```
/*================================================================================================
================

Top Box 10 Okay
Accessible Music Project code written by Philip Catterall 2018

 This work is licensed under a Creative Commons Attribution-ShareAlike 3.0
 Unported License (CC BY-SA 3.0) http://creativecommons.org/licenses/by-sa/3.0/

 THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
 AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
 LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
 OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
 THE SOFTWARE.

Functionality
=============
4 Percussion Sounds
8 Harminic Sounds
4 Selctable harmonic instruments ~done
Tone stretch facility. This has the effect of holding an atuator on from 0 - 2 seconds
It blocks th eother note at the same time

Timestretch Function.

At 0 secs tone activation is polyphonic.
Once the delay is added  the effect of holding down an actuator.
This extends the duration of continous tone instruments but has no effect on stringed type
instruments.
Other actuators are locked out until the delay releases the tone.

TBD
===
1: Add in remaining chords chords
2: Add in instrument select.
3; Add in time delay cut off. Same length as lockout.

Thoughts
========
If the actuator is held down on a wind type instrument note carries on sounding. Maybe set
an interupt tiemto shut it off?
Look at the use of milliseconds in the metronome software.
================================================================================================
================ */
//  DEFINITIONS
#include <SoftwareSerial.h>
SoftwareSerial mySerial(2, 3); // RX, TX
byte note = 0; //The MIDI note value to be played
byte resetMIDI = 4; //Tied to VS1053 Reset line
int  instrument = 0;
int prev_instrument = 0;
#define GM1 0
#define GM2 1

// === CHANNEL DEFINITIONS for the MIDI CONTROLLER
//Define channels on the SD50 NB #define used as these values are unlikly to change
#define midiChan1 0x90;//channel 1 This will be used for an instrument, say piano or organ
#define midiChan1 0x91;//channel 2 This will be used for an instrument, say piano or organ
#define midiChan1 0x92;//channel 3 This will be used for an instrument, say piano or organ
#define midiChan10 0x99;//channel 2 This will be used for  percussion

//Note Definitions
const byte C3n = 48; //Octave 3 freq 130.81  3Cn
const byte C3s = 49;
const byte D3n = 50;
const byte D3s = 51;
const byte E3n = 52;
const byte F3n = 53;
const byte F3s = 54;
const byte G3n = 55;
const byte G3s = 56;
const byte A3n = 57;
const byte A3s = 58;
const byte B3b = 58;
```

```
const byte B3n = 59;

const byte C4n = 60;//Octave 4 C freq 261.63 ~ MIDDLE "C"
const byte C4s = 61;
const byte D4n = 62;
const byte D4s = 63;
const byte E4n = 64;
const byte F4n = 65;
const byte F4s = 65;
const byte G4n = 67;
const byte G4s = 68;
const byte A4n = 69; //Octave 4 A freq 440.00  4An ~ MIDDLE "A"
const byte A4s = 70;
const byte B4b = 70;
const byte B4n = 71;


//Chords ~ 3 Note Versions, hence _3

const byte C_major_3[3] = {C4n, E4n, G4n}; //skt1
const byte A_minor_3[3] = {A3n, C4n, E4n}; //skt2
const byte F_major_3[3] = {F3n, A3n, C4n}; //skt3
const byte G_major_3[3] = {G3n, B3n, D4n}; //skt4
const byte D_minor_3[3] = {D4n, F4n, A4n}; //skt5
const byte E_minor_3[3] = {E4n, G4n, B4n}; //skt6
const byte B_flat_major_3[3] = {B3b, D4n, F4n}; //skt7
