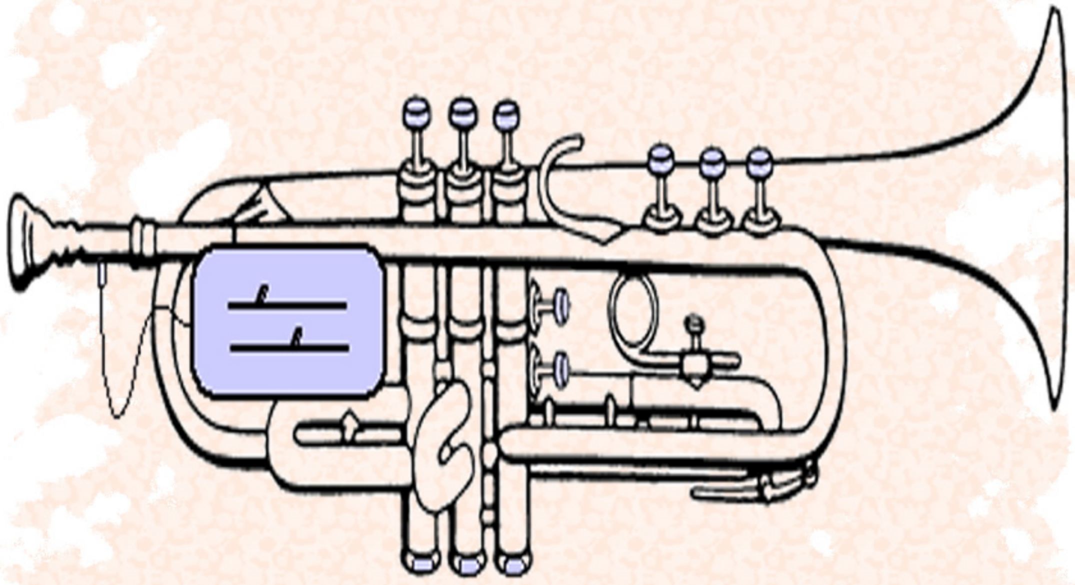
MIDI Trumpet



MIDI Trumpet

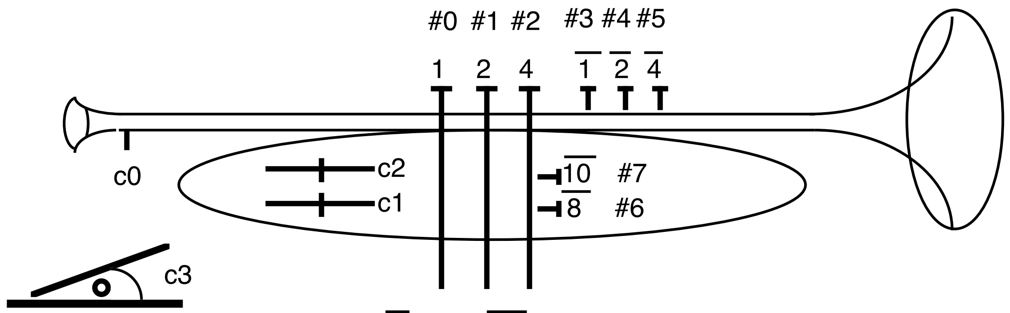
John Talbert 3/1992

The MIDI Trumpet was built in response to a composer asking to retrofit his old trumpet with MIDI capabilities. The design was based on the already developed MIDI Horn which is a microcomputer based wind controller. In this case, however, we wanted to use the actual sound of the trumpet to generate the Note Velocity data instead of using a pressure sensor.

This was accomplished by picking up the trumpet sound with a Barcus Barry contact mic whose output was fed to a Solid State Music 2011 preamp chip. The preamp output was rectified by a voltage follower circuit which provided a control voltage that follows the amplitude of the trumpet sound. The straight preamp output was also made available for amplification or other processing.

MIDI Key data from the trumpet signal might have been obtained by pitch detection circuitry; however, circuitry for pitch detection is quite involved and often slow. Instead, I opted for switches mounted at the bottom of the 3 trumpet fingering valves, and 5 more pushbuttons mounted within easy reach of the player's fingers. For continuous controllers, two slide pots were mounted on the side of the trumpet plus a jack for a foot pedal .

Like the MIDI Horn, the heart of the instrument is a single-board microcomputer - the SBC88 from Vesta Technology. This is a Forth Language based microcomputer. Unlike the MIDI Horn, it is readily programmable by the user. It includes a serial interface for connecting to a PC. User programs can be typed in from any terminal emulation program and loaded into EEPROM memory using some simple Forth Editor programs. Only a few subroutines need to be entered to produce MIDI signals from the controller voltages. These Forth routines are described in detail later. These simple subroutines are readily expandable by the user/programmer in any number of ways.



1, 2, 4, $\overline{8}$, and $\overline{10}$ are read from10 PC@.

$\overline{1}$, $\overline{2}$, and $\overline{4}$ are read from 20 PC@.

Trumpet Software Description

Forth

The Midi Trumpet was designed to be easily programmable. It uses the programming language Forth, contained on an 8K ROM chip within the computer. The SBC-Forth Programming Manual will tell you what exactly is included in your Forth system.

Basically, Forth is a collection of functions called a dictionary. Use the command VLIST to list out all the "words" in your system's dictionary. Your job as a programmer is to add your own "words" to this dictionary by combining previously created words. The "HORN" program was done completely in Forth.

Program Storage

The micro has a special kind of memory chip called an EEPROM (Electrically Erasable Rom Memory). When you write into this chip, the data stored there will remain even when the computer is powered down. The Load/Hold switch is connected directly to this EEPROM. In order to load data or programs onto the chip, the switch must be up in the LOAD position. Normally, however, you will want to safeguard your stored programs by keeping the switch in its down or HOLD position, especially when powering up.

This EEPROM chip is located at addresses A000 to BFFF (hexidecimal). To the Forth System, this is the location of eight Editor screens - specifically screens numbered 3 through 10. (Screens 1 and 2 are in Ram memory.) I have loaded 7 of these screens with words for the HORN program. You can read them using the command EDITOR 3 LIST, 4 LIST, etc. A word of **WARNING** -- the HOLD switch must be down when the power is turned on or the RESET button pushed; otherwise, Screens 3 through 10 will be erased when the system starts up.

