POLY 1.0

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

CONTENTS

1020 W. Wilshire Blvd., Oklahoma City, OK 73116

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 works. 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 algorhythm:

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 algorhythm 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
Algorhythm

DOCUMENTED LISTING

MEMORY MAP

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

POLY 1.0

written by John S. Simonton, Jr.

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

```
00 06  INIT       A9 00     LDA #0                ; prepare
   08  INIT 1     A2 18     LDX #18
   0A  INIT 0     95 CF     STA KTABLE-1,X        ; clear everything
   0C             CA        DEX
   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 algorhythm 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 algorhythm.
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 (D6  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 D7
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 subroutine 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 INIT      ; 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 combinatio 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 INITILIZING 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.

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.

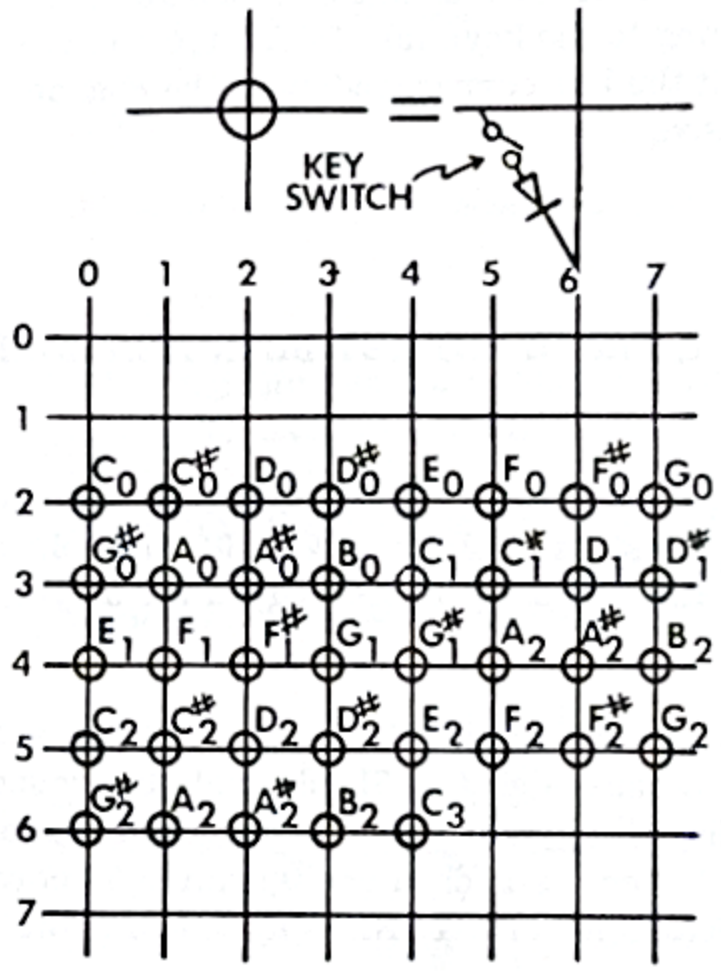
TWO FEATURES OF POLY 1.0 HAVE NOT YET BEEN MENTIONED

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.

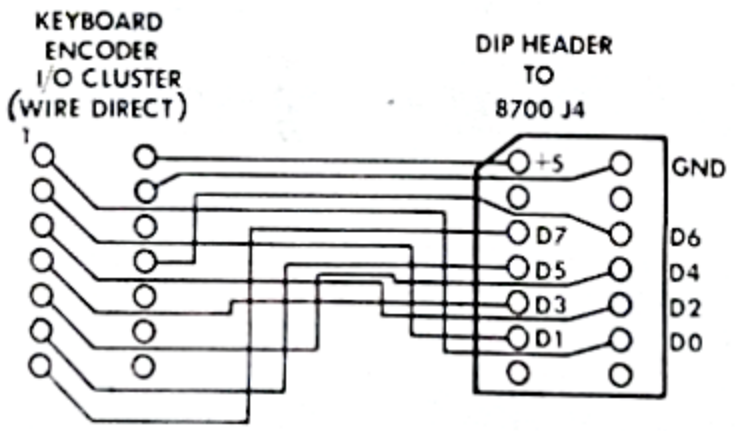
Maybe more importantly, touching any of the keys 4-7 (the second row on the 8700) provides a tuning function and causes all QuASH channels to produce the same note with the trigger flags set, allowing all oscillators to be set to the same pitch. The note produced corresponds to the 2nd C on a standard 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



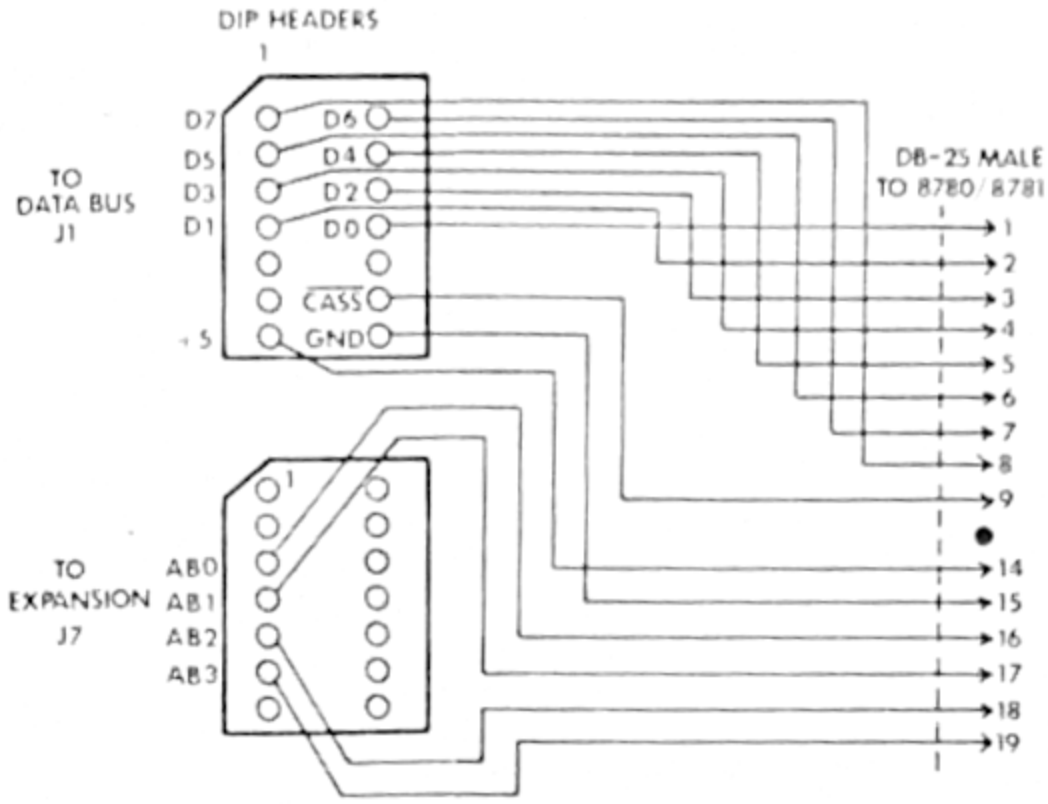
ENCODER TO KEYBOARD



ENCODER I/O	line name	8700 IN #1 J4
1	D0	9
2	D1	6
3	D2	10
4	D3	5
5	D4	11
6	D5	4
7	START/D6	12
•	•	•
11	STROBE/D7	3
•	•	•
13	GROUND	14
14	5	1

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	$\overline{\text{CASS}}/\overline{\text{RDY}}$	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