const byte E_major_3[3] = {E4n, G4s, B4n};//skt 8


//Jack socket mapping to Arduino Pins

const byte skt1 = 5;
const byte skt2 = 6;
const byte skt3 = 7;
const byte skt4 = 8;
const byte skt5 = 9;
const byte skt6 = 10;
const byte skt7 = 11;
const byte skt8 = 12;

const byte skt9 = 17;
const byte skt10 = 18;
const byte skt11 = 13; // Might need pull and resokder
const byte skt12 = 19;

int actuator1_value = 0;
int actuator2_value = 0;
int actuator3_value = 0;
int actuator4_value = 0;
int actuator5_value = 0;
int actuator6_value = 0;
int actuator7_value = 0;
int actuator8_value = 0;

int actuator9_value = 0;
int actuator10_value = 0;
int actuator11_value = 1;
int actuator12_value = 0;

int prev_actuator1_value = 0;
int prev_actuator2_value = 0;
int prev_actuator3_value = 0;
int prev_actuator4_value = 0;
int prev_actuator5_value = 0;
int prev_actuator6_value = 0;
int prev_actuator7_value = 0;
int prev_actuator8_value = 0;

int prev_actuator9_value = 0;
int prev_actuator10_value = 0;
int prev_actuator11_value = 1;
int prev_actuator12_value = 0;

//Instruments
int Piano1 = 0;
int Cello = 43;
```

```
int Harpsichord = 6;
int Organ1 = 17;
int Shakuhachi = 75; // ethnic flute
int Instrument_Select = 0;//default on piano;
int prev_Instrument_Select;

// ~~~ PGenera;l ercussion ~~~~~~~~~
int agogo = 113;
int melo_tom = 117;
int pitched_percussion =114;
int taiko = 116; // a type of drum
int steeldrum = 114;
int synth_drum = 118;
int tinkle_bell = 112;
int woodblock = 115;
int timpani = 47;

//Channel 10 Percussion
int AcousticBassDrum = 35;
int AcousticSnare = 38;
int LowTom = 43;
int HighTom = 45;
int MidTom = 47;
int SplashCymbal = 55;
int CowBell = 56;
int HighBongo = 60;
int LowBongo = 61;
int High_Agogo = 67;
int Low_Agago = 68;
int High_Woodblock = 76;
int Low_Woodblock = 77;
int OpenTriangle = 81;
int JingleBell = 83;
int BellTree = 84;


// ~~ Set up some default instruments
int instrument_type1 = Piano1; //intial set up to piano = 0
int instrument_type2 = Organ1;
int instrument_type3 = Harpsichord;
int instrument_type4 = Shakuhachi;
int prev_instrument_type1 = Piano1; //intial set up to piano = 0// 5Jun14 added so we only
send
int prev_instrument_type2 = Organ1; // out data on th emidi port when there is an chang
eof instrument
int prev_instrument_type3 = Harpsichord; // this is to stop th epotential overolad of
buffers on some MIDI syntghs
int prev_instrument_type4 = Shakuhachi; //

/*int percussion_type1 = taiko; // initila set up to wood block = 115
int percussion_type2 = melo_tom;
int percussion_type3 = synth_drum;
int percussion_type4 = timpani;

int prev_percussion_type1 = taiko; // initila set up to wood block = 115
int prev_percussion_type2 = melo_tom;
int prev_percussion_type3 = synth_drum;
int prev_percussion_type4 = timpani;
*/
//Controls
int InstSelectA0 = HIGH;
int InstSelectA1 = HIGH;
long SoundStretch = 0;
long TimeDelayMilliSeconds = 0;
long MaxDelayMilliSeconds = 2000;
//  midiSerial.write(192); // select channel 1
//  midiSerial.write(instrument_type1); // tell it what chordedinstrument


int velo_main = 0; // initially set at 0 to cut sound at startup
int velocity_0 = 0;
int velo_adj = 2; // chorded instruments are loader than percussion, causing distortion in
amp/speakers

void setup() {
```

```
pinMode(skt1,INPUT_PULLUP);//Chords
pinMode(skt2,INPUT_PULLUP);
pinMode(skt3,INPUT_PULLUP);