You can also use the EEPROM space to store data - just make sure the Forth Editor commands stay away from it (for example, don't do a 10 CLEAR command if you know that you have data previously stored at that screen location). See the Memory Map for the exact locations of screen memory, dictionary, etc.

Autostart

The computer has an autostart feature. On startup, the system looks at the very first character of Screen 3. If that character is a colon (i.e. the start of a word definition) then the startup program will begin loading Screens. All colon definitions of new words on the Screen will be compiled and added to the dictionary, and any word commands will be executed. The HORN program is loaded in this way on startup.

The word " -->" is used to tell the system to continue loading the next screen. The word ".S" tells it to stop loading.

Be patient on startup. It takes about 12 seconds to load all 9 Horn screens. When the Horn program is finally executed you will see the words " Midi Trumpet " on the screen if you are hooked up to a terminal. Press any key to exit the program and get back into the Forth system. Of course, with the autostart feature, the terminal is not needed to start the HORN program.

Talking to the Computer

The computer has a serial port labeled "Macintosh" plus a cable for connecting it to one of the two serial ports on your Macintosh. To talk to the Micro through the serial port you will need some kind of terminal emulation program - the programs used when communicating with modems. I am using White Knight 11.12, an updated version of the shareware program Red Ryder.

Set up your terminal program for a baud rate of 4800, no parity, 8 databits, 1 stopbit, full duplex, and VT100 terminal emulation. (For resetting the baud rate hit the space bar on reset.)

The computer is set up to autostart the Horn Program. After the horn program has begun, hit any terminal key to exit the Horn program and get into the Forth system. The Forth system will announce itself. Hit return a couple times until you get an "OK". You are then ready to explore in Forth. If at anytime the cursor freezes up you will have to push the Reset button and start the whole process over again, losing any words you have loaded.

Loading from the Macintosh

You may find it useful to store your programs on the Mac, loading them into the micro only when you need them. This involves two requirements. First, your terminal emulation program must be able to send text files with a delay after each character of about 1 second. The White Knight does it like so: use the SEND TEXT FILE entry under the FILE menu and set the Delay setting under the Customize/Options/Text Transfer menu. Second, you need a way to make simple text files stripped of all the formatting data most word processors stick onto a file. Microsoft Word does this by allowing you to choose a Text Only File Format when saving a file.

The easiest way to load Forth definitions from the Mac is in the form of an Editor Screen. First set your Forth system to the Editor vocabulary:

```
EDITOR   DECIMAL   1 CLEAR
```

This sets up the Editor set of words, specifies decimal numbers for the screen line numbers, and directs the Editor's attention to screen #1 by clearing it (you may use any of the ten screen numbers here). At this point, when typing by hand, you would normally use the Editor "P" command which uses the format: line#   P   stuff you want on this line#   Return
get an OK. You would do this 15 times to completely fill up a screen. The Macintosh can automatically type these commands for you as long as they are in the same "P" command format and don't come at the micro too fast. Just choose the SEND TEXT FILE from your Mac's

terminal emulation program, choose your previously stored TEXT ONLY file when the Find File dialog box comes up, sit back and watch. When it is finished type 1 LIST to make sure it is all there. If you are loading EEPROM screens (3 thru 10) you will need to flip the Load/Hold switch to Load and then execute the word FLUSH. It will take a few seconds. Be sure and switch back to HOLD after the micro returns an OK. (The reason for FLUSH is that the Editor works on screens in a buffer area first, not the final location of the screen. The word Flush will transfer the data in the working buffer to the actual screen memory location.) Once your screens are filled, use the LOAD command to compile and/or execute the screens.

<u>The HORN Program</u>

The 8-screen listing of this program is my attempt at a workable program for the Midi Trumpet. Please feel free to modify or cannibalize this program for your own purposes. When looking through this program keep in mind that the hardware is basically two simple devices: a MIDI command generator (the micro) and a director (the trumpet, using 8 switches and 4 continuous controllers). What I wrote was a general performance program. More specific performance piece programs also might be interesting.

Revising stored EEPROM screens is an easy matter. First, type - EDITOR screen# LIST. Thenb revise the screen using the Editor vocabulary (I only bothered learning the P command of the Editor). Use the LIST command to check on your revisions. When ready, throw the Load switch and execute the FLUSH command. Be sure to set the switch back to HOLD.

What follows is a short description of the Horn program as stored on Screens 3 through 9. Refer to the program listing when reading this description.

Screen 3

```
0 P  : TABLE <BUILDS 0 DO C, LOOP DOES> + ;   DECIMAL
1 P  0 0 0 0  4 TABLE CV  ( newly read controller values )
2 P  0 0 0 0  4 TABLE MCV ( midi'd controller values )
3 P  4 7 1 2 4 TABLE CTL ( midi controller #'s used)
4 P  ( breath, modulation, volume, foot controller )
5 P  78 81 79 77 80 83 82 84  73 76 74 74 72 75 72 72
6 P  66 69 71 70 68 71 67 67  61 64 62 65 63 59 60 60
7 P  32 TABLE NOTE ( see Trumpet Fingering Charts )
8 P  0 VARIABLE PRG   ( current midi prog number )
9 P  0 VARIABLE KEYS  ( newly read pushbuttons )
10 P  0 VARIABLE MKEYS  ( midi'd pushbuttons )
11 P  0 VARIABLE PITCH ( calculated midi key value )
12 P  0 VARIABLE CHNL  ( midi channel )
13 P  0 VARIABLE ?ON   ( note on = 1, note off = 0 )
14 P  0 VARIABLE ?CTL  ( true if midi control was sent last )
15 P -->
```

TABLE is a construct which sets aside locations in RAM memory for storing variables. For example, the CTL table is created for storing the midi controller numbers to be associated with the Midi Trumpet's 4 continuous controllers.

```
4 7 1 2       4 TABLE CTL
```

To create the table you first specify the values you want to initially store in the table (4, 7, 1, 2), then you tell it how big the table is (4), then the word TABLE , and finally a name to call the table (CTL). After creating the table you can use it in the following way:

2 CTL C@	Reads the second slot of table CTL, its value (1) is placed on the stack.
12 2 CTL C!	Loads the value 12 into the second slot of table CTL.

