The MIDI / Schober Project - Part 2

The MD-2 PC Board

The job of the MD-2 board is to convert swell shoe movement into the appropriate electrical signals. This page describes how it works and what it does.

Keep in mind that the MD-2 program is easily changeable, and can be adapted to many different purposes. In my organ I have two swell shoes, which will use two MD-2 boards. At this point I am planning to use one for each keyboard, but I may at some point decide to use one of these as a crescendo pedal.

Swell Shoe Sensor

In the original Schober organs, the swell shoe controlled a potentiometer ("pot") through a rack-and-pinion gear arrangement. In early organs this pot directly controlled the volume, but after some time this developed a problem --- the pot would become noisy and introduce crackling into the audio. Schober thus redesigned the circuit: the pot would control the intensity of a light bulb, the bulb would shine on a cadmium sulfide cell, the cell's resistance would change with the light intensity, and this would be used to control the volume. The advantage of this system is that the thermal inertia of the lamp causes just enough delay to smooth out any irregularities.

I've replaced the potentiometer with a digital rotary encoder, shown in the photo. The encoder is a three-terminal device which acts like a pair of switches that alternately open and close as the encoder shaft is turned. As you turn the shaft, the switches operate like this:

Switch B	Switch A
open	open
open	close
close	close
close	open
open	open

etc. etc.

In other words, both switches start open. After a tiny turn, switch A closes. A slight turn further and B closes. Another slight turn and A opens, and then a bit later B also opens. This repeats, over and over, a total of 24 times over one 360-degree revolution. (For digital freaks, this is called a 2-bit Gray code.) If the encoder shaft turns in the opposite direction, the sequence is reversed.

The advantage of this scheme is that if the switches age and become intermittently noisy, the digital effect is minimal. The encoders I chose cost about $2 each and are guaranteed for 10,000 revolutions (they are part number 318-ENC160-24P from Mouser Electronics, a Taiwan Alpha Rotary Encoder model RE160-40E3-20A-24P). If that isn't good enough, I can always replace them with $70 encoders which use optics instead of physical switches and which are guaranteed for something like 10 million revolutions. (There are several other mechanical encoders in the Mouser catalog as well.)

But there is also a disadvantage. Since the two-bit code repeats over and over, the system can recognize tiny swell shoe movement, but it can't be sure exactly how far the swell shoe pedal is depressed. My software will require a learning sequence - when you first turn the organ on, regardless of where the swell shoe is actually positioned, the MD-2 board will assume that it is at minimum volume. The organist will have to do one complete top-to-bottom-to-top movement so that the MD-2 software can learn what the maximum and minimum settings are. After that, it will figure out the current position by keeping track of the history of up and down motions.

MD-2 function

Swell shoe sensor position is monitored by a 68HC11 microprocessor; its circuitry is very similar to the MD-1 board, and is explained later. The processor keeps track of swell shoe position and does three things each time it senses that the swell shoe has been moved:

1. It outputs a MIDI controller message via its 5-pin MIDI OUT connector. The channel number is set by a four-pole DIP switch, similar to that on the MD-1 board.

2. It outputs a 7-bit code out the 9-pin serial connector. A serial cable sends this code to the MD-1 board. The MD-1 program accepts this number and uses it as the velocity byte in current note-on messages.

3. The MD-2 board also has space for a PGA2310PA audio integrated circuit. This is a dual volume control (for stereo applications) which is controlled digitally by the microprocessor. This allows me to directly control audio volume for other signal sources.

My main organ tone generator is either the Hauptwerk program, or the Kloria MyOrgan program; both of these use samples of real organ pipes. These two programs both accept MIDI controller messages, but Hauptwerk ignores velocity commands. (I don't yet know about MyOrgan.)

My secondary tone source is a Roland CM32P. This is a MIDI tone module which uses short samples to play sounds. It has a number of sounds, but I anticipate using only the piano in conjunction with Hauptwerk's theatre organ samples. This module uses the velocity code in note-on messages.