pinMode(skt4,INPUT_PULLUP);
pinMode(skt5,INPUT_PULLUP);
pinMode(skt6,INPUT_PULLUP);
pinMode(skt7,INPUT_PULLUP);
pinMode(skt8,INPUT_PULLUP);

pinMode(skt9,INPUT_PULLUP);// Percussion
pinMode(skt10,INPUT_PULLUP);
pinMode(skt11,INPUT);  //Pin 13
pinMode(skt12,INPUT_PULLUP);

pinMode(14,INPUT_PULLUP);// Inst select
pinMode(15,INPUT_PULLUP);

actuator1_value = 1; //NB HIGH is "OFF"
actuator2_value = 1;
actuator3_value = 1;
actuator4_value = 1;
actuator5_value = 1;
actuator6_value = 1;
actuator7_value = 1;
actuator8_value = 1;

actuator9_value = 1;
actuator10_value = 1;
actuator11_value = 0;
actuator12_value = 1;

prev_actuator1_value = 1;
prev_actuator2_value = 1;
prev_actuator3_value = 1;
prev_actuator4_value = 1;
prev_actuator5_value = 1;
prev_actuator6_value = 1;
prev_actuator7_value = 1;
prev_actuator8_value = 1;

prev_actuator9_value = 1;
prev_actuator10_value = 1;
prev_actuator11_value = 0;
prev_actuator12_value = 1;

delay(250);

  Serial.begin(9600);
  //Setup soft serial for MIDI control
  mySerial.begin(31250);
  //Reset the VS1053
  pinMode(resetMIDI, OUTPUT);
  digitalWrite(resetMIDI, LOW);
  delay(100);
  digitalWrite(resetMIDI, HIGH);
  delay(100);
  talkMIDI(0xB0, 0x07, 120); //0xB0 is channel message, set channel volume to near max
(127)
//  talkMIDI(0xB0, 0x07, 0);

}
// ================================ Main Program============================
void loop() {

 if (millis() < 1000)
  {
//     Serial.print(" Time Delay Works ");
//     Serial.print("  Timestamp in mSec = ");
//     Serial.print(millis());
//     Serial.println();
     delay(1000);
     velo_main = 125; //added in V10 to cut noise on sart up, velo_main initially 0
    } // Let things settle

scan_all_pins ();
```

```
TimeDelayMilliSeconds= (SoundStretch*2000)/1024;

//======== Instrument select =======


if (InstSelectA1 == LOW && InstSelectA0 == LOW)
            {instrument = Piano1;}

  else if (InstSelectA1 == LOW && InstSelectA0 == HIGH)
            {instrument = Organ1;}

  else if (InstSelectA1 == HIGH && InstSelectA0 == LOW)
             {instrument = Harpsichord;}

   else if (InstSelectA1 == HIGH && InstSelectA0 == HIGH)
             {instrument = Shakuhachi;}

   if (instrument != prev_instrument)// Only send instrument if there is a change
   {
    talkMIDI(0xC0, instrument, 0);
    talkMIDI(0xC1, instrument, 0);
    talkMIDI(0xC2, instrument, 0);
    prev_instrument = instrument;
    //Set instrument number. 0xC0 is a 1 data byte command
   }

// ==== Play chords

 // ======== Begin  ========= 8 Chord Xmit Routines ==================================

  //============================ C CHORD  pin 5 etc ================================

    if (prev_actuator1_value == HIGH && actuator1_value == HIGH)  //ast two percussion
  {
    prev_actuator1_value = HIGH;
  }

  else if (prev_actuator1_value == HIGH && actuator1_value == LOW)  //
  {
 //        Serial.println();
//  Serial.print( " C major sent ON");
//  Serial.println();
//      Serial.println();
    noteOn(0x90, C_major_3[0], velo_main/velo_adj);
  //  delay(10);
    noteOn(0x91, C_major_3[1], velo_main/velo_adj);
  //  delay(5);
    noteOn(0x92, C_major_3[2], velo_main/velo_adj);
  //  delay(10);
    prev_actuator1_value = LOW ;
     delay(TimeDelayMilliSeconds);
/*     Serial.print("TimeDelayMilliSeconds = ");
     Serial.print(TimeDelayMilliSeconds);
     Serial.print("   ");
