```
/* ~~~~~~~~~~~~~~~~~ MIDI Chord and Percusion Player
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 Functional Sections
 TOP
 1: Instrument Selction
 1.1: Ployphonic Instruments / also  Monophonic (tones)
 1.2: Percusion Instruments
 1.3 Metronome Instruments (These nee dto work in in pairs)
 V01 - metronome patch added
 V02 mode reworking added
 V04  3Aug14 Modal working now added
 V05. Need to add in metronome and associated instruments#
 V10: Metronome working with own instrumnet channnles and RGB slaved to beat sper min (to Comapre with input rate
 V12: Mode Change to 1:Percussion, 2:Chord, 3:Tone 4:Mmetronome with chords. 1, 2, 3 modes compare metronome rate with P1 rate with tick rate
 For the future could have tick - tock compared with P1 to P2 (say).

 When this is sorted add in Ch 10 drum sounds.
 Not to mention cleaning up code and getting RB btter matcheg to iRate in Mode 4, eg is the "30" offset causing the mismatch

 Fault Fix 5 Apr 2018, fixed sticking notes
 ==================TO BE DONE  ===== Now at the end of the file
 */

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

//midi sheild stuff
#include <SoftwareSerial.h>
byte resetMIDI = 4;/// fixed on the sparfun version ashardwired ~ unless you mount it away from the UNO
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 6 This will be used for Metronome / Rate Tick
#define midiChan7 0x96;//channel 7 This will be used for the time signature Tock
//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 ~ change these to more meaningful names later

int ledTriRed= 6;  // for RGB target measure 1Aug2014//const byte BUTTON2 = 2; // Assigned to instrument selection 00;01;10,11
int ledTriGreen= 5;
int ledTriBlue= 2;

//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; // Metronome OF /ON
const byte BUTTON10 = 10; // input for P1/C1/T1 [P1/A chord/ tone C
const byte BUTTON11 = 11; // input for P2/C1/T1 [P1/A chord/ tone C
const byte BUTTON12 = 12; // input for P3/C1/T1 [P1/A chord/ tone C
const byte BUTTON13 = 13; // input for P4/C1/T1 [P1/A chord/ tone C

//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; // 0101 input for mode select 00:Perc; 01 Invert Perc
const byte BUTTON19 = 19; // 0011 input for mode select 10: Chords ; 11 Tones

//== 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 metronome_note1 = 0x45; // notes needed if say a glocken spiel
int metronome_note2 = 0x50;  // might need to send note appropriate fro the instrument
int metronome_note3 = 0x55;
int metronome_note4 = 0x3C;
int metronome_note5 = 0x3C;
int metronome_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 = 0.999;
double logVol = .999;
double velo_preset = 0.999; // this is to peak limit an amplifier// Especiall LEPAI T-Class

// Percussion Xfer

int P1_count = 0;
int P1_Prev_count = 0;
int P2_count = 0;
int P2_Prev_count = 0;
int P3_count = 0;
int P3_Prev_count = 0;
int P4_count = 0;
int P4_Prev_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 metronome_type1 =taiko;
```

```
int metronome_type2 =timpani;

int prev_metronome_type1 =taiko;
int prev_metronome_type2 =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 valInstSelectPrev = 0;
int valMetroSelect = 0;
int valMetroSelectPrev = 0;
int valMetroRange = 0;
int valTempo = 0;

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; //PCT1 select
int valbutton11 = HIGH; //PCT2 select
int valbutton12 = HIGH; //PCT3 select
int valbutton13 = HIGH; //PCT4 select
//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;// mode select
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
byte DigiPins[19];
byte prevDigiPins[19];
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

// Time related stuff
int iSecond = 0;
int iPrevSecond = 0;
int iBeatRate = 0;
int iBeatCount =0;
int iBeatCountLast=0;
int iRate = 40 ; // range on a metrnome is 40 to 208, this will need to be prorated via a 10K linear pot
int iInterval = 500;// default 500 mSec actual 60000mSec/iRate
int iTimeSig = 0;// As a deafult, then 2,3,4  switch postion ~ for future coulb be OFF (no metro) One, simple betat,
them 1:2, 3/4, 4/4
int iSigCount =0; //for calculting time to "tocks"
int iSigCountLast = 0;
int RateP1; // how fast P1 switch goes o and off/
// if millsec now - milsec past is greater thaninterval then Tick
// count intervl, whilst sendoing ticks is on, count tick, evey 2nd, third, or forth, send a tock.
