/* ~~~~~~~~~~~~~~~~ MIDI Chord and Percusion Player
~~~~~~

==========TO BE DONE ==========

9th May 014: Main and preset volum control added

1: Change i/p selct log to a do whilw to get rid off loop strop.
2: DONE:Create initilisation to set up instruments under software control Do in seperat skecth to test Ext SD50 control
3: DONE Put instrument select under external switch control ~ Using resitive ladder
4: Idea, have different chord sets that can be switched in, dependent on music score selected.
5: Needs an external reset button.
6: Done Jus t use a switch to give two sets of percussion fo rth em but imorved with rotar switch to give thre sest of 3
7: Use the second rotary switch to give alternaticve chord sets or maybe between mon and polyphonic.
8: Process use two diff for next lops (3, or 6 to differentiayt ebetween 3 note and 6 note chords
9: Dead / redundant code yet to be removed 22Aug13
10: Addin extra chords
========== version control

Box4 V01 move over to 6 strings, use of new note identifiers, volume tuning added to individual notes to balance loudness.
Box 4 V03 sort ot percussion
ba_Box4v03dot6: All chord now done a one dimensinalarary, in addition each channel has a changabl evelocity setting to deal with unsounded strings, e,g, Sttring 6 in Dmajor.

aaMbox4c3p_1dot4_OK ~ Chord and percussion select moved over to DigiPin[] arrays.

*/

// ========== DECLARE GLOBAL VARIABLRS ==========

//midi sheild stuff
#include <SoftwareSerial.h>

byte resetMIDI = 4;/// Change - 9Nov13 A0;//A5//change to A5 from A0 14Oct12
SoftwareSerial midiSerial(2, 3); //midiSerial(2, 3)// MIDI out put tp Seeduino sheild
#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 midiChan2 0x91;//channel 2 This will be used for P1 percussion
#define midiChan3 0x92;//channel 3 This will be used for P2 percussion
#define midiChan4 0x93;//channel 4 This will be used for P3 percussion
#define midiChan5 0x94;//channel 5 This will be used for P4 percussion
#define midiChan6 0x95;//channel 5 This will be used for Metronome / Rate Tick
#define midiChan7 0x96;//channel 5 This will be used for Appluase
//Box4 new note definitions to match MIDI standrads (Major chord notes for the Mo
~~~~~~

```
//NB need to ad in flats for minor keys


//=============== NEW NOTE DEFINITIONS Defs 1st July 13

const byte note0 = 0;  //Octave -5 C midi note no 0 / freq   8.18  m1Cn
const byte note39 =39; //Octave 2 D# midi note no 39/ freq  77.78  2Ds
const byte note40 =40; //Octave 2 E  midi note no 40/ freq  82.41  2En
const byte note41 =41; //Octave 2 F  midi note no 41/ freq  87.31  2Fn
const byte note42 =42; //Octave 2 F# midi note no 42/ freq  92.50  2Fs
const byte note43 =43; //Octave 2 G  midi note no 43/ freq  98.00  2Gn
const byte note44 =44; //Octave 2 G# midi note no 44/ freq 103.83  2Gs
const byte note45 =45; //Octave 2 A  midi note no 45/ freq 110.00  2An
const byte note46 =46; //Octave 2 A# midi note no 46/ freq 116.54  2As
const byte note47 =47; //Octave 2 B  midi note no 47/ freq 123.47  2Bn

const byte note48 =48; //Octave 3 C  midi note no 48/ freq 130.81  3Cn
const byte note49 =49; //Octave 3 C# midi note no 49/ freq 138.59  3Cs
const byte note50 =50; //Octave 3 D  midi note no 50/ freq 146.83  3Dn
const byte note51 =51; //Octave 3 D# midi note no 51/ freq 155.56  3Ds
const byte note52 =52; //Octave 3 E  midi note no 52/ freq 164.81  3En ( F3 )
const byte note53 =53; //Octave 3 F  midi note no 53/ freq 174.61  3Fn
const byte note54 =54; //Octave 3 F# midi note no 54/ freq 185.00  3Fs
const byte note55 =55; //Octave 3 G  midi note no 55/ freq 196.00  3Gn
const byte note56 =56; //Octave 3 G# midi note no 56/ freq 207.66  3Gs
const byte note57 =57; //Octave 3 A  midi note no 57/ freq 220.00  3An
const byte note58 =58; //Octave 3 A# midi note no 58/ freq 233.08  3As
const byte note59 =59; //Octave 3 B  midi note no 59/ freq 246.94  3Bn

const byte note60 =60; //Octave 4 C  midi note no 60/ freq 261.63  4Cn ~ MIDDLE "C"
const byte note61 =61; //Octave 4 C# midi note no 61/ freq 277.18  4Cs
const byte note62 =62; //Octave 4 D  midi note no 62/ freq 293.66  4Dn
const byte note63 =63; //Octave 4 D# midi note no 63/ freq 311.13  4Ds
const byte note64 =64; //Octave 4 E  midi note no 64/ freq 329.63  4En
//const byte E4n = 64;
const byte note65 =65; //Octave 4 F  midi note no 65/ freq 349.23  4Fn
const byte note66 =66; //Octave 4 F# midi note no 66/ freq 369.99  4Fs
const byte note67 =67; //Octave 4 G  midi note no 67/ freq 392.00  4Gn
const byte note68 =68; //Octave 4 G# midi note no 68/ freq 415.30  4Gs
const byte note69 =69; //Octave 4 A  midi note no 69/ freq 440.00  4An ~ MIDDLE "A"
const byte note70 =70; //Octave 4 A# midi note no 70/ freq 446.16  4As
const byte note71 =71; //Octave 4 B  midi note no 71/ freq 493.88  4Bn

const byte Guitar_Chord_A[6] = {note64, note61, note57 , note52, note45, note40 }; // E4n C4s A3n E3n A2n E2n
const byte Guitar_Chord_E[6] = {note64, note59, note56 , note52, note47, note40 }; // E4n B3n G3s E3n B2n E2n
const byte Guitar_Chord_D[6] = {note66, note62, note57 , note50, note45 };         // F4s D4n A3n D3n A2n

const byte Guitar_Chord_G[6] = {note67, note59, note55 , note50, note47, note43 }; // G4n B3n G3n D3n B2n G2n
const byte Guitar_Chord_A7[6] = {note64, note61, note55 , note52, note45, note40 }; //E4n C4s G3n E3n A2n E2n vel
0
const byte Guitar_Chord_C[6] = {note64, note60, note55 , note52, note48, note40 }; // E4n C4n G3n E3n C3n E2n vel
0

const byte Guitar_Chord_B7[6] = {note66, note59, note57 , note51, note47, note40 };// F4s B3n A3n  D3s  B2n E2n
```

```
const byte Guitar_Chord_D7[6] = {note66, note60, note57 , note51, note45, note40 }; // F4s C4n A3n D3s A2n 0vel E2 0 vel

// tghis does not look correct ~const byte Guitar_Chord_Am[6] = {note64, note61, note57 , note52, note45, note40 };//

//const byte notes[6] = {64, 61, 57, 52,45, 40}; this was used befoer Guitar Chrd array
// == DEFINE USE of INPUTS ~ cahnge these to more meaningful names later

//const byte BUTTON2 = 2; // Assigned to instrument selection 00;01;10,11
//const byte BUTTON3 = 3; // Assigned to instrument selection

//const byte BUTTON4 = 4; // Assigned for Chord A //percussion selection 00;01;10,11
const byte BUTTON5 = 5; // Assigned for Chord E//percussion selection
const byte BUTTON6 = 6; // Chord G / input button for 1st Chord (A fro the mo)
const byte BUTTON7 = 7; // Chord D /input button for 2nd chord (E for the mo)
const byte BUTTON8 = 8; // input button for 3rd chord (D

const byte BUTTON9 = 9; // input button for 4th chord (G
const byte BUTTON10 = 10; //
const byte BUTTON11 = 11; //
const byte BUTTON12 = 12; //

const byte BUTTON13 = 13; // input button for bass beat
const byte BUTTON14 = 14; // A0 input button for bass beat
const byte BUTTON15 = 15; // A1 input button for bass beat
const byte BUTTON16 = 16; // A2 input button for bass beat
//const byte BUTTON17 = 17;// A3 // input button for bass beat
//const byte BUTTON18 = 18; // input button for bass beat
//const byte BUTTON19 = 19; // input button for bass beat

//== Setup6 individual chord/notes to middle C keyboard range
int chordnote1 = 0x3C;
int chordnote2 = 0x3C;
int chordnote3 = 0x3C;
int chordnote4 = 0x3C;
int chordnote5 = 0x3C;
int chordnote6 = 0x3C;

// Initialies drum "note"
int percussion_note1 = 0x45; // notes needed if say a glocken spiel
int percussion_note2 = 0x50;  // might need to send note appropriate fro the instrument
int percussion_note3 = 0x55;
int percussion_note4 = 0x3C;
int percussion_note5 = 0x3C;
int percussion_note6 = 0x3C;

int velocity = 45; //set up a default velocity
int velocity01 = 100; // These are now used to decide if a not eis actually sent eg.D chord does not sound the 6th striug
int velocity02 = 100; // intiall set with a defalt velocity value of 100
int velocity03 = 100;
int velocity04 = 100;
int velocity05 = 100;
int velocity06 = 100;
int velocity_0 = 0;
```

```
int velocity_25 = 25;
int velocity_50 = 50;
int velocity_75 = 75;
int velocity_100 = 100;
int velocity_125 = 125;

int velo_percchordA = 125;
int velo_percchordE = 125;
int velo_percchordD = 125;
int velo_percchordG = 125;
int velo_percchordC = 125;
int velo_percchordDm = 125;

int velo_perc11 = 125; //default drum velocities
int velo_perc12 = 125;
int velo_perc13 = 125;
int velo_perc10 = 125;

// Digital volume and pan controls

int velo_pan_left = 0;
int velo_pan_right = 0;
int velo_3str_chord = 0;//piano type chord
int velo_6str_chord = 0;//string instr type chord

// DIGITAL VOLUME
double velo_main = 0.6;//set at this level to stop D class AMPs from "tripping"
double logVol = 0.999;
double velo_preset = 0.999; // this is to peak limit an amplifier// Especiall LEPAI T-Class

// Percussion Xfer

int P1_count = 0;
int P3_count = 0;
int P2_count = 0;
int P4_count = 0;


int test_inst = 120;
// ~~ Ployphonic instruments  ~ nned to distinguish between  3 chord piano and 6 string guitar
int acoustic_bass = 32;
int cello = 43;
int choir_aahs = 52;
int cleanguitar = 27;
int harpsichord = 6;
int marimba = 12;
int nylonguitar1 = 24;
int organ1 = 17;
int organ3 = 18;
int piano1 = 0;  // Use PC values from SD50 instrument list ( minus 1).