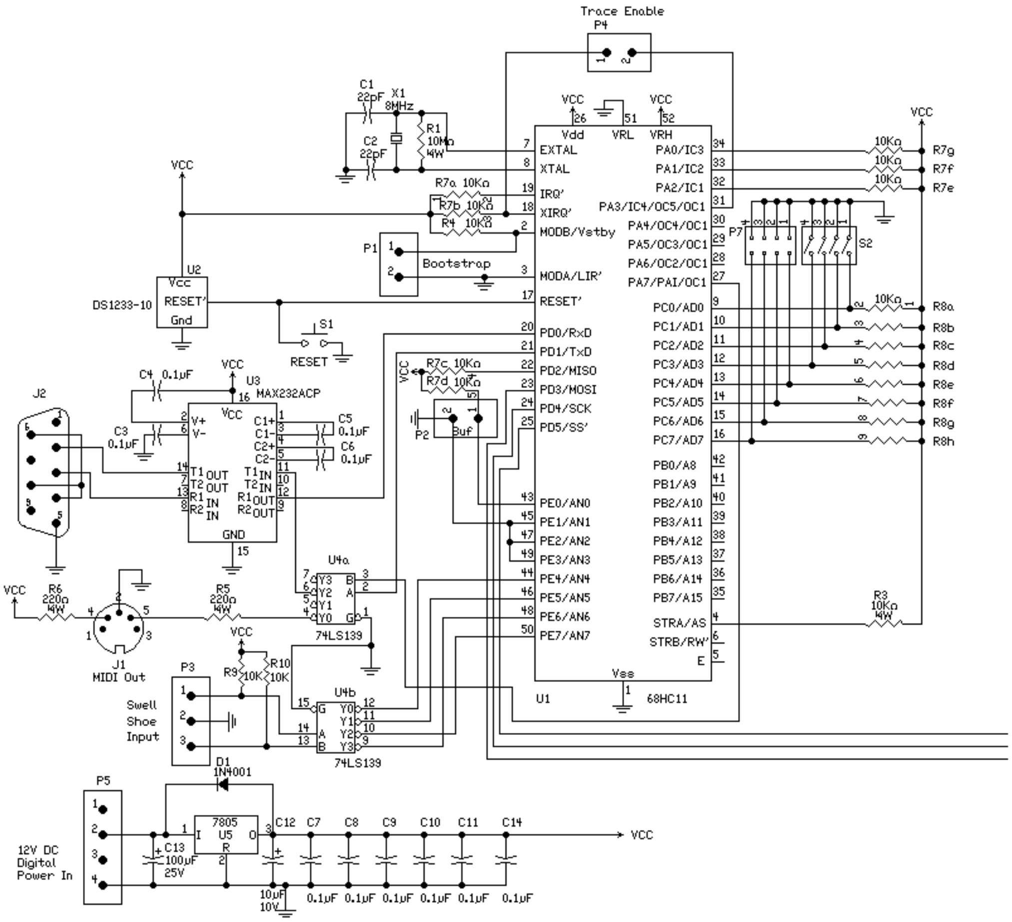
At this point I don't see the need for an audio volume control so, although the MD-2 board has space for the PGA2310PA audio volume control, that part of the boards is not populated, nor have I written the software for that function.

MD-2 Circuitry

The MD-2 board is divided into two parts --- the digital side on the left, and the analog side on the right. To prevent the digital control signals or noise from getting into the analog circuits, the two sides use separate grounds and separate power supplies for isolation. The digital side is shown in Figure 1, while the analog side is in Figure 2.

Digital circuitry

The digital circuit in Figure 1 is actually very similar to the MD-1 board.



Let's explain the various parts, starting at the top left corner of the 68HC11.

(a) Top left is the 8 MHz crystal and attached circuitry, which sets the speed of the processor. The 68HC11 internally divides this by 4, so the main clock frequency of the processor is 2 MHz.

(b) Several resistors under that (and resistors elsewhere) connect various pins of the HC11 to +5-volt Vcc, to provide a pullup function - that is, to hold these pins at +5 volts when they are not used for other functions.

(c) Jumper P1 allows bootstrapping the HC11. In normal mode, this jumper would be open. When pins 1 and 2 of the jumper are shorted together, the processor goes into a bootstrap mode, where it can be loaded with a program through the serial port. This is the method that the EEPROM software would initially be loaded.

(d) The DS1233-10 IC, U2, is a reset circuit. It monitors the Vcc power, and automatically resets the processor -- i.e., restarts it -- when the power goes below about 4.5 volts.

(e) The 68HC11 has a built-in serial I/O port which can run at the standard baud rates, as well as at 31.25 kHz, the MIDI rate. It uses pins 20 and 21. This serial port connects both to 9-pin RS-232 connector J2, and also to MIDI OUT jack J1.