     Serial.print("SoundStretch = ");
     Serial.print(SoundStretch);
     Serial.println();
  */
     //delay(500);
  }

 else if (prev_actuator1_value == LOW && actuator1_value == HIGH)  //ast two percussion
  {
 //      Serial.println();
 //  Serial.print( " C major sent OFF");
 //  Serial.println();
 //     Serial.println();
    noteOn(0x90, C_major_3[0], velocity_0);
    noteOn(0x91, C_major_3[1], velocity_0);
    noteOn(0x92, C_major_3[2], velocity_0);

    prev_actuator1_value = actuator1_value;
   }

  else if (prev_actuator1_value == LOW && actuator1_value == LOW)  //ast two percussion
  {
```

```
    prev_actuator1_value = actuator1_value;
  }
  //======================END OF ======= C CHORD
================================================

 // ====================
    if (prev_actuator2_value == HIGH && actuator2_value == HIGH)  //ast two percussion
  {
    prev_actuator2_value = HIGH;
  }

  else if (prev_actuator2_value == HIGH && actuator2_value == LOW)  //
  {
 //   Serial.print( " A minor sent ON");
 //  Serial.println();
 //      Serial.println();
    noteOn(0x90, A_minor_3[0], (velo_main*0.8)/velo_adj);
  //  delay(10);
    noteOn(0x91,  A_minor_3[1], velo_main/velo_adj);
  //  delay(5);
    noteOn(0x92,  A_minor_3[2], velo_main/velo_adj);
  //  delay(10);
/*     delay(TimeDelayMilliSeconds);
     Serial.print("TimeDelayMilliSeconds = ");
     Serial.print(TimeDelayMilliSeconds);
     Serial.print("   ");
     Serial.print("SoundStretch = ");
     Serial.print(SoundStretch);
     Serial.println();
  */
    prev_actuator2_value = actuator2_value ;
  }

 else if (prev_actuator2_value == LOW && actuator2_value == HIGH)  //ast two percussion
  {
 //   Serial.print( " A minor sent OFF");
 //   Serial.println();
 //      Serial.println();
    noteOn(0x90, A_minor_3[0], velocity_0/velo_adj);
    noteOn(0x91, A_minor_3[1], velocity_0/velo_adj);
    noteOn(0x92, A_minor_3[2], velocity_0/velo_adj);

 //   Serial.print( "A minor   Chord OFF ");
 //   Serial.print("  Timestamp in mSec = ");
 //    Serial.print(millis());
  //  Serial.println();


    prev_actuator2_value = actuator2_value;
   }

  else if (prev_actuator2_value == LOW && actuator2_value == LOW)  //ast two percussion
  {
    prev_actuator2_value = actuator2_value;
  }

//================ F Major ==========================

    if (prev_actuator3_value == HIGH && actuator3_value == HIGH)  //ast two percussion
  {
    prev_actuator3_value = HIGH;
  }

  else if (prev_actuator3_value == HIGH && actuator3_value == LOW)  //
  {
    noteOn(0x90, F_major_3[0], velo_main/velo_adj);
 //   delay(10);
    noteOn(0x91, F_major_3[1], velo_main/velo_adj);
 //   delay(5);
    noteOn(0x92, F_major_3[2], velo_main/velo_adj);
 //   delay(10);
     delay(TimeDelayMilliSeconds);
     prev_actuator3_value = actuator3_value ;
  }

 else if (prev_actuator3_value == LOW && actuator3_value == HIGH)  //ast two percussion
```

```
  {
    noteOn(0x90, F_major_3[0], velocity_0);
    noteOn(0x91, F_major_3[1], velocity_0);
    noteOn(0x92, F_major_3[2], velocity_0);

    prev_actuator3_value = actuator3_value;
   }

  else if (prev_actuator3_value == LOW && actuator3_value == LOW)  //ast two percussion
  {
    prev_actuator3_value = actuator3_value;
  }

//================== G major ============================

   if (prev_actuator4_value == HIGH && actuator4_value == HIGH)  //ast two percussion