int piccolo = 72;
int slow_strings = 49; // ensemble strings
int strings = 48;
int tubular_bells = 14;
```

```
// ~~~ Monophonic ~~~~
int bassoon = 70;
int shakuhachi = 75; // ethnic flute

// ~~~ Percussion ~~~~~~~~~~
int agogo = 113;
int melo_tom = 117;
int pitched_percussion =114;
int taiko = 116; // a type of drum
int reverse_cymbal = 119;
int steeldrum = 114;
int synth_drum = 118;
int tinkle_bell = 112;
int woodblock = 115;
int timpani = 47;

// Misc
int applause = 126;

// ~~ Set up some default instruments
int instrument_type1 = piano1; //intial set up to piano = 0
int instrument_type2 = organ1;
int instrument_type3 = nylonguitar1;
int instrument_type4 = slow_strings;
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 = nylonguitar1; // this is to stop th epotential overolad of buffers on some MIDI syntghs
int prev_instrument_type4 = slow_strings; // also facilitates dat alogiing by not send too much "stuff".

//percussion groups 3 of 3.
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;

int acoustic_grand_piano = 1;
int bright_acoustic_piano = 2;
int electric_grand_piano = 3;

int celesta = 9;
int music_box = 11;
int acoustic_guitar_steel = 26;
int electric_guitar_jazz = 27;
int electric_guitar_clean = 28;
int citar = 105;

int valPercSelect = 0;
int valPercSelectPrev = 0;
```

```
int valInstSelect = 0; //Variable for resister ladder network for instrumet slect
int valRange = 0;
int valPercRange = 0;
// int testnote = 0x48;
// intitiliase start values of input pins
//int valbutton02 = LOW; // butons to be pressed
//int valbutton03 = LOW;

//int valbutton04 = HIGH;
int valbutton05 = HIGH;
int valbutton06 = HIGH;
int valbutton07 = HIGH;
int valbutton08 = HIGH;

int valbutton09 = HIGH;
int valbutton10 = HIGH;
int valbutton11 = HIGH;
int valbutton12 = HIGH;

int valbutton13 = HIGH;
int valbutton14 = HIGH;
int valbutton15 = HIGH;
int valbutton16 = HIGH; //Using Analogue buttons A0, A1, A2 as digital buttonns for percussion.
//int valbutton17 = HIGH;
//int valbutton18 = HIGH;
//int valbutton19 = HIGH;

int playnotes = LOW; //for checking if instruments or percussion or being pressed
int any_key_pressed = HIGH; // check if any peddle or buttomn etc pressed
int play_poly_notes = HIGH; // play selected chord
int play_percussion_notes = HIGH;

int play_perc_note1 = HIGH; // play percussion inst 1
int play_perc_note2 = HIGH; // play percussion inst 2
int play_perc_note3 = HIGH;
int play_perc_note4 = HIGH;

byte midiXmitFlag = 0; //used to enable / disable transmit of midi signals to external device
byte midiXmitNoteFlag = 0;
byte midiXmitPercFlag = 0;

// DigiPins new routine to move towards use of arrays for eventula use in datalogging.
// also hopefully solve the "blocking chord" problem 14Jul13
static byte DigiPins[19];
static byte prevDigiPins[19];
static int chord_note_vels [128]; //Added20Jul13
byte channel_num = 0; // channel number for us eon synth range 0 to 15 dec, 0 to F in Hex




//============== SETUPS ================================================
void setup() {
//== INTIALISE Communication with SD50 and assign initial channel voices
```

```
  Serial.begin(9600); // setup serialchannel for time stamping
  midiSerial.begin(31250);
  pinMode(resetMIDI, OUTPUT);
  digitalWrite(resetMIDI, LOW);
  delay(100);
  digitalWrite(resetMIDI, HIGH);
  delay(100);

//default instruments

//  Serial.begin(31250);  // Set up MIDI baud rate
  midiSerial.write(192); // select channel 1
  midiSerial.write(instrument_type1); // tell it what chordedinstrument

  midiSerial.write(193); // select channel 2
  midiSerial.write(percussion_type1); // tell it 1st percussion
  midiSerial.write(194); // select channel 3
  midiSerial.write(percussion_type2); // tell it 2nd percussion
  midiSerial.write(195); // select channel 4
  midiSerial.write(percussion_type3);
  midiSerial.write(196); // select channel 5
  midiSerial.write(percussion_type4);
// CANT USED 5 PIN MIDI output in normal use as serial output is required via USB for data logging

// test copy to SD-50 to determine inst select ~ extended bass use of.
//  Serial.begin(31250);  // Set up MIDI baud rate
//  Serial.write(192); // select channel 1
//  Serial.write(instrument_type1); // tell it what chordedinstrument

//  Serial.write(193); // select channel 2
//  Serial.write(percussion_type1); // tell it 1st percussion
//  Serial.write(194); // select channel 3
//  Serial.write(percussion_type2); // tell it 2nd percussion
//  Serial.write(195); // select channel 4
//  Serial.write(percussion_type3);
//  Serial.write(196); // select channel 5
//  Serial.write(percussion_type4);
//================================


  // ===== SET UP INPUTS ~ NB Digital Pins 0 and 1 are used for Tx and Rx
//  pinMode(BUTTON2, INPUT);//select percussion
//  pinMode(BUTTON3, INPUT);// This is now used bytheresety on th enew soaftware!!!!!!!!!!!