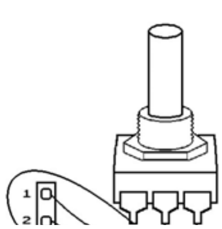
Like the MD-1, the MD-2 board has two serial ports, though only one of them can be used at a time:

- There is 9-pin DB9F connector, which can be connected to the comm port of a PC. It is does two jobs: (1) During program development, it is used for loading the software into the processor, and also for program testing. (2) During operation, it outputs velocity information which is sent to the corresponding MD-1 board. The MAX232 IC, U3, is used as the converter between the low-level TTL voltages used by the HC11, and the positive and negative larger voltages used by typical RS-232 ports at the DB9F connector. It includes a charge pump which provides the necessary +10 and -10 power for output).

- There is also a 5-pin MIDI connector, which is the main MIDI output.

Serial output from U1 pin 21 goes through U4a, a 74LS139. I'm using it to steer the output to either the MAX232 or the MIDI jack. It is controlled by its B input pin, which comes from pin 27 of the HC11 - it is normally held high on CPU reset, so the normal connection is to the 9-pin connector. If the 68HC11 wants to output MIDI, it grounds this B input (via U1 pin 27), and changes the baud rate to 31.25 kHz.

(f) P3 is the input connection for the swell shoe sensor. IC U4b is a buffer which isolates the sensor inputs from the microprocessor to prevent damage to it, but it also converts the Gray code output of the sensor into a binary output to the processor.
This function could just as well be done in software, but it seemed like a good use for the spare half of U4, and it also provides some buffering and isolation. The encoder wiring is shown here (it is labelled C4 in the block diagram).



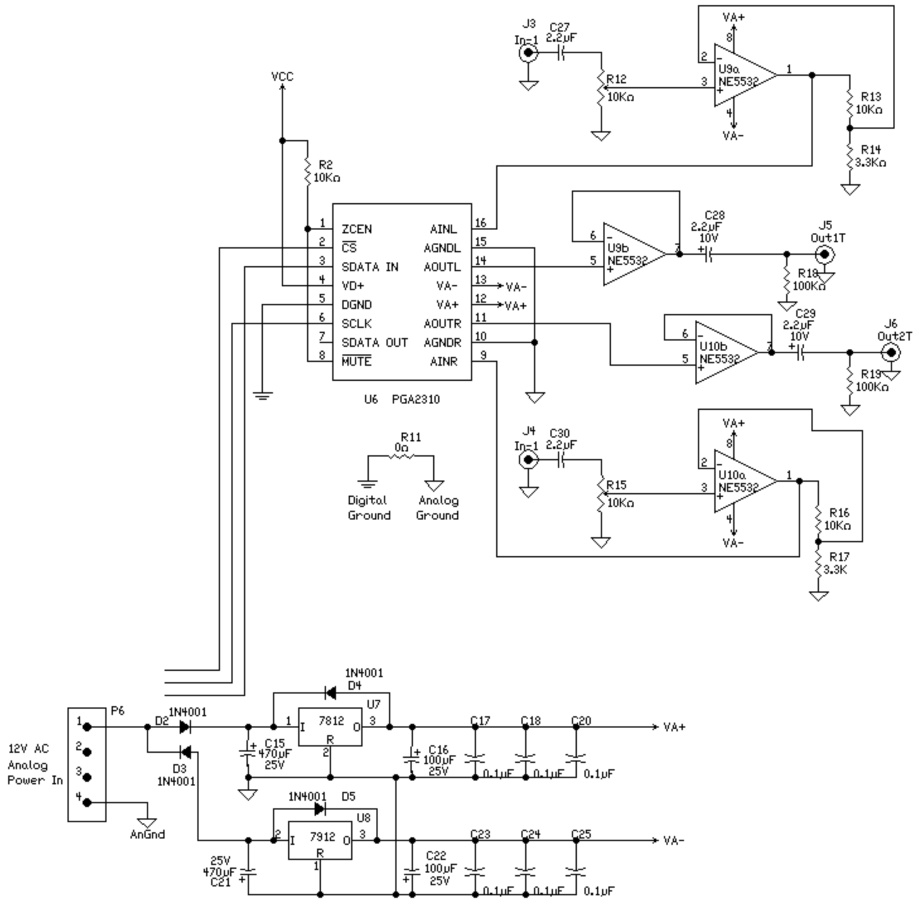
(f) Jumpers P2 and P4 are left open in organ applications, but they allow this board to run with the Motorola Buffalo version of the 68HC11 for debugging purposes. P2 selects whether the Buffalo program will run, or whether the HC11 will instead jump to location B600 on bootup. P4 is used to enable tracing and breakpoints. As mentioned, these two jumper headers are here strictly for debugging and other applications.