  { prev_actuator4_value = HIGH;  }

  else if (prev_actuator4_value == HIGH && actuator4_value == LOW)  //
  {
    noteOn(0x90,G_major_3[0], velo_main/velo_adj);  //x90 is channel 1
  //  delay(10);
    noteOn(0x91, G_major_3[1], velo_main/velo_adj);
  //  delay(5);
    noteOn(0x92, G_major_3[2], velo_main/velo_adj);
  //  delay(10);
    delay(TimeDelayMilliSeconds);
    prev_actuator4_value = actuator4_value ;
  }

 else if (prev_actuator4_value == LOW && actuator4_value == HIGH)  //ast two percussion
  {
    noteOn(0x90, G_major_3[0], velocity_0);
    noteOn(0x91, G_major_3[1], velocity_0);
    noteOn(0x92, G_major_3[2], velocity_0);
    prev_actuator4_value = actuator4_value;
   }
  else if (prev_actuator4_value == LOW && actuator4_value == LOW)  //ast two percussion
  {  prev_actuator4_value = actuator4_value;  }

//======================== D minor ============================

   if (prev_actuator5_value == HIGH && actuator5_value == HIGH)  //ast two percussion
  { prev_actuator5_value = HIGH;  }

  else if (prev_actuator5_value == HIGH && actuator5_value == LOW)  //
  {
    noteOn(0x90,D_minor_3[0], velo_main/velo_adj);  //x90 is channel 1
  //  delay(10);
    noteOn(0x91, D_minor_3[1], velo_main/velo_adj);
  //  delay(5);
    noteOn(0x92, D_minor_3[2], velo_main/velo_adj);
  //  delay(10);
    delay(TimeDelayMilliSeconds);
    prev_actuator5_value = actuator5_value ;
  }
 else if (prev_actuator5_value == LOW && actuator5_value == HIGH)  //ast two percussion
  {
    noteOn(0x90, D_minor_3[0], velocity_0);
    noteOn(0x91, D_minor_3[1], velocity_0);
    noteOn(0x92, D_minor_3[2], velocity_0);
    prev_actuator5_value = actuator5_value;
   }
  else if (prev_actuator5_value == LOW && actuator5_value == LOW)  //ast two percussion
  {  prev_actuator5_value = actuator5_value;  }

// ========================== E minor ====================

   if (prev_actuator6_value == HIGH && actuator6_value == HIGH)  //ast two percussion
  { prev_actuator6_value = HIGH;  }
  else if (prev_actuator6_value == HIGH && actuator6_value == LOW)  //
  {
    noteOn(0x90, E_minor_3[0], velo_main/velo_adj);  //x90 is channel 1
  //  delay(10);
    noteOn(0x91, E_minor_3[1], velo_main/velo_adj);
  //  delay(5);
```

```
    noteOn(0x92, E_minor_3[2], velo_main/velo_adj);
  //  delay(10);
    delay(TimeDelayMilliSeconds);
    prev_actuator6_value = actuator6_value ;
  }
 else if (prev_actuator6_value == LOW && actuator6_value == HIGH)  //ast two percussion
  {
    noteOn(0x90, E_minor_3[0], velocity_0);
    noteOn(0x91, E_minor_3[1], velocity_0);
    noteOn(0x92, E_minor_3[2], velocity_0);
    prev_actuator6_value = actuator6_value;
   }
  else if (prev_actuator6_value == LOW && actuator6_value == LOW)  //ast two percussion
  {  prev_actuator6_value = actuator6_value;  }

//============================== B Flat -Major=====================

   if (prev_actuator7_value == HIGH && actuator7_value == HIGH)  //ast two percussion
  { prev_actuator7_value = HIGH;  }
  else if (prev_actuator7_value == HIGH && actuator7_value == LOW)  //
  {
    noteOn(0x90, B_flat_major_3[0], velo_main/velo_adj);  //x90 is channel 1
  //  delay(10);
    noteOn(0x91, B_flat_major_3[1], velo_main/velo_adj);
  //  delay(5);
    noteOn(0x92, B_flat_major_3[2], velo_main/velo_adj);
  //  delay(10);
    delay(TimeDelayMilliSeconds);
    prev_actuator7_value = LOW ;