According to this table, the Trumpet breath will control midi breath (midi controller #2), slider 1 will control modulation (#1), slider 2 will control volume (#7), and the foot pedal will control the midi foot controller (#4). As you can see, these values are easily changed. Look up midi controller numbers in the midi specifications.

The 32 NOTE table values set up what midi pitch will be played using info from the 3 valve switches and the 2 side switches. There was no easy way to detect harmonics from the trumpet pickup, so we need an extra couple switches to let the computer know what harmonic to play. This table of values is a wild guess on my part. Take a look at the sheet explaining how these values are derived and change them as you like.

The remaining words give names and storage locations for several variables used later.

Screen 4

```
0 P DECIMAL
1 P  0 VARIABLE VEL   ( midi key velocity )
2 P  0 VARIABLE CSUS   ( current sustain value )
3 P  0 VARIABLE COCT   ( current octave offset )
4 P  : SETOCT CASE     ( set oct with side-mount key value)
5 P         24 OF 0 COCT C! ENDOF
6 P         16 OF 24 MINUS COCT C! ENDOF
7 P         8  OF 12 MINUS COCT C! ENDOF
8 P         0  OF 12 COCT C! ENDOF  ENDCASE ;
9 P   HEX
10 P  10 CONSTANT OFF_TH    12 CONSTANT ON_TH
11 P -->
12 P
13 P
14 P
15 P
```

More variables and a word for changing pitch ranges. Five bits (3 valves and 2 side switches) can only give 31 different pitches. The SETOCT word and the COCT variable are used to change the register of these 31 pitches.

The constants ON_TH and OFF_TH are threshold values. When the breath controller value goes above ON_TH (12) I turn on a note and when it goes below OFF_TH (10) I turn off the note.

Screen 5

```
0 P : MLD  ( x---| send midi value x )
1 P        10 PC!  0 0 PC!   1 0 PC! ;
2 P  : ON   ( velocity, key# ---| send midi key-on )
3 P        90 CHNL C@ OR MLD MLD MLD ; ( status, key#, vel )
4 P  : OFF  ( key#---| send midi key-off )
5 P         80 CHNL C@ OR MLD MLD 0 MLD ;
6 P  : PROG ( p---| send midi program change )
7 P         C0 CHNL C@ OR MLD MLD ;  ( status, program # )
8 P  : CONT ( val,ctl---| send midi control change )
9 P         ?CTL C@ IF ( if cont was just sent forget stat )
10 P      ELSE B0 CHNL C@ OR MLD ( else send status byte )
11 P        ENDIF MLD  1 ?CTL C!  MLD ;
12 P  : PTCH ( x---| send midi pitch bend value x )
13 P        E0 CHNL C@ OR MLD MLD ;
14 P  : SUS ( x---| send midi sustain, FF-ON, 0-OFF )
15 P       40 CONT ; -->
```

This screen has all the Midi Commands. The basic word, MLD (midi load), sends out one byte of Midi data. The other words are based on the Midi software specifications.

The word CONT takes care of most of the continuous controller data. Since continuous controllers tend to send out long streams of data at a time, the midi specifications allow you to bypass the normal format of status byte followed by data byte. In the word CONT, after the first status/data pair is sent (B0, data) only the data byte is sent for any remaining uninterrupted controller data.

Screen 6

```
0 P   : SEND ( n---| send continuous controller #n value )
1 P         DUP CV C@ OVER MCV C@ OVER - ABS
2 P        ( compare current CV value with last one read )
3 P        2 < IF DROP DROP ( do nothing if change < 2 )
4 P        ELSE SWAP 2DUP MCV C! ( else store new value )
5 P        CTL C@ CONT ENDIF ;  ( and send midi control value )
6 P  : READ ( ---| read pushbuttons and continuous controllers )
7 P        0 ?CTL C!
8 P        10 PC@ KEYS C@ = ( test for change in pshbuttons )
9 P        3 0 DO I AIN ( read each of the 4 contin controls )
10 P       2/ I CV C! ( shift out lsb, store 7-bit midi val  )
11 P       I SEND LOOP ( send new control value out midi )
12 P       IF ELSE ( return to test after keys have settled )
13 P       10 PC@ KEYS C! ENDIF ;  ( read keys again and store )
14 P -->
15 P
```

This screen is the heart of the Midi Trumpet. Five of the switches are read and stored, and the continuous controllers are read and sent out the Midi port. The READ word first tests for changes in the pushbuttons. If a change is detected the program first waits for the keys to settle before storing the new values in the variable KEYS. During the wait the continuous controllers are read (AIN) and stored in CV. Notice that the least significant bit of the 8-bit controller value is thrown away (AIN 2/) since the Midi data accepts only a 7-bit value.

The continuous controller analog to digital conversion takes a relatively long time (200 microseconds). If you are not using the pedal you can cycle through only the first 3 controllers in the DO LOOP to save some time (3 0 DO instead of 4 0 DO). Notice also that the amount of continuous data sent out is cut down by sending a value only if it has changed by 2 or more.

Screen 7

```
0 P : DOIT   CASE
1 P          1 OF     CSUS C@    IF 0 SUS 0 CSUS C!
2 P                              ELSE     FF DUP SUS CSUS C!
3 P                              ENDIF   ENDOF
4 P         2 OF   10 PC@    18 AND   SETOCT     ENDOF
5 P         4 OF    10 PC@    18 AND  2/ 2/ 2/   PROG   ENDOF
6 P   ENDCASE ;
7 P    0 VARIABLE CHK
8 P : SPECIAL
9 P         20 PC@ FF XOR 7 AND DUP       IF
10 P       CHK C@ IF   DOIT   0 CHK C!    ELSE DROP ENDIF
11 P        ELSE   DROP 1 CHK C! ENDIF ;
12 P -->
13 P
14 P
15 P
```

This is one of the more "fun" programming parts of the Midi Trumpet.. There are 3 switches on the Trumpet which are not being used for pitch information. They can be programmed for any function you desire. I have chosen three possible applications but I encourage you to try your own.