//  pinMode(BUTTON4, INPUT);//select instrument
  pinMode(BUTTON5, INPUT);//
  pinMode(BUTTON6, INPUT);//select chords or up to 8 notes
  pinMode(BUTTON7, INPUT);//select chards
  pinMode(BUTTON8, INPUT);//

  pinMode(BUTTON9, INPUT);//
  pinMode(BUTTON10, INPUT);//
  pinMode(BUTTON11, INPUT);// percussion instrument 0
  pinMode(BUTTON12, INPUT);// percussion instrument 1
```

```
  pinMode(BUTTON13, INPUT);// percussion instrument 2
  pinMode(BUTTON14, INPUT);// equates to Analogue A0 used to move mide pin no over back to A0 for MIDI reset
17Oct13
  pinMode(BUTTON15, INPUT);// equates to Analogue A1 used to move mide pin no over
  pinMode(BUTTON16, INPUT);// equates to Analogue A3 used to move mide pin no over
//pinMode(BUTTON17, INPUT);// equates to Analogue A4 used to move mide pin no over
//pinMode(BUTTON18, INPUT);// equates to Analogue A5 used to move mide pin no over
//pinMode(BUTTON19, INPUT);// equates to Analogue A4 used to move mide pin no over


 // digitalWrite(BUTTON4, HIGH);
  digitalWrite(BUTTON5, HIGH);
  digitalWrite(BUTTON6, HIGH);
  digitalWrite(BUTTON7, HIGH);
  digitalWrite(BUTTON8, HIGH);

  digitalWrite(BUTTON9, HIGH);
  digitalWrite(BUTTON10, HIGH);
  digitalWrite(BUTTON11, HIGH);
  digitalWrite(BUTTON12, HIGH);

  digitalWrite(BUTTON13, HIGH);
  digitalWrite(BUTTON14, HIGH);//A0
  digitalWrite(BUTTON15, HIGH);//A1
  digitalWrite(BUTTON16, HIGH);//A2
 delay(500);// give time to set up

  //INITILIAISE TONES AND PERCUSSION OFF
clearNotes_Ch01 (chordnote1, chordnote2, chordnote3, chordnote4, chordnote5, chordnote6 );
clearNotes_Ch02 (chordnote1, chordnote2, chordnote3, chordnote4, chordnote5, chordnote6 );
clearNotes_Ch03 (chordnote1, chordnote2, chordnote3, chordnote4, chordnote5, chordnote6 );
clearNotes_Ch04 (chordnote1, chordnote2, chordnote3, chordnote4, chordnote5, chordnote6 );
clearNotes_Ch05 (chordnote1, chordnote2, chordnote3, chordnote4, chordnote5, chordnote6 );
  // Set all note velocities to 0
  for (int i=0; i<127; i++)
      { chord_note_vels [i] = 0; };

 for (int i=0; i<19; i++)
      {
        DigiPins [i] = LOW;
       prevDigiPins [i] = DigiPins[i]; //pevDigiPins used for checking change of state
       };


  Serial.println();
  Serial.println("Pin_No, Direct, Mill, TSec, Sec, Min"); //  iSec"";"Min");



}
// ~~~~~~~~~~~~~~ MAIN PROGRAM ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
void loop() {

scan_all_pins (); // Find out what is on any pint, including isntrumnets and notes
```

```
//INITIAL Check of buttons CHECK VALUE of BUTTONS ~ Whats going on in the real world
// NB some notes can be held on e.g wind instrument, others not i.e. plucked or hit.

//====Detect Instrument to be played  played given by the rotary switches ======

valInstSelect = analogRead(4);

valRange = valInstSelect;
// Resistor change range 0, 203, 408, 613, 819, 1023. see analog test routine.
if (valRange <=100) // Is it  Piano
   { instrument_type1 = acoustic_grand_piano; }
else if(valRange <=250) // Is it Organ01
   { instrument_type1 = piccolo; }
else if(valRange <=500) // Is it Organ01
   { instrument_type1 = electric_guitar_jazz; }
else if(valRange <=700) // Is it Organ01
   { instrument_type1 = slow_strings; }
else if(valRange <=900) // Is it Organ01
   { instrument_type1 = cello; }
else if(valRange <1050) // Is it Organ01
   { instrument_type1 = choir_aahs; }


valPercSelect = analogRead(5);
valPercRange = valPercSelect;

if (valPercRange <=100) // Switch Position 1
    {
      percussion_type1 = taiko; // initila set up to wood block = 115
      percussion_type2 = melo_tom;
      percussion_type3 = synth_drum;
      percussion_type4 = timpani;
//      percussion_type1 = taiko ;
//     percussion_type2 = melo_tom;
//     percussion_type3 = synth_drum;
//     percussion_type3 = agogo;
    }
else if(valPercRange <=250) // Switch Position 2
    {
     percussion_type1 = woodblock;
     percussion_type2 = tinkle_bell;
     percussion_type3 = agogo;
     percussion_type4 = timpani;
}

else if(valPercRange <=500) // Switch Position 3
    {
     percussion_type1 = pitched_percussion ;
     percussion_type2 = reverse_cymbal;
     percussion_type3 = timpani;
     percussion_type4 = taiko ;
 }
else if(valPercRange <=700) // Switch Position 4
```

```
 {
    percussion_type1 = timpani ;
    percussion_type2 = melo_tom;
    percussion_type3 = synth_drum;
    percussion_type4 = agogo;
   }
else if(valPercRange <=900) // Switch Position 5
   { percussion_type1 = tinkle_bell;
     percussion_type2 = woodblock;
     percussion_type3 = agogo;
     percussion_type4 = synth_drum;
   }

else if(valPercRange <1050) // Switch Position 6
    { percussion_type1 = agogo ;
      percussion_type2 = reverse_cymbal;
      percussion_type3 = timpani;
      percussion_type4 = melo_tom;
 }


//  vol control
/*
logVol = analogRead(3);
velo_main = logVol/1024;
Remmed out 17Nov14 ~ currently not uded.
*/
//Update to instrument as switched in
  midiSerial.write(192); // select channel 1
  midiSerial.write(instrument_type1); // tell it what instrument

  midiSerial.write(193); // select channel 2
  midiSerial.write(percussion_type1); // tell it what inst (percussion)
  midiSerial.write(194); // select channel 3
  midiSerial.write(percussion_type2); // tell it what inst (percussion)
  midiSerial.write(195); // select channel 3
  midiSerial.write(percussion_type3); // tell it what inst (percussion)
  midiSerial.write(196); // select channel 4
  midiSerial.write(percussion_type4);

// ==== Added in to for testing, 5pin o/p to SD-50 //NB 5 pin MIDI CANNOT be used under normal running on a
UNO ~ see above
//  Serial.begin(31250);  // Set up MIDI baud rate Serial.write(192); // select channel 1
//  Serial.write(instrument_type1); // tell it what chordedinstrument

//  Serial.write(193); // select channel 2
//  Serial.write(percussion_type1); // tell it 1st percussion
//  Serial.write(194); // select channel 3
//  Serial.write(percussion_type2); // tell it 2nd percussion
//  Serial.write(195); // select channel 4
//  Serial.write(percussion_type3);
//  Serial.write(196); // select channel 5
//  Serial.write(percussion_type4);
```

```
// End of instrument update

//=^^^^^^^^^^^^^^^^ End of Intrument and percussion detect ^^^^^^^^^^^^^^^^
scan_DigiPins();//for Digi pins used ulitmatly just for playing rather than selection

// ======== Begin  ========= 4 Chord Xmit Routines ================================

//============================ A CHORD  pin 5 etc ================================
//chordnote1 = Guitar_Chord_A[0] ; chordnote2 = Guitar_Chord_A[1] ; chordnote3 = Guitar_Chord_A[2] ;
chordnote4 = Guitar_Chord_A[3] ; chordnote5 = Guitar_Chord_A[4] ; chordnote6 = Guitar_Chord_A[5] ;
// Chord A = Digital Pin 5 ~ then 6, 7, 8, 9, 10, 11, 12  (8 Chords)

if (prevDigiPins[5] == HIGH && DigiPins[5] == HIGH)  //ast two percussion
    {  prevDigiPins[5] = HIGH; }

if (prevDigiPins[5] == HIGH && DigiPins[5] == LOW)  //ast two percussion
    {
    ChordNoteOn(0x90, Guitar_Chord_A[0], velocity_75 * velo_preset * velo_main);delay(10);
    ChordNoteOn(0x90, Guitar_Chord_A[1], velocity_75 * velo_preset * velo_main);delay(5);
    ChordNoteOn(0x90, Guitar_Chord_A[2], velocity_75 * velo_preset * velo_main);delay(10);
    ChordNoteOn(0x90, Guitar_Chord_A[3], velocity_75 * velo_preset * velo_main);delay(5);