(g) Power supply circuitry at the bottom includes a 7805 regulator and filtering, so the board can be fed from an
unregulated DC source, or even from a "wall wart" converter. If you use a +5-volt supply, then the 7805 can be left out and its terminals 1 and 3 shorted together.

(h) Switch S2 is a four-pole DIP switch which sets the channel number for the MIDI output. MIDI signals can be assigned a channel from 1 to 16.

(i) Connector pins P7 provide four auxiliary inputs "just in case".

Figure 2 shows the analog portion of the board.

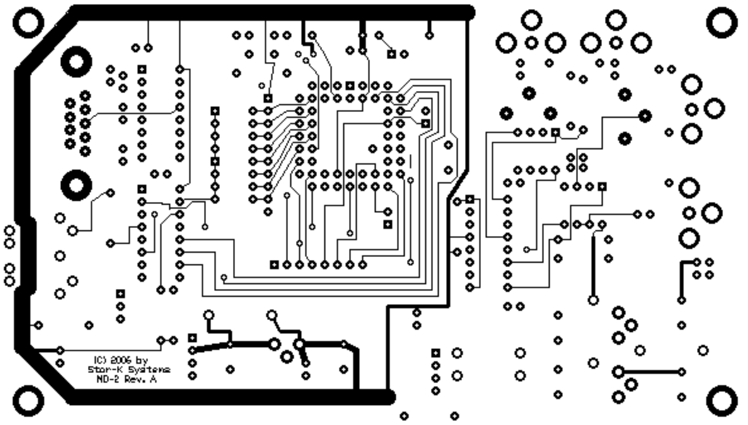


The heart of the board is the PGA2310 IC, which is a stereo volume control chip. It receives digital control signals (on the three lines coming in from the left) from pins 23, 24, and 25 of the processor. The right side of the IC is strictly analog, and has two inputs (coming from J3 and J4) and two outputs (going out to J5 and J6). Each of these is buffered by an NE5532 op-amp. A separate power supply at the bottom provides +12 and -12 volts to the analog circuits. Resistor R11 is just a jumper which connects the analog and digital grounds together at one specific point.

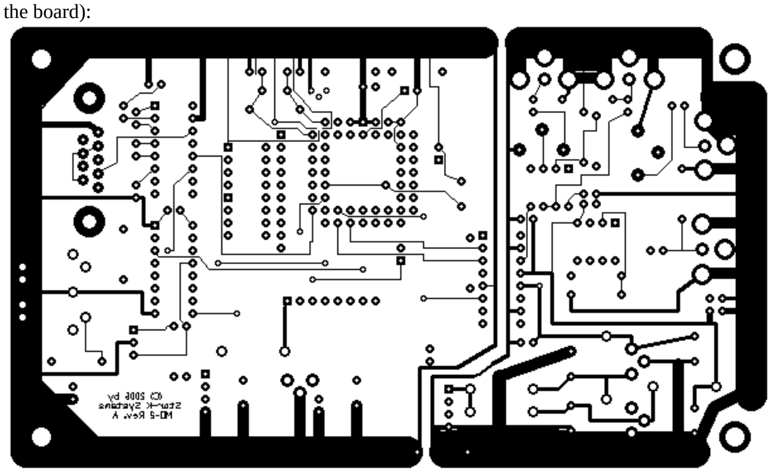
PCB Layout

The following figures show the MD-2 PC board layout. The board is a two-sided board with plated through holes, and the size is 3.5 by 6 inches. While I do not sell any blank PC boards, kits, or completed boards, you can make your own boards from the layouts shown below, or there are a number of PC board manufacturers who can make a board for you if you send them the Gerber files available [here.]