The first of the green pushbuttons in line with the valves is programmed to toggle the sustain off and on. The second key is used along with the two side buttons to change the octave ranges - middle range, down 2 octaves, down 1 octave, and up one octave (see SETOCT on screen 4, the fourth and fifth bit go low when the key is pushed). The third key is used along with the two side buttons to change the synthesizer program from 0 to 3. The DOIT word sets up these three special key functions.

Note that the side buttons must be pressed and held while the special key is pressed. Think of the special key as entering the value being held on the two side buttons.

The SPECIAL word (special function key) implements the DOIT word. A check (CHK) is made when a special key is first pressed and then again when is it released. The check is used to ensure that the Midi info for the special key is sent only once while the key is down. (Without this the sustain would be cycled on and off rapidly while the key is down.)

Screen 8

```
0 P : KEYCALC ( calculate pitch from note table and coct )
1 P       KEYS C@ F8 XOR 1F AND
2 P       NOTE C@   COCT C@ +   PITCH C! ;
3 P : SETVEL
4 P        0 CV C@  10 + 7F MIN  VEL C! ;
5 P : KEYON
6 P       VEL C@ PITCH C@ ON   1 ?ON C!  KEYS C@ MKEYS C! ;
7 P  : KEYOFF
8 P       PITCH C@ OFF   0 ?ON C! ;
9 P -->
10 P
11 P
12 P
13 P
14 P
15 P
```

This screen defines some words used in the main horn program coming up on the next screen. The KEYCALC word calculates the key value for a Midi Note On command by looking up a pitch value from the NOTE table and then adding an octave offset value. The SETVEL word calculates a key velocity (initial loudness) value for a Midi Note On command using the breath value stored in CV(0) during the READ cycle. The next two words use the Keycalc and Setvel values to actually send a Midi Note On command or a Midi Note Off command.

Screen 9

```
0 P   : HORN ( --- | Main Program )
1 P
2 P        CR ." Midi Trumpet "  BEGIN
3 P        READ SPECIAL 0 CV C@ ?ON C@
4 P        IF ( breath ) OFF_TH >
5 P               IF MKEYS C@ KEYS C@ =
6 P               IF ELSE KEYOFF KEYCALC 0 CV C@ VEL C!
7 P                     KEYON ENDIF
8 P               ELSE KEYOFF
9 P               ENDIF
10 P        ELSE ( breath ) ON_TH >
11 P              IF  KEYCALC SETVEL KEYON ENDIF
12 P        ENDIF
13 P        ?TERMINAL  UNTIL ." Stop program "  ;
14 P HEX 0 801A C!   HORN
15 P  ;S
```

Finally, here is the main horn program. All the words between BEGIN and UNTIL are repeated forever or until someone sends something over the serial line by hitting a key on the computer terminal (?TERMINAL).

Inside the loop, first the trumpet keys are read and one round of continuous controller data is sent out the Midi port (READ); then the special keys are checked and executed. Then the breath value is compared to the on and off thresholds. If no note is currently on and the breath value is above the on-threshold, then a Key On Midi event is executed (lines 10 and 11). If a note is currently on and breath is still above the off-threshold but the key value has changed, then the old note is turned off (KEYOFF) and a new Midi Key On command is executed (lines 5, 6 and 7). Last of all, if a note is currently on (?ON=1) but the breath has gone below the off-threshold, then the note is turned off (line 8).

Line 14 clears the terminal buffer and then executes the HORN program.

MIDI Trumpet Program

```
 : TABLE <BUILDS 0 DO C, LOOP DOES> + ;    DECIMAL
0 0 0 0  4 TABLE CV  ( newly read controller values )
0 0 0 0  4 TABLE MCV ( midi'd controller values )
4 7 1 2  4 TABLE CTL ( midi controller #'s used)
     ( breath, modulation, volume, foot controller )
 78 81 79 77 80 83 82 84   73 76 74 74 72 75 72 72
 66 69 71 70 68 71 67 67   61 64 62 65 63 59 60 60
  32 TABLE NOTE      ( see Trumpet Fingering Charts )
  0 VARIABLE PRG    ( current midi prog number )
  0 VARIABLE KEYS   ( newly read pushbuttons )
 0 VARIABLE MKEYS ( midi'd pushbuttons )
  0 VARIABLE PITCH ( calculated midi key value )
 0 VARIABLE CHNL   ( midi channel )
 0 VARIABLE ?ON    ( note on = 1, note off = 0 )
 0 VARIABLE ?CTL   ( true if midi control was sent last )
 -->

DECIMAL
0 VARIABLE VEL     ( midi key velocity )
0 VARIABLE CSUS    ( current sustain value )
0 VARIABLE COCT    ( current octave offset )
: SETOCT CASE      ( set oct with 2 side-mount keys )
          24 OF 0 COCT C! ENDOF
           16 OF 24 MINUS COCT C! ENDOF
           8   OF 12 MINUS COCT C! ENDOF
           0   OF 12 COCT C! ENDOF   ENDCASE ;
 HEX
 10 CONSTANT OFF_TH      12 CONSTANT ON_TH
-->
```

```
: MLD  ( x---| send midi value x )
        10 PC!  0 0 PC!   1 0 PC! ;
: ON   ( velocity, key# ---| send midi key-on )
       90 CHNL C@ OR MLD MLD MLD ; ( status, key#, vel )
: OFF  ( key#---| send midi key-off )
         80 CHNL C@ OR MLD MLD 0 MLD ;
: PROG ( p---| send midi program change )
        C0 CHNL C@ OR MLD MLD ;  ( status, program # )
: CONT ( val,ctl---| send midi control change )
        ?CTL C@ IF ( if cont was just sent forget stat )
         ELSE B0 CHNL C@ OR MLD  (else send status byte )
         ENDIF    MLD  1 ?CTL C!  MLD ;
 : PTCH ( x---| send midi pitch bend value x )
         E0 CHNL C@ OR MLD MLD ;
 : SUS ( x---| send midi sustain, FF-ON, 0-OFF )
       40 CONT ; -->

: SEND ( n---| send continuous controller #n value )
        DUP CV C@ OVER MCV C@ OVER - ABS
       ( compare current CV value with last one read )
       2 < IF DROP DROP ( do nothing if change < 2 )
       ELSE SWAP 2DUP MCV C! ( else store new value )
       CTL C@ CONT ENDIF ; ( and send midi control value )
: READ ( ---| read pushbuttons and continuous controllers )
        0 ?CTL C!
       10 PC@ KEYS C@ = ( test for change in pshbuttons )
       4 0 DO I AIN ( read each of the 4 contin controls )
       2/ I CV C! ( shift out lsb, store 7-bit midi val )
       I SEND LOOP ( send new control value out midi )
       IF ELSE ( return to test after keys have settled )
       10 PC@ KEYS C! ENDIF ; ( read keys again and store )
-->
```

```
: DOIT   CASE
       1 OF   CSUS C@ IF 0 SUS 0 CSUS C!
                         ELSE FF DUP SUS CSUS C!
                       ENDIF   ENDOF
       2 OF   10 PC@   18 AND   SETOCT   ENDOF
       4 OF   10 PC@   18 AND   2/ 2/ 2/ PROG ENDOF
 ENDCASE ;
  0 VARIABLE CHK
 : SPECIAL ( special function keys )
      20 PC@   FF XOR 7 AND DUP   IF
      CHK C@   IF DOIT 0 CHK C! ELSE  DROP  ENDIF
      ELSE DROP 1 CHK C! ENDIF ;
-->



: KEYCALC ( calculate pitch from note table and coct )

      KEYS C@ F8 XOR 1F AND
      NOTE C@    COCT C@ +    PITCH C! ;
( : BREATH    0 CV C@ ; )
 : SETVEL
      0 CV C@ 10 + 7F MIN   VEL C! ;
 : KEYON
       VEL C@ PITCH C@ ON   1 ?ON C!   KEYS C@ MKEYS C! ;
 : KEYOFF
       PITCH C@ OFF    0 ?ON C! ;
 -->