    ChordNoteOn(0x90, Guitar_Chord_A[4], velocity_75 * velo_preset * velo_main);delay(10);
    ChordNoteOn(0x90, Guitar_Chord_A[5], velocity_75 * velo_preset * velo_main);
     prevDigiPins[5] = LOW;
     Serial.print("Pin5");
      Serial.print(", ");
      Serial.print("HiLo");  // Could just put HiLo5 and save a column
      Serial.print(", ");
      Serial.print(millis()); // mill sec
      Serial.print(", ");
      Serial.print(millis()/100); // tenth sec
      Serial.print(", ");
      Serial.print(millis()/1000); // second
      Serial.print(", ");
      Serial.print((millis()/1000)/60); // minute
      Serial.print(",V= ");
      Serial.print(velo_main); // minute
      Serial.print(",IR= ");
      Serial.print(valRange); // minute
      Serial.println();
     delay(50);
    }

if (prevDigiPins[5] == LOW && DigiPins[5] == HIGH)  //ast two percussion
    {
    ChordNoteOn(0x90, Guitar_Chord_A[0], velocity_0);
    ChordNoteOn(0x90, Guitar_Chord_A[1], velocity_0);
    ChordNoteOn(0x90, Guitar_Chord_A[2], velocity_0);
    ChordNoteOn(0x90, Guitar_Chord_A[3], velocity_0);
    ChordNoteOn(0x90, Guitar_Chord_A[4], velocity_0);
    ChordNoteOn(0x90, Guitar_Chord_A[5], velocity_0);
    prevDigiPins[5] = HIGH;
    Serial.print("Pin5");
      Serial.print(", ");
```

```
      Serial.print("LoHi");
      Serial.print(", ");
      Serial.print(millis()); // mill sec
      Serial.print(", ");
      Serial.print(millis()/100); // tenth sec
      Serial.print(", ");
      Serial.print(millis()/1000); // second
      Serial.print(", ");
      Serial.print((millis()/1000)/60); // minute
      Serial.print(",V= ");
      Serial.print(velo_main); // minute
      Serial.print(",IR= ");
      Serial.print(valRange); // minute
      Serial.println();
    }

if (prevDigiPins[5] == LOW && DigiPins[5] == LOW)  //ast two percussion
    { prevDigiPins[5] = LOW; }
//======================END OF ======= A CHORD
================================================

//======================BEGIN ======= E CHORD
================================================

if (prevDigiPins[6] == HIGH && DigiPins[6] == HIGH)  //ast two percussion
    {
     prevDigiPins[6] = HIGH;
     }

if (prevDigiPins[6] == HIGH && DigiPins[6] == LOW)  //ast two percussion
    {
    ChordNoteOn(0x90, Guitar_Chord_E[0], velocity_75 * velo_preset * velo_main);delay(10);
    ChordNoteOn(0x90, Guitar_Chord_E[1], velocity_75 * velo_preset * velo_main);delay(5);
    ChordNoteOn(0x90, Guitar_Chord_E[2], velocity_75 * velo_preset * velo_main);delay(10);
    ChordNoteOn(0x90, Guitar_Chord_E[3], velocity_75 * velo_preset * velo_main);delay(5);
    ChordNoteOn(0x90, Guitar_Chord_E[4], velocity_75 * velo_preset * velo_main);delay(10);
    ChordNoteOn(0x90, Guitar_Chord_E[5], velocity_75 * velo_preset * velo_main);
     prevDigiPins[6] = LOW;
      Serial.print("Pin6");
      Serial.print(", ");
      Serial.print("HiLo");
      Serial.print(", ");
      Serial.print(millis()); // mill sec
      Serial.print(", ");
      Serial.print(millis()/100); // tenth sec
      Serial.print(", ");
      Serial.print(millis()/1000); // second
      Serial.print(", ");
      Serial.print((millis()/1000)/60); // minute
       Serial.print(",V= ");
      Serial.print(velo_main); // minute
      Serial.print(",IR= ");
      Serial.print(valRange); // minute
      Serial.println();
```

```
     delay(50);
    }
if (prevDigiPins[6] == LOW && DigiPins[6] == HIGH)  //ast two percussion
    {
    ChordNoteOn(0x90, Guitar_Chord_E[0], velocity_0);
    ChordNoteOn(0x90, Guitar_Chord_E[1], velocity_0);
    ChordNoteOn(0x90, Guitar_Chord_E[2], velocity_0);
    ChordNoteOn(0x90, Guitar_Chord_E[3], velocity_0);
    ChordNoteOn(0x90, Guitar_Chord_E[4], velocity_0);
    ChordNoteOn(0x90, Guitar_Chord_E[5], velocity_0);
     prevDigiPins[6] = HIGH;
      Serial.print("Pin6");
      Serial.print(", ");
      Serial.print("LoHi");
      Serial.print(", ");
      Serial.print(millis()); // mill sec
      Serial.print(", ");
      Serial.print(millis()/100); // tenth sec
      Serial.print(", ");
      Serial.print(millis()/1000); // second
      Serial.print(", ");
      Serial.print((millis()/1000)/60); // minute
       Serial.print(",V= ");
      Serial.print(velo_main); // minute
      Serial.print(",IR= ");
      Serial.print(valRange); // minute
      Serial.println();
    }
if (prevDigiPins[6] == LOW && DigiPins[6] == LOW)  //ast two percussion
    { prevDigiPins[6] = LOW; }

//====================== END OF ======= E CHORD
================================================


//====================== BEGIN ======= D CHORD  ===== pin 7
==========================================

if (prevDigiPins[7] == HIGH && DigiPins[7] == HIGH)  //ast two percussion
    { prevDigiPins[7] = HIGH;  }

if (prevDigiPins[7] == HIGH && DigiPins[7] == LOW)  //ast two percussion
    {
     ChordNoteOn(0x90, Guitar_Chord_D[0], velocity_75 * velo_preset * velo_main);delay(10);
     ChordNoteOn(0x90, Guitar_Chord_D[1], velocity_75 * velo_preset * velo_main);delay(5);
     ChordNoteOn(0x90, Guitar_Chord_D[2], velocity_75 * velo_preset * velo_main);delay(10);
     ChordNoteOn(0x90, Guitar_Chord_D[3], velocity_75 * velo_preset * velo_main);delay(5);
     ChordNoteOn(0x90, Guitar_Chord_D[4], velocity_75 * velo_preset * velo_main);delay(10);
     ChordNoteOn(0x90, Guitar_Chord_D[5], velocity_0 * velo_preset * velo_main);
     prevDigiPins[7] = LOW;
     Serial.print("Pin7");
      Serial.print(", ");
      Serial.print("HiLo");
      Serial.print(", ");
```

```
      Serial.print(millis()); // mill sec
      Serial.print(", ");
      Serial.print(millis()/100); // tenth sec
      Serial.print(", ");
      Serial.print(millis()/1000); // second
      Serial.print(", ");
      Serial.print((millis()/1000)/60); // minute
       Serial.print(",V= ");
      Serial.print(velo_main); // minute
      Serial.print(",IR= ");
      Serial.print(valRange); // minute
      Serial.println();

     delay(50);
    }

if (prevDigiPins[7] == LOW && DigiPins[7] == HIGH)  //ast two percussion
    {
     ChordNoteOn(0x90, Guitar_Chord_D[0], velocity_0);
     ChordNoteOn(0x90, Guitar_Chord_D[1], velocity_0);
     ChordNoteOn(0x90, Guitar_Chord_D[2], velocity_0);
     ChordNoteOn(0x90, Guitar_Chord_D[3], velocity_0);
     ChordNoteOn(0x90, Guitar_Chord_D[4], velocity_0);
     ChordNoteOn(0x90, Guitar_Chord_D[5], velocity_0);
     prevDigiPins[7] = HIGH;
     Serial.print("Pin7");
      Serial.print(", ");
      Serial.print("LoHi");
      Serial.print(", ");
      Serial.print(millis()); // mill sec
      Serial.print(", ");
      Serial.print(millis()/100); // tenth sec
      Serial.print(", ");
      Serial.print(millis()/1000); // second
      Serial.print(", ");
      Serial.print((millis()/1000)/60); // minute
       Serial.print(",V= ");
      Serial.print(velo_main); // minute
      Serial.print(",IR= ");
      Serial.print(valRange); // minute
      Serial.println();
    }
if (prevDigiPins[7] == LOW && DigiPins[7] == LOW)  //ast two percussion
    { prevDigiPins[7] = LOW;  }

//====================== END ======= D CHORD
================================================


//====================== BEGIN ======= G CHORD  === pin 8