int metroOffOn = 0; //switch metrone in or out
int iTick = HIGH;  // Using internal pull ups, so HIGH = not pressed LOW i spressed
int iTock = HIGH;
int prev_iTick = HIGH;
int prev_iTock = HIGH;

int TsigPin1 = HIGH;
int TsigPin2 = HIGH;
int TsigValue = 0; // Rane, 0= Simple Beat, 1 = 2/2 (alt beats), 2 = 3/4 time, 3=4/4 time

//Mode select ============ Percussion, Inverse percussion, chords, notes

int ModePin1 = HIGH;
int ModePin2 = HIGH;
int ModeValue = 0; // Mode 1, 2, 3 , 4 = 00, 01, 10, 11


//============== SETUPS ================================================
void setup() {
  //== INTIALISE Communication with SD50 and assign initial channel voices
  Serial.begin(9600); // setup serialchannel for time stamping
```

```
//   Serial.begin(31250);// EXTERNAL midi, NOT AT SAME TIME AS pc INTERFACE
  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
// ==========for use as a diagnosti or for use with external synth
  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 riare used for Tx and Rx
  pinMode(ledTriBlue, OUTPUT);//Pin [2]is now switching Blue LED
  //  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);// mode select input
  pinMode(BUTTON19, INPUT);// mode select input


  // 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
  digitalWrite(BUTTON18, HIGH);
  digitalWrite(BUTTON19, HIGH);



  delay(100);// 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
  };


//============  Target Rate Setup

pinMode(ledTriRed, OUTPUT);
pinMode(ledTriGreen, OUTPUT);
pinMode(ledTriBlue, OUTPUT);
digitalWrite(ledTriRed, HIGH);  //
```

```
digitalWrite(ledTriGreen, HIGH);  //
digitalWrite(ledTriBlue, HIGH);  //

  //========== TIME RELATED Values ==========
ModeValue = 2; //set up value, default to percussion until spurious metronome on percussion sorted 2 Aug 14

  Serial.println();
  Serial.println("ModeValue, Pin_No, Direct, Mill, TSec, Sec, Min, BtsMin"); //  iSec"";"Min");
}
// ~~~~~~~~~~~~~~ MAIN PROGRAM ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
void loop() {

// TEST RGB)


  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 ======


//============= Mode select ========================


  if (valbutton19 == HIGH && valbutton18 == HIGH )
  { ModeValue = 4; } // Percussion Metronome is on TICK TOCK
  else if (valbutton19 == HIGH && valbutton18 == LOW)
  { ModeValue = 3; } //  Chords from 26Aug14 //Percussion, invert to TOCK TICKInverted percussion, straight for time being
  else if (valbutton19 == LOW && valbutton18 == HIGH)
  { ModeValue =2; } // Tones from 26Aug14 //Chords Std Metro
   else if(valbutton19 == LOW && valbutton18 == LOW)
  { ModeValue =1; } // Metronome and tones from 26Aug14 Tones, Chords for th emoment 2Aug 2014 Standard metro, think about chords pairs

// TBD 3Aug14 Cross wires over on Arduino P18 - P19, P19 -> P18

//=================================== End of Mode Val ============================




   valInstSelect = analogRead(1);

  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;
  }

// ========= PERCUSSION SELECT ======================
  valPercSelect = analogRead(0);
  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;
  }
// ========= PERCUSSION SELECT END ======================

//=================== Metronome Instrument Select ================

//int valMetroSelect = 0;
//int valMetroSelectPrev = 0;



valMetroSelect = analogRead(2);
  valMetroRange = valMetroSelect;

  if (valMetroRange <=100) // Switch Position 1
  {
    metronome_type1 = melo_tom; // initila set up to wood block = 115
    metronome_type2 = woodblock;

  //  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(valMetroRange <=250) // Switch Position 2
  {
    metronome_type1 = woodblock;
    metronome_type2 = tinkle_bell;

//    percussion_type1 = woodblock;
//    percussion_type2 = tinkle_bell;
//    percussion_type3 = agogo;
//    percussion_type4 = timpani;
  }

  else if(valMetroRange <=500) // Switch Position 3
```

```
  {
    metronome_type1 = timpani ;
    metronome_type2 = agogo;
 //   percussion_type1 = pitched_percussion ;
 //   percussion_type2 = reverse_cymbal;
 //   percussion_type3 = timpani;
//    percussion_type4 = taiko ;
  }
  else if(valMetroRange <=700) // Switch Position 4
  {
   metronome_type1 = timpani ;
   metronome_type2 = melo_tom;
//    percussion_type1 = timpani ;
//    percussion_type2 = melo_tom;
//    percussion_type3 = synth_drum;
//    percussion_type4 = agogo;
  }
  else if(valMetroRange <=900) // Switch Position 5
  {
   metronome_type1 = synth_drum;
   metronome_type2 = woodblock;