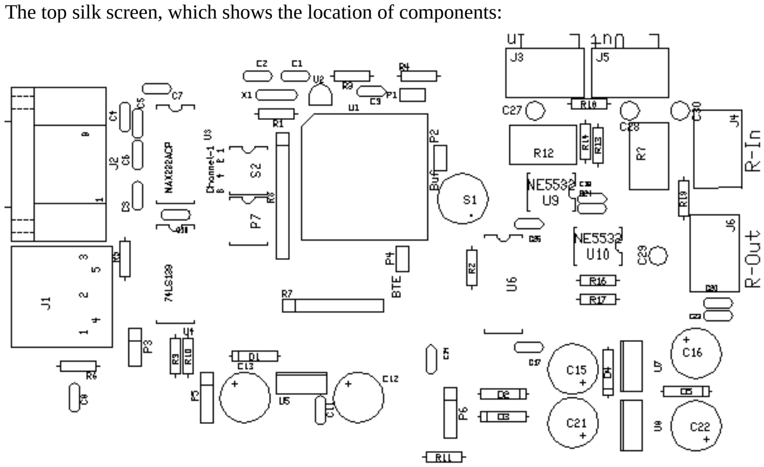
The top solder side:



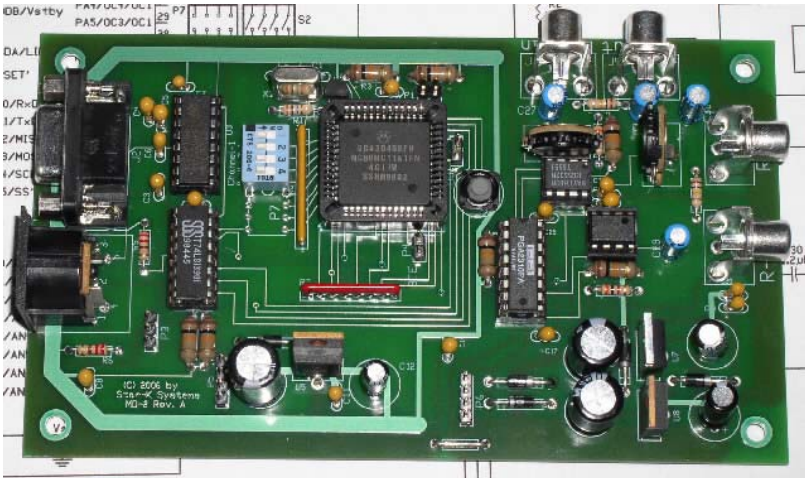
The bottom solder side (as viewed from the top, through the board):



The top silk screen, which shows the location of components:



The completed PC board:



THE SOFTWARE

The 68HC11A1 has 512 bytes of EEPROM and 256 bytes of RAM, and only about 50% of each is used in the program. Here is the listing for version 1.0.

```
                * MD-2 Swell Shoe program
                * Copyright (C) 2006 by Peter A. Stark

                * Version 1.0 8/29/2006 - original
                * THIS VERSION ONLY OUTPUTS SERIAL OUTPUT ON RS-232 AND MIDI
                * AUDIO VOLUME CONTROL IS NOT IMPLEMENTED

                * I/O bits:
                *   Inputs
                * PD0: Serial RS232 input from J2
                * PC3-0: Channel DIP switch S2
                * PC7-4: extra inputs on P7
                * PE0: Jumper to ground for Buffalo P2
                * PE7-4: Rotary encoder signal from P3
                * MODB: Bootstrap mode jumper P1
                * XIRQ': Trace enable from P4

                *   Outputs
                * PA7: H=RS232 (default), L=MIDI out
                * PD1: Serial out to RS232 or MIDI to J2
                * PD5-3: SS', SCK, MOSI to PGA2310


                **************************
                *   GENERAL SYSTEM EQUATES
                **************************

0000            RAM     EQU   $0000     START OF RAM
1000            REG     EQU   $1000     START OF REGISTERS
b600            EEPROM  EQU   $B600     START OF EEPROM
b7ff            ENDEEP  EQU   $B7FF     END OF EEPROM
1000            PORTA   EQU   $1000     PORT A DATA & TIMER - B7=BI, B6-3=O,B2-1=I
1026            PACTL   EQU   $1026     PORT A CONTROL
1004            PORTB   EQU   $1004     PORT B DATA - OUTPUT ONLY
1003            PORTC   EQU   $1003     PORT C DATA - BI
1005            DDRC    EQU   $1005     PORT C DIRECTION
1008            PORTD   EQU   $1008     PORT D DATA - 6 BITS BI & SCI/SPI
1009            DDRD    EQU   $1009     PORT D DIRECTION
100a            PORTE   EQU   $100A     PORT E - INPUT ONLY & A/D
102b            BAUD    EQU   $102B     SCI BAUD REG
102c            SCCR1   EQU   $102C     SCI CONTROL 1 REG
102d            SCCR2   EQU   $102D     SCI CONTROL 2 REG
102e            SCSR    EQU   $102E     SCI STATUS REG
102f            SCDAT   EQU   $102F     SCI DATA REG
1028            SPCR    EQU   $1028     SPI CONTROL REG
1029            SPSR    EQU   $1029     SPI STATUS REG
102a            SPDR    EQU   $102A     SPI DATA REG
103a            COPRST  EQU   $103A     COP RESET REG
000b            BXCONT  EQU   11        CONTROLLER CHANNEL = 11

                *****************
                *   RAM LOCATIONS
                *****************

0000                    ORG   RAM

                * SET UP A CIRCULAR SERIAL OUTPUT BUFFER
                * CAUTION - BUFFER MUST START AT $0000 !
```

```
                         * EMPTY IF BUFIN=BUFOUT, DON'T WORRY ABOUT FULL -
                         * HOPEFULLY WILL NEVER HAPPEN
0000                     BUFFER  RMB   64   SERIAL OUTPUT BUFFER
0040                     BUFIN   RMB   2    PUT NEXT BYTE HERE POINTER
0042                     BUFOUT  RMB   2    NEXT BYTE TO OUTPUT POINTER
0044                     SWSHOE  RMB   1    CURRENT SW SHOE POSITION
0045                     STAPOS  RMB   1    SHOE STARTING POSITION
0046                     PREVSH  RMB   1    PREVIOUS POSITION
0047                     VOLUME  RMB   1    CURRENT VOLUME
0048                     TIVOL   RMB   1    VOLUME DOCTORED UP FOR TI CHIP
0049                     RSVOL   RMB   1    VOLUME DOCTORED UP FOR RS-232 OUTPUT
004a                     BXVOL   RMB   1    VOLUME FOR BX CONTROL CHGE MSG
004b                     BXCHAN  RMB   1    BX + CHANNEL NUMBER
007f                     STACK   EQU   $007F
                         ****************
                         * Start and Initialize ports
                         ****************
b600                             ORG   EEPROM
b600 8e 00 7f            COLDST  LDS   #STACK
b603 7f 10 04                    CLR   PORTB     OUTPUT - NOT USED
b606 7f 10 05                    CLR   DDRC      PORT C IS ALL INPUT
b609 86 38                       LDAA  #$38      PORT D IS INPUT EXC SERIAL...
b60b b7 10 09                    STAA  DDRD      ...AND SPI OUTPUT
b60e 7f 10 0a                    CLR   PORTE     PORT E IS ALL INPUT
                         **********
                         *   Initialize the SCI serial port
                         *        use $30 for 9600 baud during testing
                         *        use $20 for 31.25K baud for final version
                         **********
b611 ce 10 00                    LDX   #REG      POINT TO REGISTERS
b614 1c 00 80                    BSET  PORTA-$1000,X $80 PA7=H FOR RS232
b617 1c 26 80                    BSET  PACTL-$1000,X $80 ... AND OUTPUT
b61a 86 20                       LDAA  #$20      SET 9600 BAUD TEMPORARY
b61c b7 10 2b                    STAA  BAUD      BAUD REGISTER
b61f 86 00                       LDAA  #$00      SET 8X1, NO WAKEUP
b621 b7 10 2c                    STAA  SCCR1
b624 86 0c                       LDAA  #$0C
b626 b7 10 2d                    STAA  SCCR2     ENABLE
                         **********
                         *   Initialize SPI port not needed until we add audio control
                         **********
                         **********
                         *  WARMST - WARM START
                         **********
b629 b6 10 03            WARMST  LDAA  PORTC     GET CHANNEL NUMBER
b62c 84 0f                       ANDA  #$0F      MASK IT
b62e 8b b0                       ADDA  #$B0      MAKE INTO CTRL CHG COMMAND
b630 97 4b                       STAA  BXCHAN
b632 7f 00 47                    CLR   VOLUME    INITIAL = 0
b635 7f 00 45                    CLR   STAPOS    STARTING POSITION = 0
b638 86 04                       LDAA  #4
b63a 97 46                       STAA  PREVSH    PREV POSITION = 4
b63c 4f                          CLRA
b63d 5f                          CLRB
b63e dd 40                       STD   BUFIN     BUFFER IS EMPTY
b640 dd 42                       STD   BUFOUT