==========================================

if (prevDigiPins[8] == HIGH && DigiPins[8] == HIGH)  //ast two percussion
    { prevDigiPins[8] = HIGH; }
```

```
if (prevDigiPins[8] == HIGH && DigiPins[8] == LOW)  //ast two percussion
    {
    ChordNoteOn(0x90, Guitar_Chord_G[0], velocity_75 * velo_preset * velo_main);delay(10);
    ChordNoteOn(0x90, Guitar_Chord_G[1], velocity_75 * velo_preset * velo_main);delay(5);
    ChordNoteOn(0x90, Guitar_Chord_G[2], velocity_75 * velo_preset * velo_main);delay(10);
    ChordNoteOn(0x90, Guitar_Chord_G[3], velocity_75 * velo_preset * velo_main);delay(5);
    ChordNoteOn(0x90, Guitar_Chord_G[4], velocity_75 * velo_preset * velo_main);delay(10);
    ChordNoteOn(0x90, Guitar_Chord_G[5], velocity_75 * velo_preset * velo_main);
     prevDigiPins[8] = LOW;
     Serial.print("Pin8");
      Serial.print(", ");
      Serial.print("HiLo");
      Serial.print(", ");
      Serial.print(millis()); // mill sec
      Serial.print(", ");
      Serial.print(millis()/100); // tenth sec
      Serial.print(", ");
      Serial.print(millis()/1000); // second
      Serial.print(", ");
      Serial.print((millis()/1000)/60); // minute
       Serial.print(",V= ");
      Serial.print(velo_main); // minute
      Serial.print(",IR= ");
      Serial.print(valRange); // minute
      Serial.println();
     delay(50);
    }
if (prevDigiPins[8] == LOW && DigiPins[8] == HIGH)  //ast two percussion
    {
     ChordNoteOn(0x90, Guitar_Chord_G[0], velocity_0);
     ChordNoteOn(0x90, Guitar_Chord_G[1], velocity_0);
     ChordNoteOn(0x90, Guitar_Chord_G[2], velocity_0);
     ChordNoteOn(0x90, Guitar_Chord_G[3], velocity_0);
     ChordNoteOn(0x90, Guitar_Chord_G[4], velocity_0);
     ChordNoteOn(0x90, Guitar_Chord_G[5], velocity_0);
     prevDigiPins[8] = HIGH;
     Serial.print("Pin8");
      Serial.print(", ");
      Serial.print("LoHi");
      Serial.print(", ");
      Serial.print(millis()); // mill sec
      Serial.print(", ");
      Serial.print(millis()/100); // tenth sec
      Serial.print(", ");
      Serial.print(millis()/1000); // second
      Serial.print(", ");
      Serial.print((millis()/1000)/60); // minute
       Serial.print(",V= ");
      Serial.print(velo_main); // minute
      Serial.print(",IR= ");
      Serial.print(valRange); // minute
      Serial.println();
    }
```

```
if (prevDigiPins[8] == LOW && DigiPins[8] == LOW)  //ast two percussion
    { prevDigiPins[8] = LOW;   }
//====================== END ======= G CHORD
================================================
// ===========END OFF ====== 4 Chord Xmit Routines ================================

// ====================== Next 4===========A7, C, B7, D7 ============================

//============================ A7 CHORD  ===== pin 9
================================================

if (prevDigiPins[9] == HIGH && DigiPins[9] == HIGH)  //ast two percussion
    { prevDigiPins[9] = HIGH;  }

if (prevDigiPins[9] == HIGH && DigiPins[9] == LOW)  //ast two percussion
    {
    ChordNoteOn(0x90, Guitar_Chord_A7[0], velocity_75 * velo_preset * velo_main);delay(10);
    ChordNoteOn(0x90, Guitar_Chord_A7[1], velocity_75 * velo_preset * velo_main);delay(5);
    ChordNoteOn(0x90, Guitar_Chord_A7[2], velocity_75 * velo_preset * velo_main);delay(10);
    ChordNoteOn(0x90, Guitar_Chord_A7[3], velocity_75 * velo_preset * velo_main);delay(5);
    ChordNoteOn(0x90, Guitar_Chord_A7[4], velocity_75 * velo_preset * velo_main);delay(10);
    ChordNoteOn(0x90, Guitar_Chord_A7[5], velocity_0 * velo_preset * velo_main);
     prevDigiPins[9] = LOW;
       Serial.print("Pin9");
      Serial.print(", ");
      Serial.print("HiLo");
      Serial.print(", ");
      Serial.print(millis()); // mill sec
      Serial.print(", ");
      Serial.print(millis()/100); // tenth sec
      Serial.print(", ");
      Serial.print(millis()/1000); // second
      Serial.print(", ");
      Serial.print((millis()/1000)/60); // minute
       Serial.print(",V= ");
      Serial.print(velo_main); // minute
      Serial.print(",IR= ");
      Serial.print(valRange); // minute
      Serial.println();

     delay(50);
    }
if (prevDigiPins[9] == LOW && DigiPins[9] == HIGH)  //ast two percussion
    {
     ChordNoteOn(0x90, Guitar_Chord_A7[0], velocity_0);
     ChordNoteOn(0x90, Guitar_Chord_A7[1], velocity_0);
     ChordNoteOn(0x90, Guitar_Chord_A7[2], velocity_0);
     ChordNoteOn(0x90, Guitar_Chord_A7[3], velocity_0);
     ChordNoteOn(0x90, Guitar_Chord_A7[4], velocity_0);
     ChordNoteOn(0x90, Guitar_Chord_A7[5], velocity_0);
     prevDigiPins[9] = HIGH;
      Serial.print("Pin9");
      Serial.print(", ");
```

```
      Serial.print("LoHi");
      Serial.print(", ");
      Serial.print(millis()); // mill sec
      Serial.print(", ");
      Serial.print(millis()/100); // tenth sec
      Serial.print(", ");
      Serial.print(millis()/1000); // second
      Serial.print(", ");
      Serial.print((millis()/1000)/60); // minute
       Serial.print(",V= ");
      Serial.print(velo_main); // minute
      Serial.print(",IR= ");
      Serial.print(valRange); // minute
      Serial.println();
    }

if (prevDigiPins[9] == LOW && DigiPins[9] == LOW)  //ast two percussion
    { prevDigiPins[9] = LOW;  }

//============================== End of A7 CHORD
================================================


//============================== C CHORD  == pin 10
=====================================================

if (prevDigiPins[10] == HIGH && DigiPins[10] == HIGH)  //ast two percussion
    { prevDigiPins[10] = HIGH; }

if (prevDigiPins[10] == HIGH && DigiPins[10] == LOW)  //ast two percussion
    {
    ChordNoteOn(0x90, Guitar_Chord_C[0], velocity_75 * velo_preset * velo_main);delay(10);
    ChordNoteOn(0x90, Guitar_Chord_C[1], velocity_75 * velo_preset * velo_main);delay(5);
    ChordNoteOn(0x90, Guitar_Chord_C[2], velocity_75 * velo_preset * velo_main);delay(10);
    ChordNoteOn(0x90, Guitar_Chord_C[3], velocity_75 * velo_preset * velo_main);delay(5);
    ChordNoteOn(0x90, Guitar_Chord_C[4], velocity_75 * velo_preset * velo_main);delay(10);
    ChordNoteOn(0x90, Guitar_Chord_C[5], velocity_0);
     prevDigiPins[10] = LOW;
      Serial.print("Pin10");
      Serial.print(", ");
      Serial.print("HiLo");
      Serial.print(", ");
      Serial.print(millis()); // mill sec
      Serial.print(", ");
      Serial.print(millis()/100); // tenth sec
      Serial.print(", ");
      Serial.print(millis()/1000); // second
      Serial.print(", ");
      Serial.print((millis()/1000)/60); // minute
       Serial.print(",V= ");
      Serial.print(velo_main); // minute
      Serial.print(",IR= ");
      Serial.print(valRange); // minute
      Serial.println();
```

```
      delay(50);
     }
if (prevDigiPins[10] == LOW && DigiPins[10] == HIGH)  //ast two percussion
     {
      ChordNoteOn(0x90, Guitar_Chord_C[0], velocity_0);
      ChordNoteOn(0x90, Guitar_Chord_C[1], velocity_0);
      ChordNoteOn(0x90, Guitar_Chord_C[2], velocity_0);
      ChordNoteOn(0x90, Guitar_Chord_C[3], velocity_0);
      ChordNoteOn(0x90, Guitar_Chord_C[4], velocity_0);
      ChordNoteOn(0x90, Guitar_Chord_C[5], velocity_0);
      prevDigiPins[10] = HIGH;
       Serial.print("Pin10");
       Serial.print(", ");
       Serial.print("LoHi");
       Serial.print(", ");
