK2           : "ACTIVE" SCAN
0059- A0 10 08    LK3  LDA KBD           :GET KEY
005C- 30 0F           BMI D0NE           :END SCAN? -CLR
005E- 2A           ROL                :STROBE TO D7
005F- 10 F8       BFL LK3           :D7=0, NO STROBE
0061- 6A           ROR                :RESTORE DATA
0062- 95 DF       STA *KTBL,X         :TO IN BUFFER
0064- C0 10 08    LK4  CMP KBD           :NOW WAIT FOR
0067- F0 FB           BCC LK4           :NEXT KEY
0069- CA           LK0  DEX                :PNT TO NEXT BUF
006A- D0 ED       BNE LK3           :SOME LEFT -LOOP
006C- 60           RTN  RTS             :LEAVE
006D- 94 DF       D0NE STY *KTBL,X     :ZERO IN-BUFFER
006F- 30 F8       BHI LK0           :BRANCH ALWAYS

:
: POLY
: A LIMITED RESOURCE ALLOCATION
: ALGORITHM
:*****
OUTT .DL 00EB
OUTS .DL 00EA
CTRL .DL 00E8
KTBL .DL 00DF
NTBL .DL 00CF

:POLY-FIRST HALF OF ALGORITHM
:IN THIS BLOCK DE-ACTIVATED CHANS
:ARE REACTIVATED IF THE DATA THEY
:CONTAIN APPEARS IN THE IN BUFFER
:
:D7 IN CTRL SET - ALTERNATE MODE
:D7 " " CLK - SEQUENTIAL MODE

0071- A5 EA       POLY LDA *OUTS       :# OF OUT CHANS
0073- 85 EB           STA *OUTT       :USE AS COUNTER
0075- A2 10       LDX 10           :PREPARE PNT/CNT
0077- B5 CF       POL0 LDA *NTBL,X     :GET NOTE
0079- F0 27       BEQ NKKY         :0-OLD KEYS DONE
007B- 29 7F       AND 7F           :CLEAR D7
007D- 09 40       ORA 40           :SET D6
007F- A0 09       LDY 09           :PREPARE PNT/CNT
0081- 98           LP0  DEY           :POINT NEXT KEY
0082- F0 12       BEQ NEXT         :DONE -NEXT NOTE
0084- D9 DF 08    CMP KTBL,Y         :SAME AS KEY?
0087- D0 F8       BNE LP0         :NO -NEXT KEY
0089- 95 CF       STA *NTBL,X       :SAVE NOTE D6=1
008B- C6 EB       DEC *OUTT         :ONE LESS OUTPUT
008D- 10 33       BEQ 001         :NONE LEFT-LEAVE
008F- A9 00       LDA 00           :OR PREPARE AND
0091- 99 DF 00    STA KTBL,Y         :ELIMINATE KEY
0094- F0 04       BEQ LP1         : & BRANCH ALWAYS
0096- 29 BF       NEXT AND 0BF       :CLEAR TRIG (D6)
0098- 95 CF       STA *NTBL,X       : & RESTORE NOTE
009A- 24 E8       LP1  BIT *CTRL      :ALTERNATE MODE?
009C- 10 01       BPL SKP1         :NO -DEC ONCE
009E- CA         DEX                :YES-DEC. THICE
009F- CA         SKP1 DEX           :POINT NEXT NOTE
00A0- D0 D5       BNE POLY         :SOME LEFT -LOOP

:
:NEWKEY - SECOND HALF. KEYS DOWN
:ARE ASSIGNED TO OUTPUT BUFFER
:LOCATIONS WHICH ARE STILL DE-
:ACTIVATED

00A2- A2 10       NKKY LDX 10         :NTABLE PNT/CNT
00A4- A0 09       LDY 09           :KTABLE PNT/CNT
00A6- A9 40       NK1  LDA 40         :PREPARE MASK
00A8- 25 CF       AND *NTBL,X       :NOTE TRIGGERED?
00AA- D0 0E       BNE NK3         :YES -GO TO NEXT
00AC- 88           NK2  DEY           :POINT NEXT KEY
00AD- F0 13       BEQ 00T         :NONE LEFT-LEAVE
00AF- B9 DF 00    LDA KTBL,Y         :KEY NEEDS HOME?
00B2- F0 F8       BEQ NK2         :NO -GET NEXT
00B4- 95 CF       STA *NTBL,X       :YES-PUT IN NOTE
00B6- C6 EB       DEC *OUTT         :ONE LESS OUTPUT
00B8- F0 08       BEQ 00T         :NONE LEFT-LEAVE
00BA- 24 E8       NK3  BIT *CTRL      :ALTERNATE MODE?
00BC- 10 01       BPL SKP2         :NO -DEC ONCE
00BE- CA         DEX                :YES-DEC THICE
00BF- CA         SKP2 DEX           :POINT NEXT NOTE
00C0- D0 E4       BNE NK1         :SOME LEFT -LOOP
00C2- 60           OUT  RTS             :RETURN

```

```

:
: TRGN
: TRANSIENT GENERATOR PROGRAM
:*****
:
CTRL DL 00E8
ATCK .DL 00EA
DCY .DL 00EB
SUST .DL 00EC
RLS .DL 00ED
PEAK .DL 00EE
NTBL .DL 00EF
PARM .DL 00F0
TTBL .DL 00F1
CWRD .DL 00F2

:
NTBL 00D0-00DF
TTBL 00C0-00CF

00C3- A5 E8       TRGN LDA *CTRL      :DO TRANSIENTS?
00C5- 10 38       BPL RTN1         :NO -RETURN
00C7- A2 10       LDX 10           :NTABLE PNT/CNT

:
:A/D/S/R DETERMINATION
:ROUTINE PREPARES Y TO USE AS
:CONTROL WORD, GETS NOTE AND
:SHIFTS TRIG. TO CARRY, GETS
:CURRENT STATE (CS) PARAMETER.
:IF NOTE TRIG. NOT SET STATE IS
:RELEASE. IF CS PARA IS POSI-
:TIVE STATE IS DECAY/SUSTAIN
:OTHERWISE, STATE IS ATTACK

00C9- A0 40       ADDR LDY 40         :PREPARE CWRD
00CB- B5 CF       LDA *NTBL,X       :GET NOTE AND
00CD- 2A           ROL                :ROTATE TRIGGER
00CE- 2A           ROL                :TO CARRY BIT
00CF- B5 CE       LDA *PARM,X       :GET CS PARA
00D1- 90 19       BCC RLS          :NO TRIG? -RLS
00D3- 10 0E       BPL D5           :CS>0? -DECAY/S

:
: ATTACK ROUTINE
:ADDS ATTACK PARAMETER TO CS PARA
:AND IF GREATER THAN PEAK
:SUBSTITUTES #3F AND SETS CONTROL
:WORD TO #40 (D6 SET - NO GLIDE).
:NOTE THAT CS PARA WILL BE >0
:WHEN NEXT CHECKED.