```

```
: HORN ( --- | Main Program )

        CR ." Midi Trumpet "   BEGIN
        READ SPECIAL   0 CV C@   ?ON C@
        IF ( breath ) OFF_TH >
                IF MKEYS C@ KEYS C@ =
                IF ELSE KEYOFF KEYCALC   0 CV C@   VEL C!
                      KEYON ENDIF
                ELSE KEYOFF
                ENDIF
        ELSE ( breath ) ON_TH >
                IF   KEYCALC SETVEL KEYON ENDIF
        ENDIF
        ?TERMINAL   UNTIL ". Stop program " ;
HEX 0 801A C!( so nothing is waiting for ?terminal ) HORN
;S
```

Midi Trumpet Test Programs

```
HEX ( All numbers shown are in base 16, hexidecimal )

                        PUSHBUTTON  TEST
: PTST (shows runnining status of port 10H )
            CR BEGIN
                  10 PC@ F8 XOR . (print value of port# 20 )
                  D EMIT  ( return screen cursor )
                  ?TERMINAL (stop program if any terminal key is hit)
            UNTIL ; (else repeat the program )

                  CONTINUOUS  CONTROLLER  TEST
: CCTST (x---| running status of controllers, x= 0,1,2, or 3 )
            CR     BEGIN DUP AIN . D EMIT       170 0 DO LOOP
            ?TERMINAL UNTIL DROP ;

                        MIDI  TEST  WORDS
: MLD  (x---| send value x out the midi port )
            10 PC! ( loading x )  0 0 PC! 1 0 PC! ( load pulse ) ;
: KEYON  (x---| send midi key on for note x )
            90 MLD (send keyon status byte, midi channel = 0 )
               MLD (send key number x, 0 to 127 only please )
            40 MLD (send key velocity of 40 ) ;
: KEYOFF  (x---| send midi key off for note x )
            80 MLD   MLD  0 MLD (status=80, key#=x, velocity=0) ;
: PROG  (x---| send midi program change )
            C0 MLD MLD (status=C0, prog#=x) ;



( Loading a simple Autostart program into EEPROM's Screen 3 )
( Always turn on micro with WRT ENB switch down )

EDITOR DECIMAL
3 LIST
0 P : GREET ." HELLO EVERYBODY " CR CR ;
1 P GREET
2 P  ;S
3 LIST
( The program is in a RAM buffer memory.  To load it into the EEPROM's Screen
#3, turn on the WRT ENB switch in the up position and Flush )
FLUSH
( Turn off the WRT ENB switch )
( Now whenever you reset or power on, the GREET word will be loaded and
executed. )
*****************************************************************
```

Midi Trumpet Memory Map

0000 to 7FFF	U9	ROM	32K FORTH
8000 to 9FFF	U12	RAM	Variables, Stacks, Dictionary, 2 Screens
A000 to BFFF	U10	EEPROM	Screens 3 through 10

****

8000 to 83FF	Variables and Stacks
8400 to 87FF	Start of User Dictionary Space
8800 to 8BFF	
8C00 to 8FFF	
9000 to 97FF	Editor Screen Buffers
9800 to 9BFF	Screen #1
9C00 to 9FFF	Screen #2

****

A000	Screen 3 (Autostart if Colon ASCII# is at A000)
A400	Screen 4
A800	Screen 5
AC00	Screen 6
B000	Screen 7
B400	Screen 8
B800	Screen 9
BC00	Screen 10

****

(Hardware note: Circuit diagrams show the Forth ROM residing in 8000 to FFFF since the 8088 reset vectors must be at ROM address FFFFF0. The ROM software must be inverting address bit#15 to give the memory map shown above.)