       Serial.print(millis()); // mill sec
       Serial.print(", ");
       Serial.print(millis()/100); // tenth sec
       Serial.print(", ");
       Serial.print(millis()/1000); // second
       Serial.print(", ");
       Serial.print((millis()/1000)/60); // minute
        Serial.print(",V= ");
       Serial.print(velo_main); // minute
       Serial.print(",IR= ");
       Serial.print(valRange); // minute
       Serial.println();

     }

if (prevDigiPins[10] == LOW && DigiPins[10] == LOW)  //ast two percussion
     { prevDigiPins[10] = LOW;   }



//============================== End of C CHORD
================================================


//============================== B7 CHORD  === pin 11
=====================================================

if (prevDigiPins[11] == HIGH && DigiPins[11] == HIGH)  //ast two percussion
     { prevDigiPins[11] = HIGH; }

if (prevDigiPins[11] == HIGH && DigiPins[11] == LOW)  //ast two percussion
     {
     ChordNoteOn(0x90, Guitar_Chord_B7[0], velocity_75 * velo_preset * velo_main);delay(10);
     ChordNoteOn(0x90, Guitar_Chord_B7[1], velocity_75 * velo_preset * velo_main);delay(5);
     ChordNoteOn(0x90, Guitar_Chord_B7[2], velocity_75 * velo_preset * velo_main);delay(10);
     ChordNoteOn(0x90, Guitar_Chord_B7[3], velocity_75 * velo_preset * velo_main);delay(5);
     ChordNoteOn(0x90, Guitar_Chord_B7[4], velocity_75 * velo_preset * velo_main);delay(10);
     ChordNoteOn(0x90, Guitar_Chord_B7[5], velocity_0 * velo_preset * velo_main);
```

```
      prevDigiPins[11] = LOW;
       Serial.print("Pin11");
       Serial.print(", ");
       Serial.print("HiLo");
       Serial.print(", ");
       Serial.print(millis()); // mill sec
       Serial.print(", ");
       Serial.print(millis()/100); // tenth sec
       Serial.print(", ");
       Serial.print(millis()/1000); // second
       Serial.print(", ");
       Serial.print((millis()/1000)/60); // minute
        Serial.print(",V= ");
       Serial.print(velo_main); // minute
       Serial.print(",IR= ");
       Serial.print(valRange); // minute
       Serial.println();
      delay(50);
     }
if (prevDigiPins[11] == LOW && DigiPins[11] == HIGH)  //ast two percussion
     {
      ChordNoteOn(0x90, Guitar_Chord_B7[0], velocity_0);
      ChordNoteOn(0x90, Guitar_Chord_B7[1], velocity_0);
      ChordNoteOn(0x90, Guitar_Chord_B7[2], velocity_0);
      ChordNoteOn(0x90, Guitar_Chord_B7[3], velocity_0);
      ChordNoteOn(0x90, Guitar_Chord_B7[4], velocity_0);
      ChordNoteOn(0x90, Guitar_Chord_B7[5], velocity_0);
      prevDigiPins[11] = HIGH;
       Serial.print("Pin11");
       Serial.print(", ");
       Serial.print("LoHi");
       Serial.print(", ");
       Serial.print(millis()); // mill sec
       Serial.print(", ");
       Serial.print(millis()/100); // tenth sec
       Serial.print(", ");
       Serial.print(millis()/1000); // second
       Serial.print(", ");
       Serial.print((millis()/1000)/60); // minute
        Serial.print(",V= ");
       Serial.print(velo_main); // minute
       Serial.print(",IR= ");
       Serial.print(valRange); // minute
       Serial.println();
     }

if (prevDigiPins[11] == LOW && DigiPins[11] == LOW)  //ast two percussion
    { prevDigiPins[11] = LOW;  }
//============================= End of B7 CHORD
================================================


//============================= D7 CHORD  ======= pin 12
================================================
```

```
if (prevDigiPins[12] == HIGH && DigiPins[12] == HIGH)  //ast two percussion
    {  prevDigiPins[12] = HIGH;  }

if (prevDigiPins[12] == HIGH && DigiPins[12] == LOW)  //ast two percussion
    {
    ChordNoteOn(0x90, Guitar_Chord_D7[0], velocity_75 * velo_preset * velo_main);delay(10);
    ChordNoteOn(0x90, Guitar_Chord_D7[1], velocity_75 * velo_preset * velo_main);delay(5);
    ChordNoteOn(0x90, Guitar_Chord_D7[2], velocity_75 * velo_preset * velo_main);delay(10);
    ChordNoteOn(0x90, Guitar_Chord_D7[3], velocity_75 * velo_preset * velo_main);delay(5);
    ChordNoteOn(0x90, Guitar_Chord_D7[4], velocity_75 * velo_preset * velo_main);delay(10);
    ChordNoteOn(0x90, Guitar_Chord_D7[5], velocity_0);
     prevDigiPins[12] = LOW;
     delay(50);
      Serial.print("Pin12");
      Serial.print(", ");
      Serial.print("HiLo");
      Serial.print(", ");
      Serial.print(millis()); // mill sec
      Serial.print(", ");
      Serial.print(millis()/100); // tenth sec
      Serial.print(", ");
      Serial.print(millis()/1000); // second
      Serial.print(", ");
      Serial.print((millis()/1000)/60); // minute
       Serial.print(",V= ");
      Serial.print(velo_main); // minute
      Serial.print(",IR= ");
      Serial.print(valRange); // minute
      Serial.println();
    }
if (prevDigiPins[12] == LOW && DigiPins[12] == HIGH)  //ast two percussion
    {
     ChordNoteOn(0x90, Guitar_Chord_D7[0], velocity_0);
     ChordNoteOn(0x90, Guitar_Chord_D7[1], velocity_0);
     ChordNoteOn(0x90, Guitar_Chord_D7[2], velocity_0);
     ChordNoteOn(0x90, Guitar_Chord_D7[3], velocity_0);
     ChordNoteOn(0x90, Guitar_Chord_D7[4], velocity_0);
     ChordNoteOn(0x90, Guitar_Chord_D7[5], velocity_0);
     prevDigiPins[12] = HIGH;
      Serial.print("Pin12");
      Serial.print(", ");
      Serial.print("LoHi");
      Serial.print(", ");
      Serial.print(millis()); // mill sec
      Serial.print(", ");
      Serial.print(millis()/100); // tenth sec
      Serial.print(", ");
      Serial.print(millis()/1000); // second
      Serial.print(", ");
      Serial.print((millis()/1000)/60); // minute
       Serial.print(",V= ");
      Serial.print(velo_main); // minute
      Serial.print(",IR= ");
```

```
      Serial.print(valRange); // minute
      Serial.println();
    }

if (prevDigiPins[12] == LOW && DigiPins[12] == LOW)  //ast two percussion
    {  prevDigiPins[12] = LOW;   }

//============================== End of D7 CHORD
================================================


// ======================= End of Next 4============A7, C, B7, D7
==============================


//================================== PERC12 XMIT ROUTINES
================================


//new 9th May detect xfr P1 to P3 and P3 to P1




// ======================= Perc 1  ========= pin 13 ========================== P1 Button

   if (prevDigiPins[13] == HIGH && DigiPins[13] == HIGH)  //ast two percussion
         {
     //    percNoteOn(0x91, percussion_note1, velocity_0);//ensure not eis zeroed BUT MAYBE REMOVE LATERAS
IT IS SENDING SIGNAL, BUT MAYBE KEEP FOR DATA LOG, THOUGH MAYBE NOT AS CUBASE SHOUL
JUSTRECORD SILEENE
     //    prevDigiPins[13] = HIGH;
          P1_count = P1_count +1; // for percussion xfr velocity detect
          }
   if (prevDigiPins[13] ==  HIGH && DigiPins[13] == LOW)  //ast two percussion
         {
          noteOn(1, percussion_note1, velocity_125 * velo_preset * velo_main);
          prevDigiPins[13] = LOW;
           Serial.print("Pin13");
           Serial.print(", ");
           Serial.print("HiLo");
           Serial.print(", ");
           Serial.print(millis()); // mill sec
           Serial.print(", ");
           Serial.print(millis()/100); // tenth sec
           Serial.print(", ");
           Serial.print(millis()/1000); // second
           Serial.print(", ");