00D5- 18         ATTK CLC           :PREPARE
00D6- 65 BA       ADC *ATCK         :ADD ATTACK PARA
00D8- C9 BF       CMP 0BF           :>PEAK
00DA- 90 18       BCC NEXT         :NO -PLACE PARA
00DC- A5 BE       LDA *PEAK        :YES-PEAK VALUE
00DE- D0 17       BNE NEXT         :BRANCH ALWAYS

:
: DECAY AND SUSTAIN ROUTINE
:NOTE THAT CARRY IS SET. DECAY
:PARAMETER IS SUBTRACTED FROM
:CURRENT STATE PARAMETER. IF
:RESULT IS LESS THAN SUSTAIN
:PARAMETER THEN SUST. PARA.
:BECOMES CURRENT STATE PARA.
:D6 & D7 OF CONTROL WORD SET

00E0- A0 C0       D5  LDY C0         :PREPARE CWRD
00E2- E5 BB       SBC *DCY         :SUBTRACT DCY
00E4- C5 BC       CMP *SUST        :>SUSTAIN?
00E6- 10 0F       BPL NEXT         :PLACE PARA?
00E8- A5 BC       LDA *SUST        :CS PARA=SUST
00EA- 10 0E       BPL NEXT         :PLACE PARA

:
: RELEASE ROUTINE
:MAKES SURE THAT CURRENT STATE
:GLIDE BIT IS SET (NOTE-MAKES
:CS NEGATIVE). SUBTRACTS RELEASE
:PARA. FROM CURRENT STATE. IF
:RESULT >0, MAKES CS & CWRD =0

00EC- 38         RELS SEC          :PREPARE
00ED- 09 80       ORA 80           :SET CS GLIDE
00EF- E5 BD       SBC *RLS         :SUBTRACT RLS
00F1- 30 04       BMI NEXT        :CS<0 -PLACE CS
00F3- A0 00       LDY 00         :CS=0 -DONE MAKE
00F5- A9 80       LDA 80           :CS=80: CWRD=0

:
: NEXT
:PLACES CS PARA AND CWRD IN
:PROPER CONTROL CHANNEL OUTPUTS
:DECREMENTS POINTER (TWICE) AND
:IF NOT YET DONE LOOPS FOR MORE