MIDI TRUMPET CONNECTIONS

Trumpet to Micro (D25 male to D25 male cord)

```
 13  12  11  10  9   8   7   6   5   4   3   2   1
   25  24  23  22  21  20  19  18  17  16  15  14
```

Ground - 1,2,3,14,15, 12,13,24,25

4 - minus 5 volts (for preamp)
5 - trumpet amplitude
6 - slider 1
7 -
8 - pushbutton 7
9 - pushbutton 0
10 - pushbutton 1
11 - pushbutton 2
23 - pushbutton 5
22 - pushbutton 4
21 - pushbutton 3
20 - pushbutton 6
19 - plus 4 volts (for sliders)
18 - slider 2
17 - trumpet signal (from preamp)
16 - plus 12 volts (for preamp)

Micro to Interface board (DIP ribbon cables)

J1 - ADC
1-AD0(pickup volume), 2-AD1(slider1), 3-AD2(slider2),
4-AD3(pedal), 9-gnd, 14-Ref/2, 10-13 empty, 15-16 empty

J2 - 1-5 in from trumpet pushbuttons 0,1,2,6,7
9-16 midi out to UART pins 26-33

J3 - 1-3 in from trumpet pushbuttons 3,4,5

J5 - 1-gnd, 2-, 3-500Khz to UART, 4-gnd,
5-, 6-8 Power GND, 9-, 10-WRT ENB, 11-, 12-wrt pulse to UART, 13-Reset
14-16 Power 5v.

Interface Board to Front Panel

Reset	- 14	
Midi Outs	- 15, 16	5v pullups - 1,2
Pedal	- 4	+5v - 5
Trumpet pickup	- 7	Gnd - 8
Write Enable Switch		- 6

Midi Trumpet Controls

Pushbuttons

There are 8 pushbuttons on the Trumpet - 3 mounted inside the valves, 3 more on top, 2 mounted sideways. The pushbuttons can be read from the microcomputer as an 8-bit value with each bit representing one of the 8 pushbuttons. The command, 10 PC! (10 is a hexadecimal number) will read pushbuttons 0,1,2,7,8 and 20 PC! will read pushbuttons 3, 4, 5 (the value is pushed onto the stack).

Each of the 8 bits is either high or low. For the three valves the corresponding bit is 1 when the valve is down and 0 when the valve is up. For the other bushbuttons, the corresponding bit is 0 when the key is pressed.

Each key has been assigned a bit position in one of two 8-bit locations. The three valves are the 3 least significant bits of PC address 10 hex, with numerical weights of 1, 2, and 4; while the two side mounted switches have weights of 8 and 10(hex) at the same location. The three top pushbuttons are the least significant bits at PC 20 with weights of 1, 2, and 4.

For example, a value of E2 hex read from the stack after the command 10 PC! translates to the bit pattern 111 00 010 which indicates that the middle valve is down and the two side pushbuttons are down (the 3 top bits are not connected to any keys and thus are always high).

Continuous Controllers

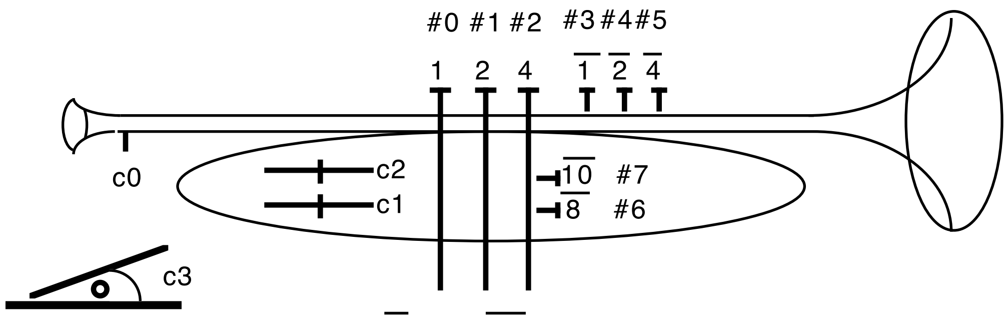
The Trumpet has 4 continuous controllers. Controller#0 is derived from the pickup on the mouthpiece. The pickup signal is amplified and then sent to an envelope follower to give a control voltage which rises and falls with the trumpet loudness. Controllers 1 and 2 are the two marked sliders mounted on the trumpet. Controller#3 is a pedal which can be connected at the micro's front panel.

Each controller can be connected to an Analog to Digital Converter within the microcomputer. The 8-bit value can be read by using the command, n AIN, after which the value for controller #n is left on the Forth stack.

Midi Output

The Microcomputer can put out midi data based on horn activity or independently. It's all in how you program it. To send an 8-bit Midi value first load the value into port 10 (hex) and then pulse port 0 (0 0 PC! 1 0 PC!). Simpler yet, use the MLD general purpose midi load word.

Read the Midi spec sheet to see how Midi data is formatted.



1, 2, 4, $\overline{8}$, and $\overline{10}$ are read from10 PC@.

$\overline{1}$, $\overline{2}$, and $\overline{4}$ are read from 20 PC@.

Trumpet Fingering Charts

Valves	Trumpet Overtones					
7 (all open)	F#1	C#1	F#2	A#2	C#2	F#3
6	A1	E1	A2 (440Hz)	C#2	E2	A3
5	G1	D1	G2	B2	D2	G3
4	A#1	F1	A#2	D2	F2	A#3
3	G#1	D#1	G32	C2	D#2	G#3
2	B1	F#2	B2	D#2	F#3	B3
1						
0 (all closed)	C1 (mid C)	G2	C2	E2	G3	C3