b642 86 55               MAINLP  LDAA  #$55
b644 b7 10 3a                    STAA  COPRST    RESET COP TIMER
b647 86 aa                       LDAA  #$AA      JUST IN CASE
b649 b7 10 3a                    STAA  COPRST
b64c b6 10 0a                    LDAA PORTE      GET DATA
b64f 43                          COMA
b650 44                          LSRA
b651 44                          LSRA
b652 44                          LSRA
b653 44                          LSRA
b654 97 44                       STAA SWSHOE
b656 90 46                       SUBA PREVSH     SUBTRACT PREVIOUS
b658 27 6a                       BEQ  OUTPUT     NO CHANGE
                         * FIGURE OUT IF UP OR DOWN
b65a 81 07                       CMPA #$7        DOWN CUR=8 PREV=1
b65c 27 0e                       BEQ DOWN
b65e 81 f9                       CMPA #$F9       UP CUR=1 PREV=8
b660 27 05                       BEQ UP
```

```
b662 4d                          TSTA
b663 2a 02                       BPL UP
b665 2b 05                       BMI DOWN
b667 7c 00 45           UP       INC STAPOS
b66a 20 05                       BRA OK
b66c 7a 00 45           DOWN     DEC STAPOS
b66f 20 00                       BRA OK
b671 96 44              OK       LDAA SWSHOE     GET CURRENT POSITION
b673 97 46                       STAA PREVSH     SAVE AS PREVIOUS
b675 81 04                       CMPA #$04       ON INDENT?
b677 26 4b                       BNE OUTPUT      NO, SO CONTINUE
                        * SWELL SHOE IS BACK ON A CLICK, SO SEE WHICH WAY IT WENT
                        * AND ADJUST THE VOLUME ACCORDINGLY
b679 96 45                       LDAA STAPOS
b67b 27 47                       BEQ  OUTPUT     JUST IGNORE IF NO CHANGE
b67d 7f 00 45                    CLR  STAPOS     CLEAR IT FOR NEXT
b680 4d                          TSTA
b681 2a 08                       BPL  INCVOL     INCREASE VOLUME IF +
                        * DECREASE VOLUME IF -, BUT NOT BELOW 0
b683 96 47              DECVOL   LDAA VOLUME
b685 4a                          DECA
b686 2a 0c                       BPL  VOLOK
b688 4f                          CLRA
b689 20 09                       BRA VOLOK
                        * INCREASE VOLUME IF +, BUT NOT ABOVE 20
b68b 96 47              INCVOL   LDAA VOLUME
b68d 4c                          INCA
b68e 81 14                       CMPA #20
b690 23 02                       BLS  VOLOK
b692 86 14                       LDAA #20
b694 97 47              VOLOK    STAA VOLUME
                        * NOW DOCTOR UP THE VOLUME TO DESIRED RANGE ~~
b696 96 47                       LDAA VOLUME     VALUE = 0 TO 20
b698 48                          LSLA
b699 48                          LSLA            VALUE = 0 TO 80
b69a 8b 1e                       ADDA #30
b69c 97 49                       STAA RSVOL      VALUE = 30 TO 110
                        * ALSO SET TIVOL AND BXVOL ~~
b69e 97 4a                       STAA BXVOL      FOR NOW USE THE SAME
                        * NEXT OUTPUT VIA SPI PORT TO TI CHIP - USE TIVOL ~~
                        * PUT RSVOL STUFF INTO CIRC BUFFER
b6a0 de 40                       LDX   BUFIN
b6a2 86 80                       LDAA  #$80   OUTPUT VIA RS232
b6a4 d6 49                       LDAB  RSVOL  VOLUME
b6a6 ed 00                       STD   0,X    PUT BOTH INTO BUFFER
b6a8 8d 3c                       BSR   INCIN  BUMP POINTER BY 2
                        * PUT BXVOL STUFF INTO CIRC BUFFER
b6aa de 40                       LDX   BUFIN
b6ac 86 00                       LDAA  #$00    OUTPUT VIA MIDI
b6ae d6 4b                       LDAB  BXCHAN  $B AND CHANNEL...
b6b0 ed 00                       STD   0,X     ...BOTH INTO BUFFER
b6b2 8d 32                       BSR   INCIN   BUMP POINTER BY 2
b6b4 de 40                       LDX   BUFIN
b6b6 c6 0b                       LDAB  #BXCONT CONTROLLER NUMBER...
b6b8 ed 00                       STD   0,X     ...INTO BUFFER
b6ba 8d 2a                       BSR   INCIN   BUMP POINTER BY 2
b6bc de 40                       LDX   BUFIN
b6be d6 4a                       LDAB  BXVOL   AND VOLUME...
b6c0 ed 00                       STD   0,X     ...INTO BUFFER
b6c2 8d 22                       BSR   INCIN   BUMP POINTER BY 2
                        * NOW OUTPUT FROM BUFFER IF NEEDED
b6c4 96 41              OUTPUT   LDAA  BUFIN+1
b6c6 91 43                       CMPA  BUFOUT+1
b6c8 26 03                       BNE   SKIP1   CONT IF NOT EMPTY
b6ca 7e b6 42                    JMP   MAINLP  ELSE REPEAT LOOP
b6cd f6 10 2e           SKIP1    LDAB  SCSR    READ STATUS
b6d0 c4 40                       ANDB  #$40    TRANSMIT COMPLETE FLAG
b6d2 26 03                       BNE   OUT2    OK IF XMTR FINISHED, ELSE...
```

```
b6d4 7e b6 42                    JMP   MAINLP  ...XMTR BUSY, SO REPEAT LOOP
b6d7 de 42              OUT2    LDX   BUFOUT
b6d9 ec 00                      LDD   0,X     GET TWO BYTES FROM BUFFER
b6db b7 10 00                   STAA  PORTA   CHOOSE RS232 IF H OR MIDI IF L
b6de f7 10 2f                   STAB  SCDAT   SEND CHARACTER
b6e1 8d 0c                      BSR   INCOUT  BUMP POINTER BY 2
                        * AND FINALLY RETURN INTO LOOP
b6e3 7e b6 42                   JMP  MAINLP
                        * INCREMENT BUFIN POINTER BY 2
b6e6 d6 41              INCIN   LDAB BUFIN+1
b6e8 5c                         INCB
b6e9 5c                         INCB
b6ea c4 3f                      ANDB #$3F     BUFFER SIZE = 64
b6ec d7 41                      STAB BUFIN+1
b6ee 39                         RTS
                        * INCREMENT BUFOUT POINTER BY 2
b6ef d6 43              INCOUT  LDAB BUFOUT+1
b6f1 5c                         INCB
b6f2 5c                         INCB
b6f3 c4 3f                      ANDB #$3F     BUFFER SIZE = 64
b6f5 d7 43                      STAB BUFOUT+1
b6f7 39                         RTS
b6f8 43 4f 50 59 52 49  COPRIT  FCC 'COPYRIGHT (c) 2006 BY '
47 48 54 20 28 63
29 20 32 30 30 36
20 42 59 20
b70e 50 45 54 45 52 20          FCC 'PETER A. STARK'
41 2e 20 53 54 41
52 4b
                                END
```