00F7- 94 BE       NEXT STY *CWRD,X   :PLACE CONTROL
00F9- 95 CE       STA *PARM,X       :PLACE CS PARA
00FB- CA         DEX                :DECREMENT POINT
00FC- CA         DEX                :AND AGAIN
00FD- D0 CA       BNE ADDR        :SOME LEFT -LOOP
00FF- 60         RTN1 RTS           :RETURN

```

It has been pointed out that some perhaps pertinent details have been omitted from the preceding explanation of MUS 1.

The most prominent example is "why would you ever want to not have dynamic mode". The most probable reason is for special effects.

In general, the difference between special effects and noise is imagination. Contemporary musical lore is full of instances where a special effect resulted from an unsuccessful attempt to do something entirely different. Phil Spector's original "flanging" effect, so popular today, was supposed to be voice doubling, but didn't work.

In this same manner, there will be those who will be able to use the "step glissando" that results from too short a QuASH settling delay as a valid musical device.

Also, the dynamic mode requires an additional 16 byte table area that might easily be put to better use in some programs.

This same philosophy of maximizing versatility is responsible for the QuASH settling delay being an externally initialized variable. For the purpose of effect, there may be times when you want a short delay.

In addition to this, we have seen systems which were marginal in their power supply complement which would have a discernible pitch "blip" when keys were pressed with long delays (in the \$30 - \$40 range) - caused by the relatively heavy charging current producing a momentary dip in supply voltage. In these systems, a short term solution has been to decrease the QuASH driver delay to something on the order of \$10. The long term solution is more power.

SEVERAL POINTS RELATIVE TO THE OPERATION OF THE STGs SHOULD BE MENTIONED.

QuASH GLIDE CONTROLS. The setting of the QuASH glide pots have an effect on the transients produced. In most cases, these controls will need to be advanced only slightly from their fully counterclockwise "off" position.

The most noticeable effect of different settings of the glides will be observed when the STGs are set to the percussive mode by PEAK.

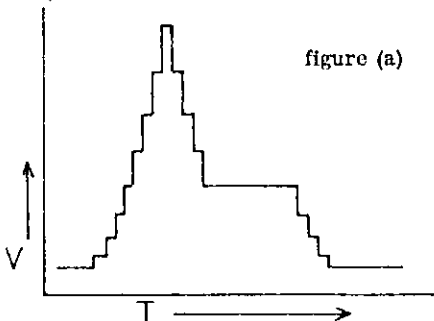
When the most significant bit of PEAK is cleared, it will effect only on the last increment of the attack cycle. For all increments other than the last, the glide will be set. A detailed example will best illustrate this.

Assume that we have set the STG parameters as follows:

- ATCK - 08
- DCY - 04
- SUST - 20
- RELS - 04
- PEAK - 3F

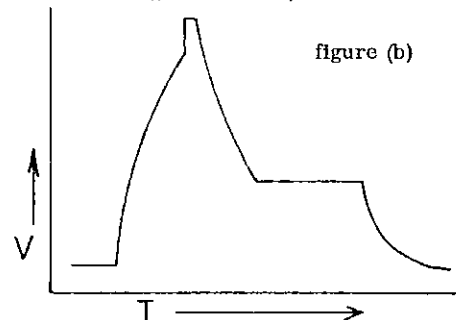
or in our more or less standard notation, \$08/\$04/\$20/\$04/\$3F.

If we were able to disable glide entirely, and then scope'd the transient we'd see this:



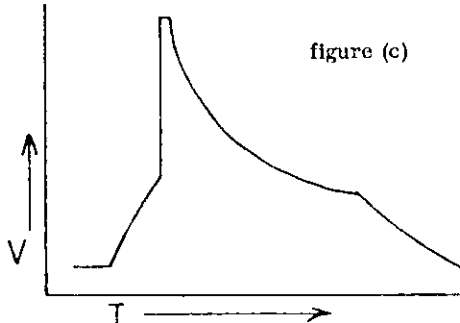
as the STG program counted up to the peak and then down to the sustain level before counting down to the base level when the key was released.

If we then enabled the glide and set them to a slightly advanced position and examined the same output we would find that this change had taken place.



The integrating action of the glide circuitry has smoothed the steps of the Attack, Decay and Release, with the exception of the last Attack step where (as we have already stated) the glide is off under all percussive circumstances. In this specific case, the last glide-less increment will be hardly noticeable.

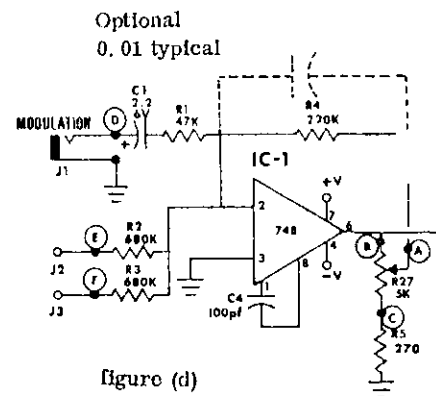
If, on the other hand, the glide is set to a long value (fully advanced, for instance) an examination of the waveform will show this:



The heavy glide has slowed the waveform to the point that when the glide-less final increment comes it takes a much greater step (one that is completely noticeable and unique). The heavy glide also has the effect of slowing the decay and release rates as shown.

It would also be appropriate to mention at this point that the instantaneous steps produced by the STG/QuASH combination is much faster than the maximum attack rate available from a 4740, or in fact from most ADSRs. Whereas a typical ADSR may have a minimum attack time of more or less 5 milliseconds, the QuASH in dynamic mode can step in a fraction of a millisecond.

This means that if there are any tendencies on the part of the VCA being used to have interaction between control and signal channels it will be aggravated when using STGs. We may hear "pops" and "thumps" that were not objectionable before. Probably the best solution here is to limit the response of the control voltage inputs of the VCA. In a 4710 Balanced Modulator, this means the addition of a small capacitor. Like this:



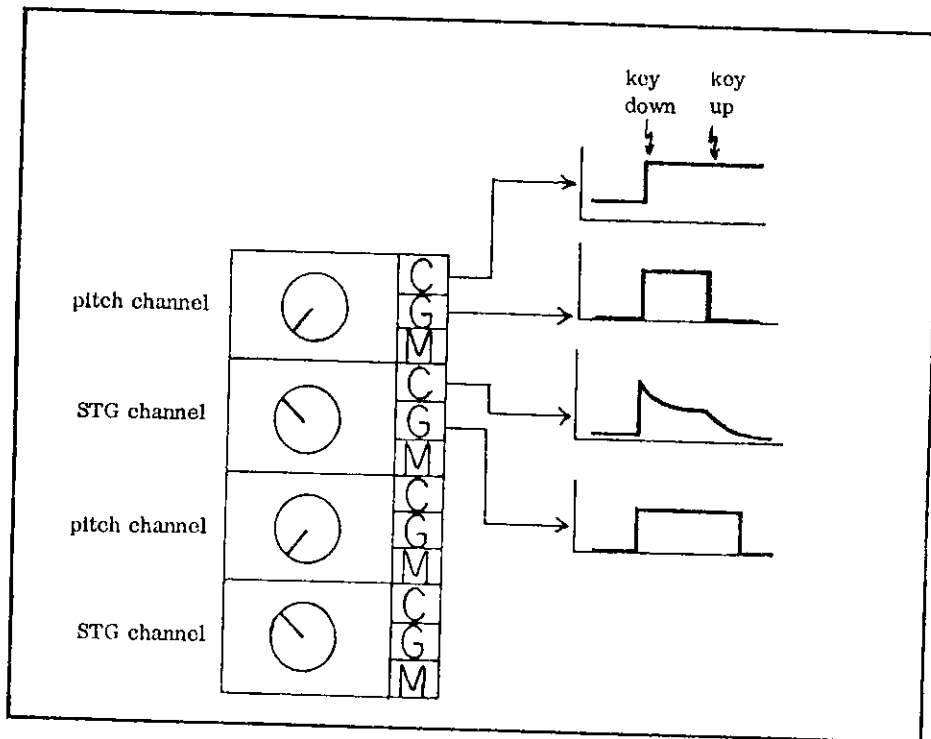
Another point to be considered is the fact that the output voltage from a QuASH channel doesn't go to zero - there may be some leakage from the VCA when it is supposed to be off. The easiest fix here is to re-adjust the Modulation rejection control of the VCA being used. In the 4710 this is R25. Be aware that this also limits somewhat the useable range of the D/A's TUNE control since wide variations in the setting of this control will affect the leakage from the VCA. Tuning changes on the order of 1/4 octave should not present any particular difficulties.