  }
 else if (prev_actuator7_value == LOW && actuator7_value == HIGH)  //ast two percussion
  {
    noteOn(0x90, B_flat_major_3[0], velocity_0);
    noteOn(0x91, B_flat_major_3[1], velocity_0);
    noteOn(0x92, B_flat_major_3[2], velocity_0);
    delay(TimeDelayMilliSeconds);

    prev_actuator7_value = HIGH;
   }
  else if (prev_actuator7_value == LOW && actuator7_value == LOW)  //ast two percussion
  {  prev_actuator7_value = LOW;  }

//============================== E major ==========================

   if (prev_actuator8_value == HIGH && actuator8_value == HIGH)  //ast two percussion
  { prev_actuator8_value = HIGH;  }
  else if (prev_actuator8_value == HIGH && actuator8_value == LOW)  //
  {
    noteOn(0x90, E_major_3[0], velo_main/velo_adj);  //x90 is channel 1
   // delay(10);
    noteOn(0x91, E_major_3[1], velo_main/velo_adj);  // notes on sepeate channels, else
tone stretch emphesis last note on a channel
   // delay(5);
    noteOn(0x92, E_major_3[2], velo_main/velo_adj);
   // delay(10);
    delay(TimeDelayMilliSeconds);
    prev_actuator8_value = actuator8_value ;
  }
 else if (prev_actuator8_value == LOW && actuator8_value == HIGH)  //ast two percussion
  {
    noteOn(0x90, E_major_3[0], velocity_0);
    noteOn(0x91, E_major_3[1], velocity_0);
    noteOn(0x92, E_major_3[2], velocity_0);
    prev_actuator8_value = actuator8_value;
   }
  else if (prev_actuator8_value == LOW && actuator8_value == LOW)  //ast two percussion
  {  prev_actuator8_value = actuator8_value;  }

// ============================== percussion routine  ==============
// ~~~ Percussion ~~~~~~~~~

// ========== Perc 1 ============================

  if (prev_actuator9_value == HIGH && actuator9_value == HIGH)  //ast two percussion
  {  prev_actuator9_value = HIGH; }
  else if (prev_actuator9_value == HIGH && actuator9_value == LOW)  //
```

```
  {
    noteOn(0x99,Low_Woodblock , velo_main);  //x99 is channel 10
    delay(TimeDelayMilliSeconds);
    prev_actuator9_value = actuator9_value ;
  }
 else if (prev_actuator9_value == LOW && actuator9_value == HIGH)
  {  noteOn(0x99, Low_Woodblock , velocity_0);
    prev_actuator9_value = actuator9_value;     }
  else if (prev_actuator9_value == LOW && actuator9_value == LOW)  //ast two percussion
  { prev_actuator9_value = actuator9_value; }

//========================== Perc2 ==========================

  if (prev_actuator10_value == HIGH && actuator10_value == HIGH)  //ast two percussion
  {  prev_actuator10_value = HIGH; }
  else if (prev_actuator10_value == HIGH && actuator10_value == LOW)  //
  {
    noteOn(0x99,High_Woodblock  , velo_main);  //x99 is channel 10
    delay(TimeDelayMilliSeconds);
    prev_actuator10_value = actuator10_value ;
  }
 else if (prev_actuator10_value == LOW && actuator10_value == HIGH)
  {  noteOn(0x99, High_Woodblock , velocity_0);
    prev_actuator10_value = actuator10_value;     }
  else if (prev_actuator10_value == LOW && actuator10_value == LOW)  //ast two percussion
  { prev_actuator10_value = actuator10_value; }

//========================== Perc 3 ==========================

  if (prev_actuator11_value == LOW && actuator11_value == LOW)  //ast two percussion
  {  prev_actuator11_value = LOW; }

  else if (prev_actuator11_value == LOW && actuator11_value == HIGH)  //
  {
    noteOn(0x99, JingleBell  , velo_main);
        delay(TimeDelayMilliSeconds);