           Serial.print((millis()/1000)/60); // minute
          Serial.print(",V= ");
      Serial.print(velo_main); // minute
      Serial.print(",IR= ");
      Serial.print(valRange); // minute
          Serial.println();
```

```
        P1_count = P1_count +1; // for percussion xfr velocity detect

        if (P1_count > 2)
         {
         noteOn(1, percussion_note1, velocity_125 * velo_preset * velo_main); //26May14 not2 >>note1 (error found applause note staying on whhenapplause tried out for test
         }

       //   delay(25);
        }
  if (prevDigiPins[13] == LOW && DigiPins[13] == HIGH)  //ast two percussion
        {
         noteOn(1, percussion_note1, velocity_0);
         prevDigiPins[13] = HIGH;
          Serial.print("Pin13");
          Serial.print(", ");
          Serial.print("LoHi");
          Serial.print(", ");
          Serial.print(millis()); // mill sec
          Serial.print(", ");
          Serial.print(millis()/100); // tenth sec
          Serial.print(", ");
          Serial.print(millis()/1000); // second
          Serial.print(", ");
          Serial.print((millis()/1000)/60); // minute
           Serial.print(",V= ");
     Serial.print(velo_main); // minute
     Serial.print(",IR= ");
     Serial.print(valRange); // minute
          Serial.println();
        }
  if (prevDigiPins[13] == LOW && DigiPins[13] == LOW)  //ast two percussion
        {
        prevDigiPins[13] = LOW;
        }
//===================END OF ======= PERC12 XMIT ROUTINES ================================


//================================= PERC13 XMIT ROUTINES ======= pin 14 (A0)============ P2 Button
  if (prevDigiPins[14] == HIGH && DigiPins[14] == HIGH)  //ast two percussion
        {
    //    percNoteOn(0x91, percussion_note2, velocity_0);//ensure not eis zeroed BUT MAYBE REMOVE LATERAS IT IS SENDING SIGNAL, BUT MAYBE KEEP FOR DATA LOG, THOUGH MAYBE NOT AS CUBASE SHOUL JUSTRECORD SILEENE
    //    prevDigiPins[14] = HIGH;
         P2_count = P2_count +1; // for percussion xfr velocity detect
         }
  if (prevDigiPins[14] ==  HIGH && DigiPins[14] == LOW)  //ast two percussion
        {
         noteOn(2, percussion_note2, velocity_125 * velo_preset * velo_main);
         prevDigiPins[14] = LOW;
```

```
          Serial.print("Pin14");
          Serial.print(", ");
          Serial.print("HiLo");
          Serial.print(", ");
          Serial.print(millis()); // mill sec
          Serial.print(", ");
          Serial.print(millis()/100); // tenth sec
          Serial.print(", ");
          Serial.print(millis()/1000); // second
          Serial.print(", ");
          Serial.print((millis()/1000)/60); // minute
           Serial.print(",V= ");
     Serial.print(velo_main); // minute
     Serial.print(",IR= ");
     Serial.print(valRange); // minute
          Serial.println();


        P2_count = P2_count +1; // for percussion xfr velocity detect
      //   delay(25);
        }
  if (prevDigiPins[14] == LOW && DigiPins[14] == HIGH)  //ast two percussion
        {
         noteOn(2, percussion_note2, velocity_0);
         prevDigiPins[14] = HIGH;
                  Serial.print("Pin14");
          Serial.print(", ");
          Serial.print("LoHi");
          Serial.print(", ");
          Serial.print(millis()); // mill sec
          Serial.print(", ");
          Serial.print(millis()/100); // tenth sec
          Serial.print(", ");
          Serial.print(millis()/1000); // second
          Serial.print(", ");
          Serial.print((millis()/1000)/60); // minute
           Serial.print(",V= ");
     Serial.print(velo_main); // minute
     Serial.print(",IR= ");
     Serial.print(valRange); // minute
          Serial.println();

        }
  if (prevDigiPins[14] == LOW && DigiPins[14] == LOW)  //ast two percussion
        {
        prevDigiPins[14] = LOW;
        }
//==================END OF ======= PERC13 XMIT ROUTINES
================================

//================================= PERC14 XMIT ROUTINES ===== pin 15  =========== P3 Button
  if (prevDigiPins[15] == HIGH && DigiPins[15] == HIGH)  //ast two percussion
        {
     //    noteOn(0, percussion_note3, velocity_0);
```

```
    //    prevDigiPins[15] = HIGH;
          P3_count = P3_count +1; // for percussion xfr velocity detect
         }
  if (prevDigiPins[15] ==  HIGH && DigiPins[15] == LOW)  //ast two percussion
        {
         noteOn(3, percussion_note3, velocity_125 * velo_preset * velo_main);
         prevDigiPins[15] = LOW;
          Serial.print("Pin15");
          Serial.print(", ");
          Serial.print("HiLo");
          Serial.print(", ");
          Serial.print(millis()); // mill sec
          Serial.print(", ");
          Serial.print(millis()/100); // tenth sec
          Serial.print(", ");
          Serial.print(millis()/1000); // second
          Serial.print(", ");
          Serial.print((millis()/1000)/60); // minute
           Serial.print(",V= ");
     Serial.print(velo_main); // minute
     Serial.print(",IR= ");
     Serial.print(valRange); // minute
          Serial.println();


         P3_count = P3_count +1; // for percussion xfr velocity detect
         }
  if (prevDigiPins[15] == LOW && DigiPins[15] == HIGH)  //ast two percussion
        {
         noteOn(3, percussion_note3, velocity_0);
         prevDigiPins[15] = HIGH;
          Serial.print("Pin15");
          Serial.print(", ");
          Serial.print("LoHi");
          Serial.print(", ");
          Serial.print(millis()); // mill sec
          Serial.print(", ");
          Serial.print(millis()/100); // tenth sec
          Serial.print(", ");
          Serial.print(millis()/1000); // second
          Serial.print(", ");
          Serial.print((millis()/1000)/60); // minute
           Serial.print(",V= ");
     Serial.print(velo_main); // minute
     Serial.print(",IR= ");
     Serial.print(valRange); // minute
          Serial.println();

        }
  if (prevDigiPins[15] == LOW && DigiPins[15] == LOW)  //ast two percussion
        {
         prevDigiPins[15] = LOW;
        }
//==================END OF ======= PERC14 XMIT ROUTINES
```

```
================================

 //================================= PERC16 XMIT ROUTINES ===== pin 16 ============== P4 Button
  if (prevDigiPins[16] == HIGH && DigiPins[16] == HIGH)  //ast two percussion
        {
     //   noteOff(0, percussion_note4, velocity_0);
     //   prevDigiPins[16] = HIGH;
       P4_count = P4_count +1; // for percussion xfr velocity detect
       }
   if (prevDigiPins[16] ==  HIGH && DigiPins[16] == LOW)  //ast two percussion
        {
       noteOn(4, percussion_note4, velocity_125 * velo_preset * velo_main);
       prevDigiPins[16] = LOW;
         Serial.print("Pin16");
         Serial.print(", ");
         Serial.print("HiLo");
         Serial.print(", ");
         Serial.print(millis()); // mill sec
         Serial.print(", ");
         Serial.print(millis()/100); // tenth sec
         Serial.print(", ");
         Serial.print(millis()/1000); // second
         Serial.print(", ");
         Serial.print((millis()/1000)/60); // minute
          Serial.print(",V= ");
    Serial.print(velo_main); // minute
    Serial.print(",IR= ");
    Serial.print(valRange); // minute
        Serial.println();