There is a point dealing with this which may not be immediately exploitable by many, but which should be mentioned in any case; the action of the trigger outputs of the QuASH channels which are being used as STG channels.

The trigger outputs QuASH channels which are being used to produce pitch setting voltages behave in the normal manner. When the AGO key

corresponding to that QuASH channel is being held down, the trigger is at a high state. When the key is released, the trigger goes low. A standard "gate" type response.

In a similar manner, the triggers of transient channels also go high as soon as the AGO key to which they correspond is pressed; but unlike the normal trigger, this level remains high until the software has completed the last increment of the Release cycle. In future hardware this will drive a "noise gate", a simple semi-conductor switch which completely quiets a channel that is inactive.





POLY 1.0

A 6500 series processor based realization of a limited resource allocation algorithm for Polyphonic control of electronic music synthesizers.

CONTENTS

- 1) Computer Requirements
- 2) The Algorithm
- 3) POLY 1.0 Documented Listing
- 4) Using POLY 1.0
- 5) Interconnection Wiring Schedules
- 6) Supplement "What the Computer Does"

COMPUTER REQUIREMENTS

POLY 1.0 is written and supplied in 6500 series processor machine language specifically for use with a PAIA 8700 Computer/Controller. It should be easily adaptable to any computer using this popular MPU and somewhat less readily adaptable to 6800 based machines.

Total memory requirements for POLY 1.0 and the buffer tables that it builds and maintains are less than 256 words. The program is written to reside fully in page zero of a 6500 based system.

POLY 1.0 uses the DECODE subroutine of the PAIA PIEBUG Monitor program to accept commands for special features.

This program also requires an appropriately encoded musical (AGO) keyboard up to 5 octaves in length which is assumed to be memory mapped at location x810. Also required is a contiguous block of output ports assumed to reside between locations x900 and x9FF. Wiring schedules consistent with these requirements are shown in the "Interconnection Wiring Schedules" section of this application note.

THE ALGORHYTHM

POLY 1.0 allocates synthesizer resources to keyboard requirements using this algorithm:

- 1) Output all notes appearing in the output buffer area (KTABLE) after adding the corresponding transposing figure from TTABLE. Go to 2.
- 2) Wait for keyboard scan to start and place a list of all keys currently being held down in the input buffer area (KTABLE). When buffer full or scan complete go to 3.
- 3) Clear the trigger flags (D6) of all notes in NTABLE (the output buffer).
- 4) Compare each entry in the input buffer to each entry in the output buffer. If they are the same, set the trigger bit of the output buffer entry and eliminate (zero) the entry from the input buffer. If all available outputs are used, or if all keys down find a home go to 1.
- 5) Place the remaining input buffer entries in output buffer locations which do not currently correspond to a key that is down (those in which D6 is cleared). When all input data has been placed or all channels available have been used go to 1.

A subtle implication of this algorithm is that if we think of "new" notes as being those corresponding to keys that have just been pressed, this method tries to place those new notes in output channels which at some point in the past were already producing those notes.

This prevents strings of identical eighth notes (for example) from being assigned to different outputs each time they're used. Notes, once assigned, tend to stay assigned to the same control channel regardless of other keyboard activity. They don't move around from channel to channel in some totally unpredictable fashion.

It also means that once the number of output channels available is "used up" by down keys which need to be placed, all other keys that are down are simply ignored (this is exactly what we want).

One important aspect of the above is that the program must "know" how many output channels are available to it; otherwise there is the possibility that notes may be assigned to non-existent outputs (ones that do not have corresponding hardware). By itself, this is not disastrous as we are faced with a similar situation anyway - POLY 1.0 ignores notes that exceed its resources.

The really bad part would be that once the notes were assigned to these non-channels they would tend to remain assigned there; producing the effect of having some "dead" synthesizer keys that appear to be doing nothing.

POLY 1.0

A Limited Resource Allocation
Algorithm

DOCUMENTED LISTING

MEMORY MAP

| | |
|-----|-------------------|
| 9FF | |
| ... | Control Channels |
| 900 | |
| ... | |
| 810 | AGO KBD |
| ... | |
| 0FF | |
| ... | used by PIEBUG |
| 0ED | |
| 0EC | not used |
| 0EB | OUTTEMP |
| 0EA | OUTS |
| 0E9 | TEMP 1 |
| 0E8 | CLK |
| 0E7 | |
| ... | TTABLE |
| 0E0 | (transpose) |
| 0DF | |
| ... | NTABLE |
| 0D8 | (output buffer) |
| 0D7 | |
| ... | KTABLE |
| 0D0 | (input buffer) |
| 0CF | |
| ... | not used |
| 0CB | |
| 0CA | |
| ... | POLY 1.0 |
| 006 | |
| 005 | |
| ... | Interrupt Vectors |
| 000 | (if required) |

POLY 1.0

written by John S. Simonton, Jr.
(c) 1978 by PAIA Electronics, Inc.
All rights reserved.

INIT clears input buffer KTABLE, output
buffer NTABLE and transpose buffer
TTABLE.