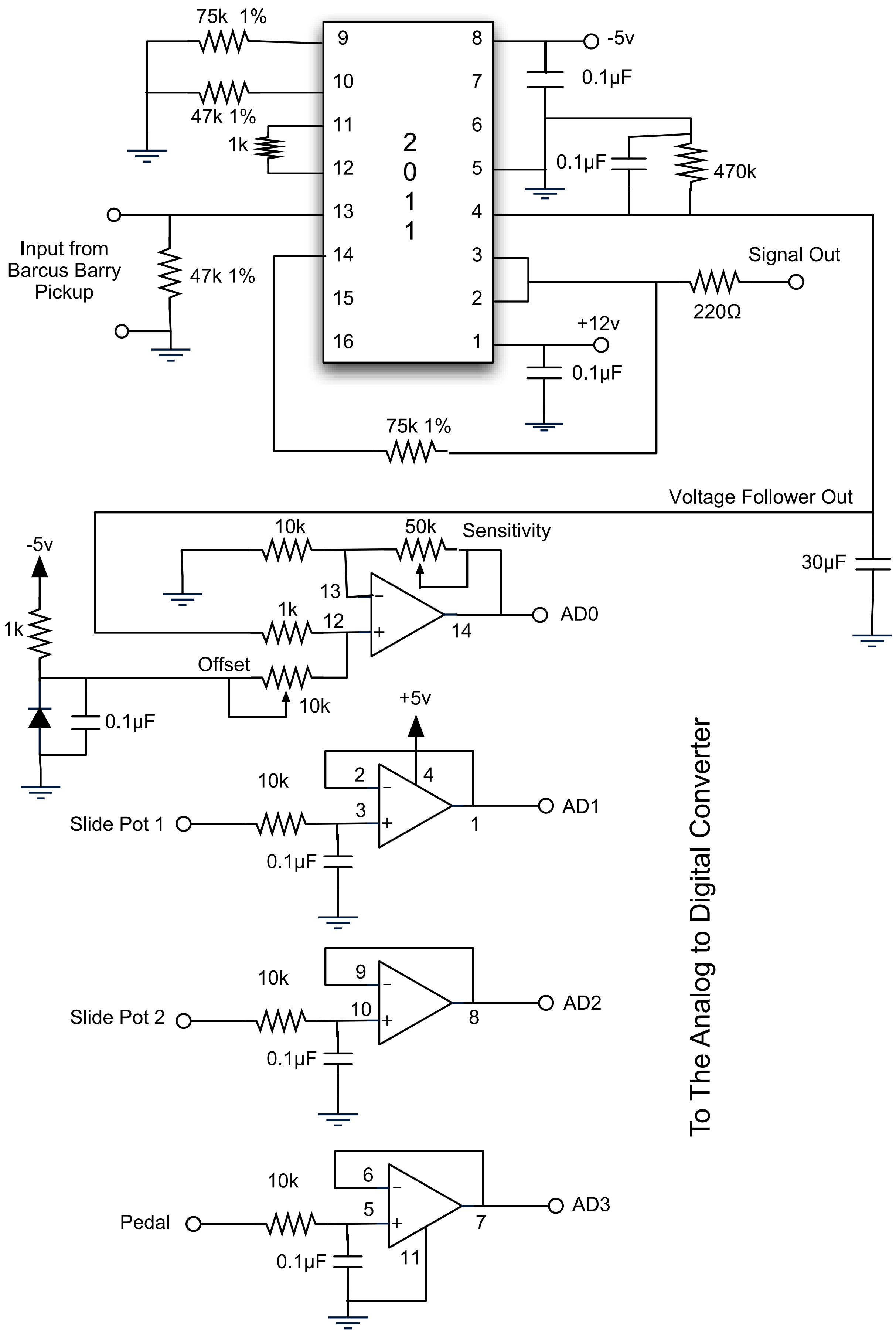
Scheme for translating valve positions to Midi Key numbers, using the two side-mounted switches for extra "overtone" information. (Midi Key Number / Key pitch = 3 valve positions)

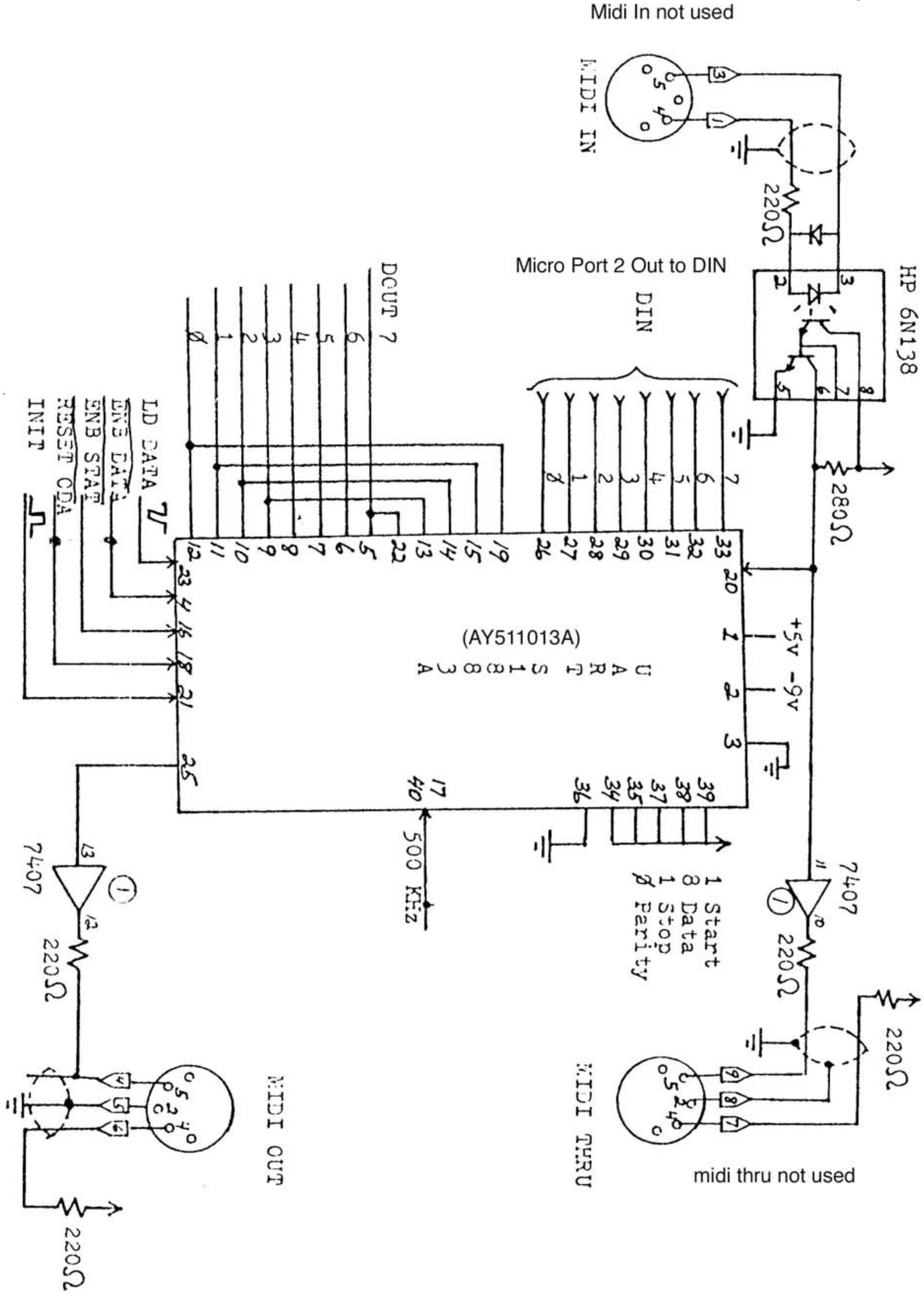
Position 00	Position 01	Position 10	Position 11
59/B1 = 2	66/F#2 = 7	72/C2 = 0,3	77/F2 = 4
60/C1 = 0	67/G2 = 0	73/C#2 = 7	78/F#3 = 7
61/C#1 = 7	68/G#2 = 3	74/D2 = 4,5	79/G3 = 5
62/D1 = 5	69/A2 = 6	75/D#2 = 2	80/G#3 = 3
63/D#1 = 3	70/A#2 = 4	76/E2 = 6	81/A3 = 6
64/E1 = 6	71/B2 = 2,5		82/A#3 = ?
65/F1 = 4			83/B3 = 2
			84/C3 = 0