The above program is subject to change. Aside from the lack of PGA2310PA audio support -- which may not ever really be needed -- there may be some slight adjustments needed to provide the correct scaling for the MIDI controller output and velocity output. Right now, both of these settings range from a minimum of 30 to a maximum of 110, but this may need to be changed. See the sections identified with two tildes (~~) in the above listing.

A Note on the DIP Switch, and Jumpers

DIP switch S2 sets the channel number *minus 1*. That is, MIDI channels are numbered 1 through 16, but the binary code used is 0 through 15. For example, if I want to play on channel 13, I actually set S1 to 1100, which is the binary number for 12. The way the switches are wired, an open switch generates a 1 while a closed switch is a 0, so the four switches are set to open-open-closed-closed.

Jumpers: There are several header strips for jumpers. When using a 68HC11A1 that has Motorola's Buffalo debugger programmed into its ROM, P1, P2, and P4 determine the operating mode for the software, as follows:

1. To run Buffalo, place a shorting jumper on P2 and P4. This allows you to do various debugging functions via the serial port; if you have programmed the MD-1 software into the chip, you can do a manual jump to it at location B600.
2. To have Buffalo start, but then right away automatically jump to the MD-1 software at location B600, remove all jumpers. This is the normal operating mode once you have programmed the chip and installed the board into your organ.
3. To go into bootstrap mode, so you can program the chip via the serial port, place a shorting jumper on P1.

Location P7, right next to the DIP switch, is "for future use". It is an extra set of four inputs that can be used for other applications, if needed, but not used in this application. For example, the 4-pole DIP switch could be replaced with an 8-pole switch, with the extra four poles used for other input information.