```
00 06 INIT      A9 00    LDA #0          ; prepare
08 08 INIT 1    A2 18    LDX #18
0A 0A INIT 0    95 CF    STA KTABLE-1,X  ; clear everything
0C 0C           CA      DEX
0D 0D           D0 FB    BNE INIT 0      ; loop until done
```

NOTEOUT adds together corresponding
entries in TTABLE and NTABLE and
reads the result into the control channels
in descending order. Also takes care
of timing to D/A.

```
0F NOTE OUT    A2 08    LDX #08        ; initialize pointer
11 LOOP        B5 D7    LDA NTABLE,X   ; get note
13             18      CLC                          ; prepare
14             75 DF    ADC TTABLE,X               ; add transpose
16             8D 00 09 STA D/A          ; write to D/A, settle
19             9D F7 09 STA S/H,X          ; write control channel
1C             A0 04    LDY #4                      ; prepare delay time
1E LOOP 1      88      DEY                          ; delay
1F             D0 FD    BNE LOOP 1
21             CA      DEX                          ; done?
22             D0 ED    BNE LOOP                    ; no, continue
```

LOOK clears input buffer and waits for
keyboard scan to begin. Places keys down
in the input buffer. This routine also
serves to make the keyboard's encoder
clock the tempo clock for the entire
system.

| | | | | |
|----|------|----------|---------------|-----------------------------|
| 24 | LOOK | A2 08 | LDX #8 | ; set pointer/counter |
| 26 | | A9 00 | LDA #0 | ; prepare accum. |
| 28 | LK0 | 95 CF | STA KTABLE, X | ; zero input buffer |
| 2A | | CA | DEX | ; point to next |
| 2B | | D0 FB | BNE LK0 | ; done? no-loop |
| 2D | | A2 08 | LDX #8 | ; yes set pointer |
| 2F | LK1 | 2C 10 08 | BIT KBD | ; scan started? |
| 32 | | 30 FB | BMI LK1 | ; no D7 high, loop |
| 34 | LK2 | 2C 10 08 | BIT KBD | ; yes-look again |
| 37 | | 30 0F | BMI LOUT | ; when scan done, leave |
| 39 | | 50 F9 | BVC LK2 | ; if key not down, loop |
| 3B | | AD 10 08 | LDA KBD | ; get the key |
| 3E | | 95 CF | STA KTABLE, X | ; put it in input buffer |
| 40 | LK3 | CD 10 08 | CMP KBD | ; still same key? |
| 43 | | F0 FB | BEQ LK3 | ; yes, loop |
| 45 | | CA | DEX | ; advance pointer |
| 46 | | DO EC | BNE LK2 | ; if some buffer left, loop |
| 48 | LOUT | E6 E8 | INC CLK | ; increment clock |
| 4A | | A5 E8 | LDA CLK | ; get it |
| 4C | | 8D 20 08 | STA DISP | ; show it |
| 4F | | EA | NOP | ; hook |
| 50 | | EA | NOP | |
| 51 | | EA | NOP | |

POLY first half of allocation algorithm as explained in text. In this block de-activated channels are re-activated if the data they contain appears in the input buffer.

| | | | | |
|----|--------|-------|---------------|-----------------------------|
| 52 | POLY | A5 EA | LDA OUTS | ; number of output |
| 54 | | 85 EB | STA OUTTEMP | ; channels to counter |
| 56 | | A2 08 | LDX #8 | ; set up a counter |
| 58 | LOOP 0 | A9 BF | LDA #BF | ; prepare for following: |
| 5A | | 35 D7 | AND NTABLE, X | ; clear trigger bit of note |
| 5C | | 95 D7 | STA NTABLE, X | ; and put it back |
| 5E | | CA | DEX | ; pointer to next note |
| 5F | | D0 F7 | BNE LOOP 0 | ; if not yet done, loop |
| 61 | | A9 09 | LDA #9 | ; prepare |
| 63 | | 85 E9 | STA TEMP 1 | ; set up pointer to |
| | | | | ; KTABLE |
| 65 | LOOP 1 | C6 E9 | DEC TEMP 1 | ; decrement the pointer |
| 67 | | F0 23 | BEQ NEWKEY | ; if done go to newkey |
| 69 | | A6 E9 | LDX TEMP 1 | ; prepare x pointer |
| 6B | | B4 CF | LDY KTABLE, X | ; get key in Y register |
| 6D | | F0 1D | BEQ NEWKEY | ; no keys left, place new |
| | | | | ; keys |

| | | | | |
|----|--------|-------|---------------|--|
| 6F | | A2 09 | LDX #9 | ; initialize pointer to ; NTABLE |
| 71 | LOOP 2 | CA | DEX | ; decrement pointer |
| 72 | | F0 F1 | BEQ LOOP 1 | ; if all NTABLE done, ; get new key |
| 74 | | 98 | TYA | ; copy of next key to ac. |
| 75 | | 55 D7 | EOR NTABLE, X | ; compare to NTABLE |
| 77 | | 0A | ASL | ; and shift high order |
| 78 | | 0A | ASL | ; bits out to ignore |
| 79 | | D0 F6 | BNE LOOP 2 | ; if diff., get next NTABLE |
| 7B | | 98 | TYA | ; if same, key to ac. |
| 7C | | 15 D7 | ORA NTABLE, X | ; and preserve glide |
| 7E | | 95 D7 | STA NTABLE, X | ; and save note |
| 80 | | C6 EB | DEC OUTEMP | ; one less output available |
| 82 | | F0 31 | BEQ OUT | ; if none remain, leave |
| 84 | | A6 E9 | LDX TEMP 1 | |
| 86 | | A9 00 | LDA #0 | ; prepare accumulator |
| 88 | | 95 CF | STA KTABLE, X | ; and then zero this key |
| 8A | | F0 D9 | BEQ LOOP 1 | ; then branch always ; for more |

NEWKEY second half of allocation algorithm.
 Keys down are allocated to output buffer
 locations which are currently de-activated.
 NOTE that both this routine and POLY
 preserve the status of D7 in the output
 buffer locations.

| | | | | |
|----|--------|-------|---------------|--|
| 8C | NEWKEY | A9 00 | LDA #0 | ; prepare accum. |
| 8E | | A2 09 | LDX #9 | ; prepare pointer |
| 90 | NK3 | CA | DEX | ; to KTABLE, decrement |
| 91 | | F0 22 | BEQ OUT | ; if done, leave |
| 93 | | B4 CF | LDY KTABLE, X | ; key to Y reg. |
| 95 | | F0 F9 | BEQ NK3 | ; if key zero, get next key |
| 97 | | 95 CF | STA KTABLE, X | ; a key that needs a ; home ; zero the key in KTABLE |
| 99 | | A2 09 | LDX #9 | ; prepare pointer |
| 9B | NK4 | CA | DEX | ; decrement |
| 9C | | F0 17 | BEQ OUT | ; if zero, no NTABLE ; left |
| 9E | | A9 40 | LDA #40 | ; otherwise prepare a ; mask |
| A0 | | 35 D7 | AND NTABLE, X | ; and check for free ; NTABLE entry (D ₆ 0) |
| A2 | | D0 F7 | BNE NK4 | ; not this one |
| A4 | | A9 80 | LDA #80 | ; yes-now prepare a mask |
| A6 | | 35 D7 | AND NTABLE, X | ; D7 set on clear |
| A8 | | 95 D7 | STA NTABLE, X | ; put back in NTABLE |

| | | | |
|----|-------|---------------|--|
| AA | 98 | TYA | ; copy of key to accum. |
| AB | 15 D7 | ORA NTABLE, X | ; set/clear D ₇ |
| AD | 95 D7 | STA NTABLE, X | ; and back to NTABLE |
| AF | C6 EB | DEC OUTTEMP | ; one less output avail- ; able |
| B1 | F0 02 | BEQ OUT | ; if none remaining, out. |
| B3 | D0 D7 | BNE NEWKEY | ; otherwise, branch always ; for more |

OPTION uses PIEBUG's DECODE sub-routine to read computer keyboard and take appropriate action as required.

| | | | | |
|----|--------|----------|--------------|-------------------|
| B5 | OPTION | 20 00 FF | JSR DECODE | ; get command |
| B8 | | C9 04 | CMP #4 | ; greater than 4? |
| BA | | B0 03 | BCS OP1 | ; yes test next |
| BC | | 4C 06 00 | JMP INTT | ; no-clear all |
| BF | OP1 | C9 08 | CMP #8 | ; greater than 8? |
| C1 | | B0 05 | BCS OP2 | ; yes test next |
| C3 | | A9 2E | LDA #2E | ; no prepare |
| C5 | | 4C 08 00 | JUMP INIT 1 | ; go to tune |
| C8 | OP2 | 4C 0F 00 | JUMP NOTEOUT | ; run full poly |

ADDITIONAL PROGRAM NOTES

- 1) Memory location OEA contains the number of control channels available to the system. This number may be changed as situations require, but should under no conditions be set to a number greater than the number of output channels.
- 2) Note the three NOPs at locations 04F-051. This hook may be used to insert special effects code in place of, or prior to, POLY. May also be used to turn the combination of NOTEOUT and LOOK into a subroutine to be accessed by other users programs.
- 3) All code (with the exception of OPTION) is written in a relocatable form and may be positioned in available memory as required.

LOADING AND INITIALIZING POLY 1.0

If you have a cassette interface on your 8700 and the POLY 1.0 tape, loading is simply a matter of connecting your tape recorder to the cassette input connectors on the 8700 and loading the tape using the following entry sequence:

```
0-0-0-0-0-0-F-F-0-0-1-1-TAPE
```

If you don't have the CS-87 option, you must enter the code manually from the 8700 keyboard.

The cassette version of this program loads all of page zero of memory (its total requirement) and in the process initializes a couple of things that you will need to care for manually if the cassette is not available. When entering manually, be sure to set the number of outputs to correspond to the number you have available. For example, assuming that you have a system with a single QuASH, the number of channels available should be set to 4 using the following computer keyboard sequence:

```
RESET-0-0-E-A-DISP
```

```
0-4-ENTER
```

The tape version initializes the number of outputs at the most likely number of 4. If you want to use less channels (because of lack of modules, say) or have a system with more, do it as was shown above.

When entering the program manually, make sure that the decimal mode flag in the status register is cleared by using this sequence:

```
RESET-0-0-F-F-DISP
```

```
0-0-ENTER
```

This is automatically taken care of when the tape version is loaded.

USING POLY 1.0

With everything connected, loaded and initialized, we're ready to begin making music. Go to the beginning of the program and begin running it.

```
RESET-0-0-0-6-RUN
```

If everything is working properly, we will see the 8700 displays counting quickly, incrementing by one for each scan of the keyboard. All of the QuASH outputs should be at a very low output voltage (the program initializes them as zero) and the trigger flags for each channel should be cleared.

As we press synthesizer keys, QuASH channels should "come alive" and produce control voltages corresponding to the keys that POLY 1.0 has assigned to them. The trigger flags should be set if the key corresponding to the channel is currently down and clear when the key is released.

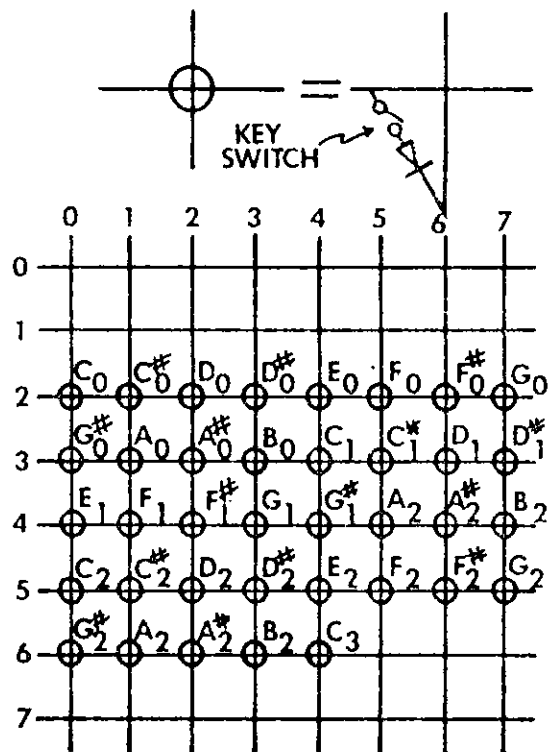
TWO FEATURES OF POLY 1.0 HAVE NOT YET BEEN MENTIONED

While POLY is running, touching any of the keys from 0-3 on the 8700 keyboard (the first row of keys) causes the system to clear all QuASH channels to zero and wait for new data to be assigned. You'll figure out what this is good for as you become familiar with the system.

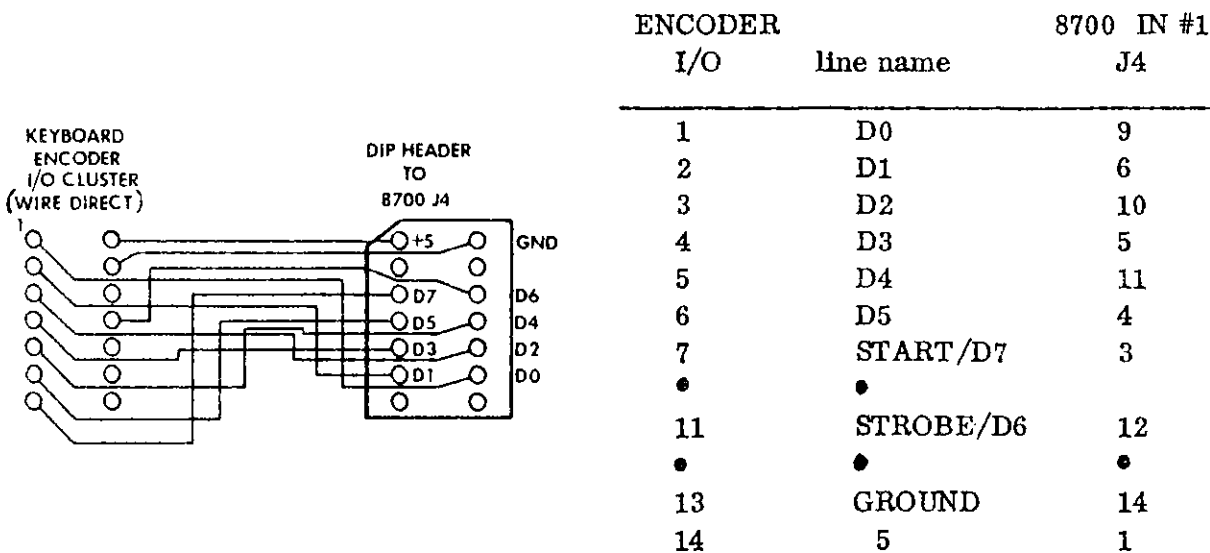
Maybe more importantly, touching any of the keys 4-7 (the second row on the 8700) provides a tuning function and causes all QuASH channels to produce the same note with the trigger flags set, allowing all oscillators to be set to the same pitch. The note produced corresponds to the 2nd C on a standard configuration 3 octave keyboard. **THE CHANNELS MUST BE CLEARED AFTER TUNING** by touching the first row of 8700 keys.