    prev_actuator11_value = HIGH;
  }
 else if (prev_actuator11_value == HIGH && actuator11_value == LOW)

  {
   noteOn(0x99, JingleBell  , velocity_0);  //x99 is channel 10
    prev_actuator11_value = LOW ;
   }

  else if (prev_actuator11_value == HIGH && actuator11_value == HIGH)  //ast two
percussion
  { prev_actuator11_value = HIGH; }

//========================== Perc 4 ==========================

  if (prev_actuator12_value == HIGH && actuator12_value == HIGH)  //ast two percussion
  {  prev_actuator12_value = HIGH; }
  else if (prev_actuator12_value == HIGH && actuator12_value == LOW)  //
  {
    noteOn(0x99, HighBongo , velo_main);  //x99 is channel 10
    delay(TimeDelayMilliSeconds);
    prev_actuator12_value = actuator12_value ;
  }
 else if (prev_actuator12_value == LOW && actuator12_value == HIGH)
  {  noteOn(0x99, HighBongo , velocity_0);
    prev_actuator12_value = actuator12_value;     }
  else if (prev_actuator12_value == LOW && actuator12_value == LOW)  //ast two percussion
  { prev_actuator12_value = actuator12_value; }
}

//========================== Sub routines ==================================

void scan_all_pins () // get value of glable variables
// rewrite this in context of top box, include analogue
{
actuator1_value = digitalRead(skt1);
actuator2_value = digitalRead(skt2);
actuator3_value = digitalRead(skt3);
actuator4_value = digitalRead(skt4);
actuator5_value = digitalRead(skt5);
```

```
actuator6_value = digitalRead(skt6);
actuator7_value = digitalRead(skt7);
actuator8_value = digitalRead(skt8);

actuator9_value = digitalRead(skt9);//          *
actuator10_value = digitalRead(skt10);//             *
actuator11_value = digitalRead(skt11);//         *    NB Pin 13

//Serial.print("Skt11_value = ");
//Serial.print(skt11);
//Serial.print("  actuator11_value = ");
//Serial.print(actuator11_value);
//Serial.println();
//delay(500);
actuator12_value = digitalRead(skt12); //   *

InstSelectA0 = digitalRead(14);
InstSelectA1 = digitalRead(15);
SoundStretch = analogRead(2);
}


// ============  MIDI Routines ===============
//Send a MIDI note-on message.  Like pressing a piano key
//channel ranges from 0-15
void noteOn(byte channel, byte note, byte attack_velocity) {
  talkMIDI( (0x90 | channel), note, attack_velocity);
/*    Serial.println();  // nb leaving print diagnostc in add approc 100ms latency between
notes
    Serial.print("  Channel = ");
    Serial.print( channel, HEX);
    Serial.print("  MIDI note = ");
    Serial.print( note);
    Serial.print("  Velocity = ");
    Serial.print( attack_velocity);
    Serial.println();
*/
 }

//Send a MIDI note-off message.  Like releasing a piano key
void noteOff(byte channel, byte note, byte release_velocity) {
  talkMIDI( (0x80 | channel), note, release_velocity);
}

//Plays a MIDI note. Doesn't check to see that cmd is greater than 127, or that data
values are less than 127
void talkMIDI(byte cmd, byte data1, byte data2) {
  mySerial.write(cmd);
  mySerial.write(data1);
/*
  Serial.print(" MIDIxmit  accessed = ");
  Serial.print( "   Data1 = ");
  Serial.print(data1);
  Serial.print("  Timestamp in mSec = ");
  Serial.print(millis());
  Serial.println();
*/

  //Some commands only have one data byte. All cmds less than 0xBn have 2 data bytes
  //(sort of: http://253.ccarh.org/handout/midiprotocol/)
  if( (cmd & 0xF0) <= 0xB0)
    mySerial.write(data2);
}
```