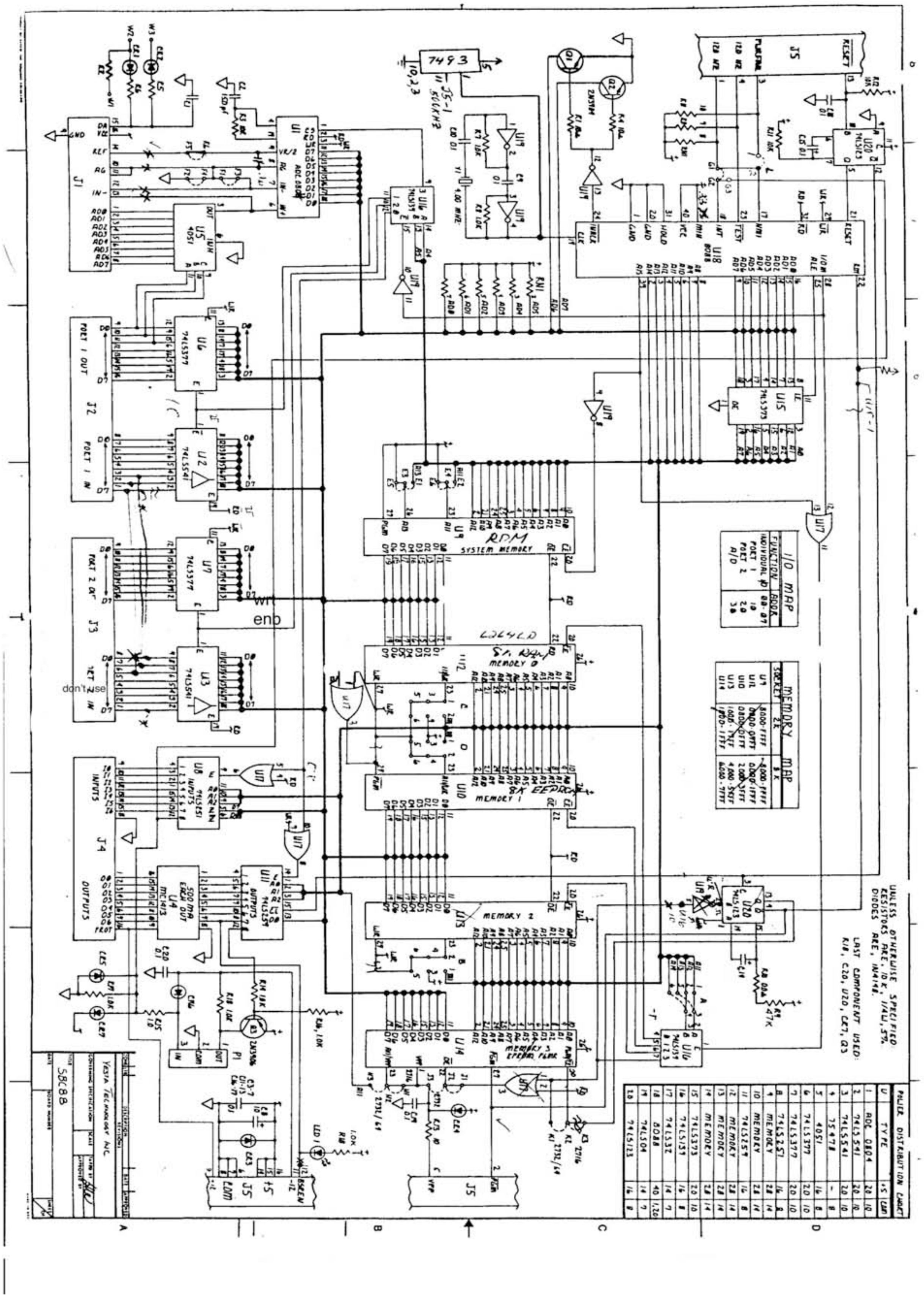
Table above rearranged with Midi Key numbers. To be used as the Note Table in Trumpet program.

Valves	00	01	10	11
0	60	67	72	84
1	-	-	-	-
2	59	71	75	83
3	63	68	72	80
4	65	70	74	77
5	62	71	74	79
6	64	69	76	81
7	61	66	73	78

Preamp and Follower for MIDI Trumpet




Midi In not used
MIDI IN
Micro Port 2 Out to DIN
HP 6N138
DIN
DOUT 7
LD DATA
ENB DATA
ENB STAT
RESET CDA
INIT
(AY511013A)
UART S1883A
+5v -9v
500 KHz
1 Start
8 Data
1 Stop
Ø Parity
7407
220Ω
280Ω
MIDI THRU
midi thru not used
MIDI OUT


7493
U18 8088
ROM
SYSTEM MEMORY
MEMORY 0
MEMORY 1
8K EEPROM
MEMORY 2
MEMORY 3
I/O MAP
MEMORY MAP
POWER DISTRIBUTION CHART
PORT 1 OUT
PORT 1 IN
PORT 2 OUT
wr
enb
don't use
INPUTS
OUTPUTS
J1
J2
J3
J4
J5
YESTA TECHNOLOGY INC
SBC88