INTERCONNECTION WIRING SCHEDULES

KEYBOARD TO COMPUTER

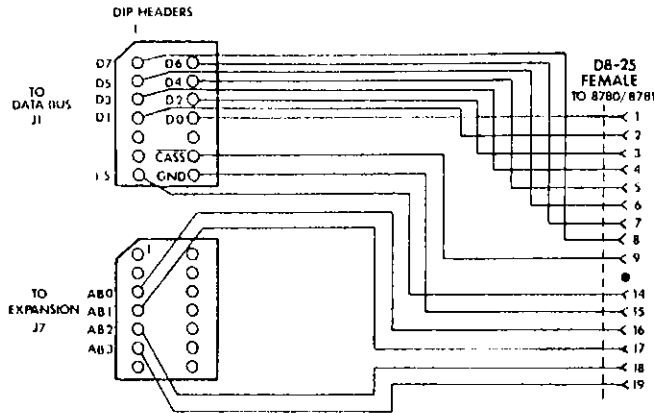


ENCODER TO KEYBOARD

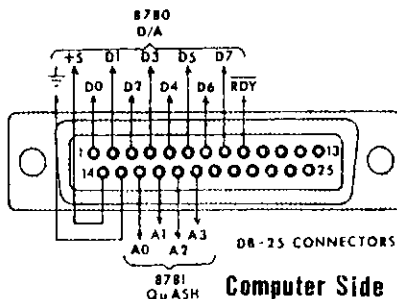
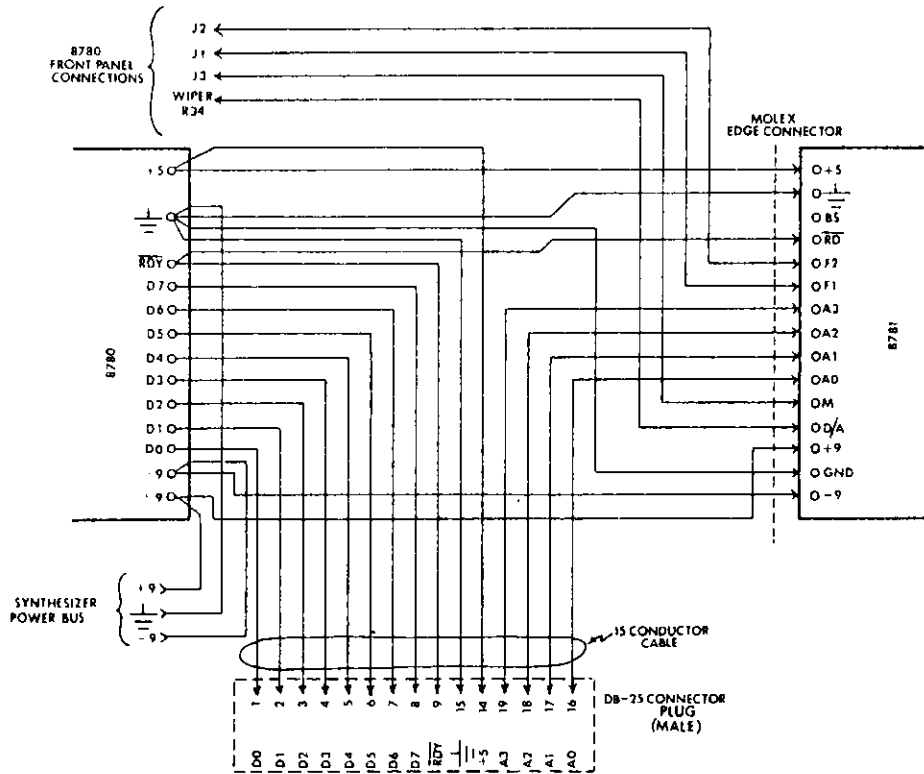


ENCODER TO COMPUTER

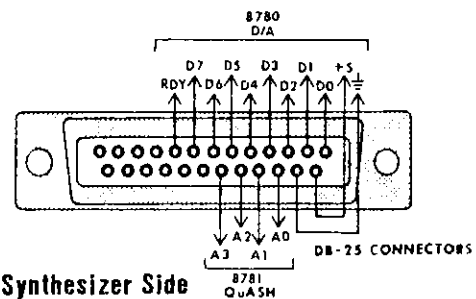
COMPUTER TO SYNTHESIZER



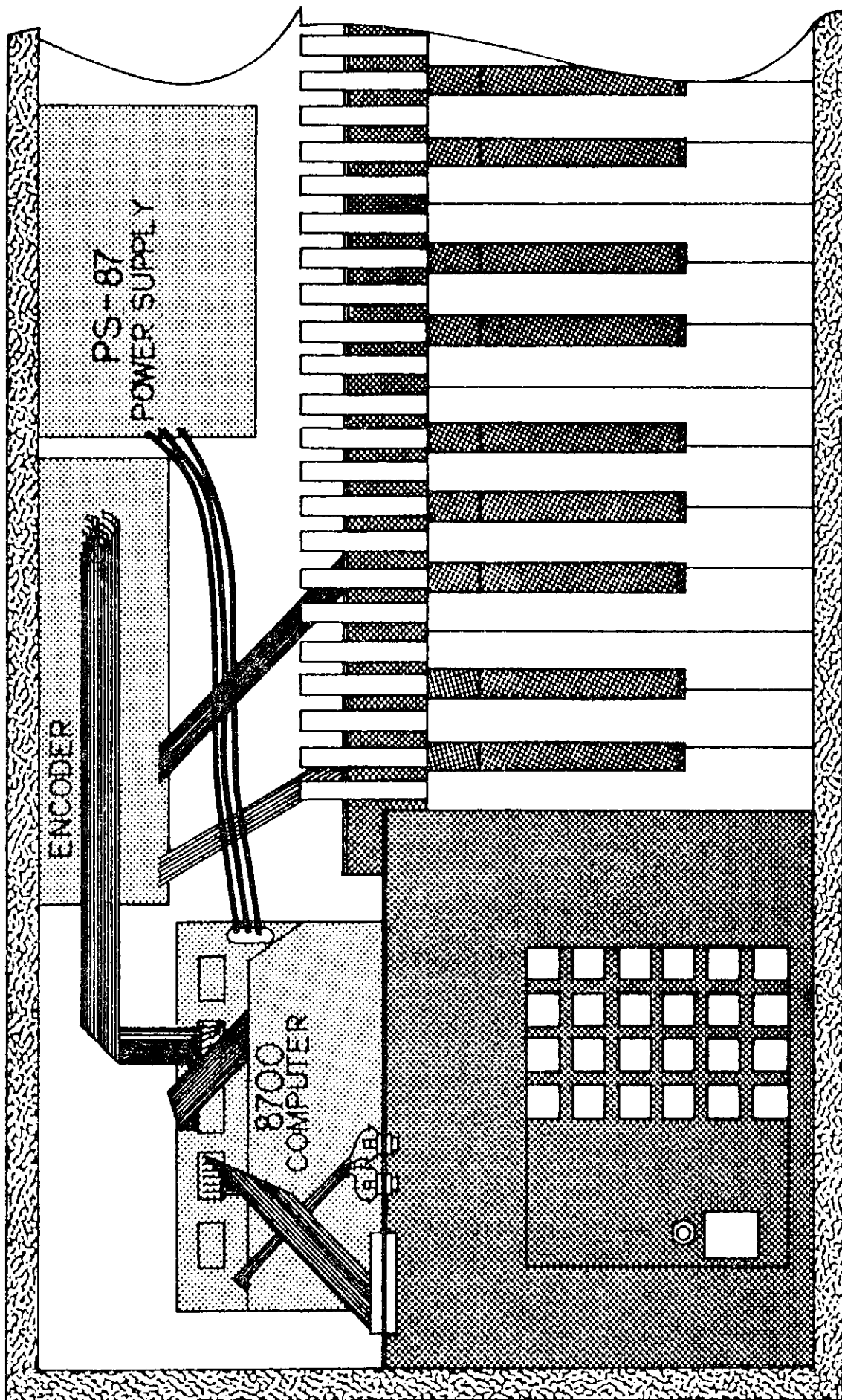
| COMPUTER DIP HEADERS Con # | Pin # | line name | 8780/8781 D Connector Pin # |
|----------------------------------|-------|-----------|-----------------------------------|
| J1 | 1 | D7 | 8 |
| | 2 | D5 | 6 |
| | 3 | D3 | 4 |
| | 4 | D1 | 2 |
| | 7 | 5 | 14 |
| | 8 | Gnd | 15 |
| | 9 | CASS/RDY | 9 |
| | • | • | • |
| | 11 | D0 | 1 |
| | 12 | D2 | 3 |
| | 13 | D4 | 5 |
| | 14 | D6 | 7 |
| J7 | 3 | A0 | 16 |
| | 4 | A1 | 17 |
| | 5 | A2 | 18 |
| | 6 | A3 | 19 |



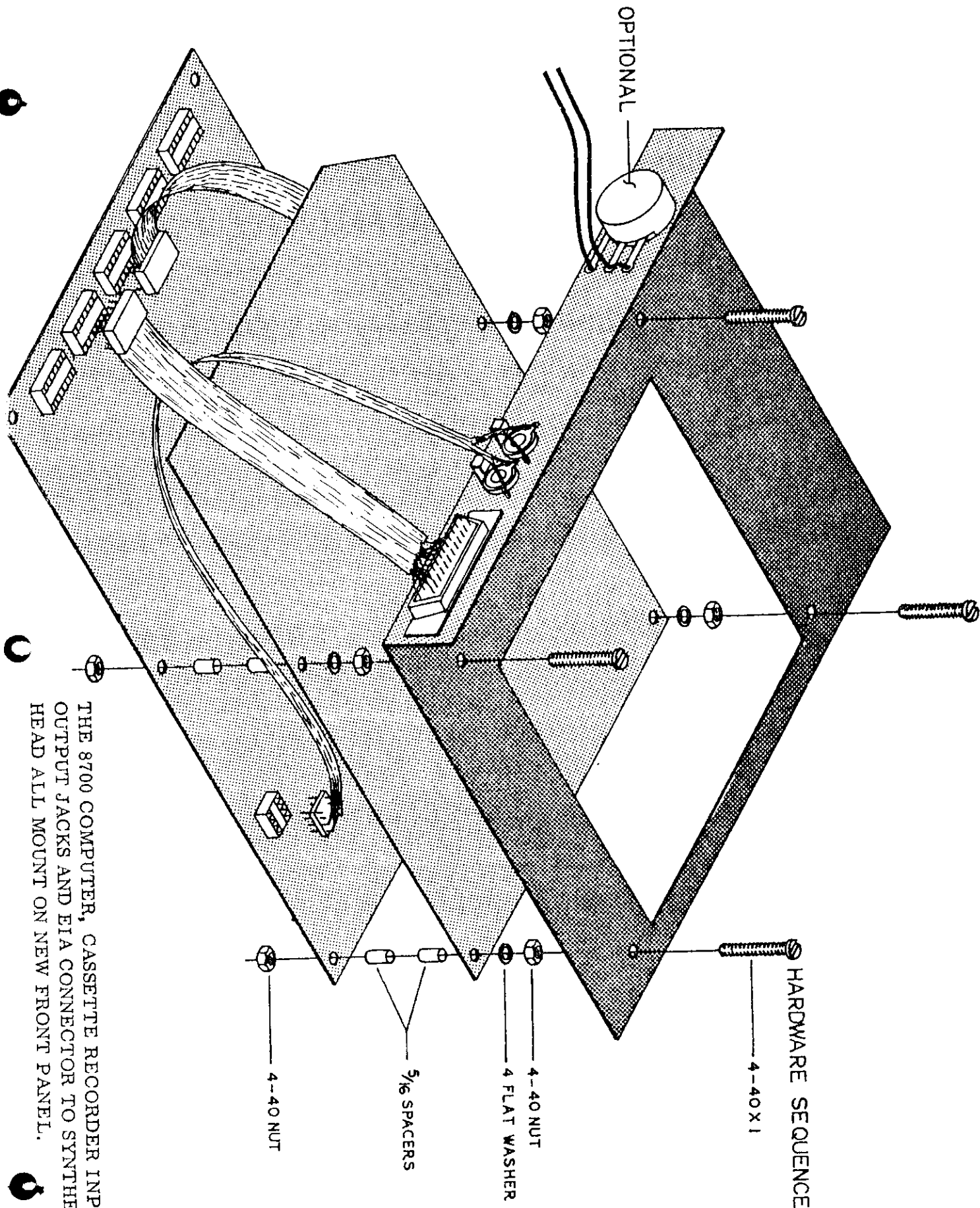
Computer Side



Synthesizer Side



PAIA 8700 COMPUTER, POWER SUPPLY AND KEYBOARD ENCODER
RETRO-FIT TO ANY 4700 OR 8700 SERIES KEYBOARD.



HARDWARE SEQUENCE

4-40 X 1

4-40 NUT

4 FLAT WASHER

5/16 SPACERS

4-40 NUT

OPTIONAL

THE 8700 COMPUTER, CASSETTE RECORDER INPUT & OUTPUT JACKS AND EIA CONNECTOR TO SYNTHESIZER HEAD ALL MOUNT ON NEW FRONT PANEL.