       P4_count = P4_count +1; // for percussion xfr velocity detect

       }
   if (prevDigiPins[16] == LOW && DigiPins[16] == HIGH)  //ast two percussion
        {
       noteOff(4, percussion_note4, velocity_0);
       prevDigiPins[16] = HIGH;
         Serial.print("Pin16");
         Serial.print(", ");
         Serial.print("LoHi");
         Serial.print(", ");
         Serial.print(millis()); // mill sec
         Serial.print(", ");
         Serial.print(millis()/100); // tenth sec
         Serial.print(", ");
         Serial.print(millis()/1000); // second
         Serial.print(", ");
         Serial.print((millis()/1000)/60); // minute
          Serial.print(",V= ");
    Serial.print(velo_main); // minute
    Serial.print(",IR= ");
    Serial.print(valRange); // minute
        Serial.println();
```

```
        }
    if (prevDigiPins[16] == LOW && DigiPins[16] == LOW)  //ast two percussion
        {
        prevDigiPins[16] = LOW;
        }
 //====================END OF ======= PERC15 XMIT ROUTINES ============================


}//  END OF MAIN PROGRAMME ; go back andresca keys and look for note
//============================================================


/*void percNoteOn(int cmd01, int pitch, int velocity)
  {
   Serial.write(cmd01);
   Serial.write(pitch);
   Serial.write(velocity);
  }
*/
void ChordNoteOn(int cmd01, int pitch, int velocity)
  {
   midiSerial.write(cmd01);
   midiSerial.write(pitch);
   midiSerial.write(velocity);
  }

void noteOnCh01(int cmd01, int pitch, int velocity)
{
  Serial.write(cmd01);
  Serial.write(pitch);
  Serial.write(velocity);
}

void noteOnCh02(int cmd02, int pitch, int velocity)
{
  Serial.write(cmd02);
  Serial.write(pitch);
  Serial.write(velocity);
}

void noteOnCh03(int cmd03, int pitch, int velocity)
{
  Serial.write(cmd03);
  Serial.write(pitch);
  Serial.write(velocity);
}

void noteOnCh04(int cmd04, int pitch, int velocity)
{
  Serial.write(cmd04);
  Serial.write(pitch);
  Serial.write(velocity);
}
```

```
//===============Clear Note Routines ================================================
// these clear note routines just do somthing, do not return a value
void clearNotes_Ch01 (int &chordnote1, int &chordnote2, int &chordnote3, int &chordnote4, int &chordnote5, int
&chordnote6 )
{
  noteOnCh01(0x90, chordnote1, 0x00);
  noteOnCh01(0x90, chordnote2, 0x00);
  noteOnCh01(0x90, chordnote3, 0x00);
  noteOnCh01(0x90, chordnote4, 0x00);
  noteOnCh01(0x90, chordnote5, 0x00);
  noteOnCh01(0x90, chordnote6, 0x00);
}

void clearNotes_Ch02 (int &chordnote1, int &chordnote2, int &chordnote3, int &chordnote4, int &chordnote5, int
&chordnote6 )
{
  noteOnCh02(0x91, chordnote1, 0x00);
  noteOnCh02(0x91, chordnote2, 0x00);
  noteOnCh02(0x91, chordnote3, 0x00);
  noteOnCh02(0x91, chordnote4, 0x00);
  noteOnCh02(0x91, chordnote5, 0x00);
  noteOnCh02(0x91, chordnote6, 0x00);
}

void clearNotes_Ch03 (int &chordnote1, int &chordnote2, int &chordnote3, int &chordnote4, int &chordnote5, int
&chordnote6 )
{
  noteOnCh03(0x92, chordnote1, 0x00);
  noteOnCh03(0x92, chordnote2, 0x00);
  noteOnCh03(0x92, chordnote3, 0x00);
  noteOnCh03(0x92, chordnote4, 0x00);
  noteOnCh03(0x92, chordnote5, 0x00);
  noteOnCh03(0x92, chordnote6, 0x00);
}

void clearNotes_Ch04 (int &chordnote1, int &chordnote2, int &chordnote3, int &chordnote4, int &chordnote5, int
&chordnote6 )
{
  noteOnCh04(0x93, chordnote1, 0x00);
  noteOnCh04(0x93, chordnote2, 0x00);
  noteOnCh04(0x93, chordnote3, 0x00);
  noteOnCh04(0x93, chordnote4, 0x00);
  noteOnCh04(0x93, chordnote5, 0x00);
  noteOnCh04(0x93, chordnote6, 0x00);
}

void clearNotes_Ch05 (int &chordnote1, int &chordnote2, int &chordnote3, int &chordnote4, int &chordnote5, int
&chordnote6 )
{
  noteOnCh04(0x93, chordnote1, 0x00);
  noteOnCh04(0x93, chordnote2, 0x00);
  noteOnCh04(0x93, chordnote3, 0x00);
  noteOnCh04(0x93, chordnote4, 0x00);
```

```
  noteOnCh04(0x93, chordnote5, 0x00);
  noteOnCh04(0x93, chordnote6, 0x00);
}

void reScan_Instrument_keys ()
{
//  valbutton04 = digitalRead(BUTTON4);
  valbutton05 = digitalRead(BUTTON5);
  valbutton06 = digitalRead(BUTTON6);// detect up to 6 chords ( or 6 notes, possible 6 string chord).
  valbutton07 = digitalRead(BUTTON7);
  valbutton08 = digitalRead(BUTTON8);

  valbutton09 = digitalRead(BUTTON9);
  valbutton10 = digitalRead(BUTTON10);
  valbutton11 = digitalRead(BUTTON11);// up to three seperate percussions
  valbutton12 = digitalRead(BUTTON12);//

  valbutton13 = digitalRead(BUTTON13);
  valbutton14 = digitalRead(BUTTON14);
  valbutton15 = digitalRead(BUTTON15);
  valbutton16 = digitalRead(BUTTON16);
//  valbutton17 = digitalRead(BUTTON17);
//  valbutton18 = digitalRead(BUTTON18);
//  valbutton19 = digitalRead(BUTTON19);

}

void scan_DigiPins () // get value of globle variables NB This new routine detached Inst Chord fromphysical Pin
Number
{
  //NB Fits loaction Left "spare so as to prevent miscounting)

//  DigiPins[2]  = digitalRead(BUTTON2);//Not used since the move to anague selection
//  DigiPins[3]  = digitalRead(BUTTON3);
//  DigiPins[4]  = digitalRead(BUTTON4);// Percussion to play

  DigiPins[5]  = digitalRead(BUTTON5);//spare
  DigiPins[6]  = digitalRead(BUTTON6);// detect up to 6 chords ( or 6 notes, possible 6 string chord).
  DigiPins[7]  = digitalRead(BUTTON7);
  DigiPins[8]  = digitalRead(BUTTON8);

  DigiPins[9]  = digitalRead(BUTTON9);
  DigiPins[10] = digitalRead(BUTTON10);// four seperate percussions
  DigiPins[11] = digitalRead(BUTTON11);
  DigiPins[12] = digitalRead(BUTTON12);//

  DigiPins[13] = digitalRead(BUTTON13);
  DigiPins[14] = digitalRead(BUTTON14);//
  DigiPins[15] = digitalRead(BUTTON15);
  DigiPins[16] = digitalRead(BUTTON16);
 //DigiPins[17] = digitalRead(BUTTON17);//
 //DigiPins[18] = digitalRead(BUTTON18);
 //DigiPins[19] = digitalRead(BUTTON19);
}
```

```
void scan_all_pins () // get value of glable variables
{
//  valbutton04 = digitalRead(BUTTON4);// Percussion to play
  valbutton05 = digitalRead(BUTTON5);//spare
  valbutton06 = digitalRead(BUTTON6);// detect up to 4 chords ( or 6 notes, possible 6 string chord).
  valbutton07 = digitalRead(BUTTON7);
  valbutton08 = digitalRead(BUTTON8);

  valbutton09 = digitalRead(BUTTON9);
  valbutton10 = digitalRead(BUTTON10);// four percs
  valbutton11 = digitalRead(BUTTON11);
  valbutton12 = digitalRead(BUTTON12);

  valbutton13 = digitalRead(BUTTON13);
  valbutton14 = digitalRead(BUTTON14);
  valbutton15 = digitalRead(BUTTON15);
  valbutton16 = digitalRead(BUTTON16);
//  valbutton17 = digitalRead(BUTTON17);
//  valbutton18 = digitalRead(BUTTON18);
//  valbutton19 = digitalRead(BUTTON19);

}
//======================== Select Instrument sub-routine ========
void selectInstrument(unsigned char group_num, int instrument_, byte inst_channel)
{
      talkMIDI(0xB0, 0x07, 120);
      if(group_num == GM1)
            talkMIDI(0xB0, inst_channel, 0x00); //Default bank GM1
      else talkMIDI(0xB0, inst_channel, 0x78); //Bank select drums  // , 0, is chanel number?
      talkMIDI(0xC0, instrument_, 0);
}

//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);
}

//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) {
  midiSerial.write(cmd);
  midiSerial.write(data1);

  //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)
```

```
    midiSerial.write(data2);
}
```