 //   percussion_type1 = tinkle_bell;
//    percussion_type2 = woodblock;
 //   percussion_type3 = agogo;
 //   percussion_type4 = synth_drum;
  }

  else if(valMetroRange <1050) // Switch Position 6
  {
    metronome_type1 = taiko ;
   metronome_type2 = agogo;


    //   percussion_type1 = agogo ;
  //  percussion_type2 = reverse_cymbal;
  //  percussion_type3 = timpani;
  //  percussion_type4 = melo_tom;
  }


//=================== Metronome Instrrument Select End ==========



  //  vol control

  logVol = 100;//analogRead(3); switched out digital voliume 30Jul14, using stero volume controlinstead
  velo_main = 100;//logVol/1024;


  //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);

  midiSerial.write(197); // select channel 5
  midiSerial.write(metronome_type1);
  midiSerial.write(198); // select channel 5
  midiSerial.write(metronome_type2);

/*  ONLY for use with externalMIDI device

  */
  // ==== 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



if (ModeValue ==1) // ====== Percusion plus standard metronome setup Tick Tick Tick Tock
   {

//================================== PERC 1 2 3 4  XMIT ROUTINES
================================
//new 9th May detect xfr P1 to P3 and P3 to P1
// ======================== Perc 1  ========= pin 10 ========================== P1 Button

   if (prevDigiPins[10] == HIGH && DigiPins[10] == 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[10] = HIGH;
         // P1_count = P1_count +1; // for percussion xfr velocity detect
          }
   else if (prevDigiPins[10] ==  HIGH && DigiPins[10] == LOW)  //ast two percussion
```

```
        {
         noteOn(1, percussion_note1, velocity_125 * velo_preset * velo_main);
         prevDigiPins[10] = LOW;

       //   Serial.print("ModeValue= ");
       //   Serial.print(", ");
          Serial.print(ModeValue);
          Serial.print(", ");
          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();

         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);
         }
  else if (prevDigiPins[10] == LOW && DigiPins[10] == HIGH)  //ast two percussion
         {
         noteOn(1, percussion_note1, velocity_0);
         prevDigiPins[10] = HIGH;
    //      Serial.print("ModeValue= ");
    //      Serial.print(", ");
          Serial.print(ModeValue);
          Serial.print(", ");
          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();
        }
  else if (prevDigiPins[10] == LOW && DigiPins[10] == LOW)  //ast two percussion
        {
        prevDigiPins[10] = LOW;
        }
//==================END OF ======= PERC10 XMIT ROUTINES
================================


//================================== PERC11 XMIT ROUTINES =======  P2 Button
   if (prevDigiPins[11] == HIGH && DigiPins[11] == 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[11] = HIGH;
          P2_count = P2_count +1; // for percussion xfr velocity detect
         }
  else if (prevDigiPins[11] ==  HIGH && DigiPins[11] == LOW)  //ast two percussion
        {
         noteOn(2, percussion_note2, velocity_125 * velo_preset * velo_main);
         prevDigiPins[11] = LOW;
   //        Serial.print("ModeValue= ");
   //        Serial.print(", ");
          Serial.print(ModeValue);
          Serial.print(", ");
          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();

         P2_count = P2_count +1; // for percussion xfr velocity detect
       //   delay(25);
        }
  else if (prevDigiPins[11] == LOW && DigiPins[11] == HIGH)  //ast two percussion
        {
         noteOn(2, percussion_note2, velocity_0);
```

```
        prevDigiPins[11] = HIGH;
    //      Serial.print("ModeValue= ");
    //      Serial.print(", ");
          Serial.print(ModeValue);
          Serial.print(", ");
          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();

        }
  else if (prevDigiPins[11] == LOW && DigiPins[11] == LOW)  //ast two percussion
        {
        prevDigiPins[11] = LOW;
        }
//====================END OF ======= PERC11 XMIT ROUTINES ======== P2
======================

//================================== PERC 3  XMIT ROUTINES ===== pin 12 ========== P3 Button
   if (prevDigiPins[12] == HIGH && DigiPins[12] == HIGH)  //ast two percussion
        {
      //    noteOn(0, percussion_note3, velocity_0);
      //    prevDigiPins[12] = HIGH;
          P3_count = P3_count +1; // for percussion xfr velocity detect
         }
   else if (prevDigiPins[12] ==  HIGH && DigiPins[12] == LOW)  //ast two percussion
        {
        noteOn(3, percussion_note3, velocity_125 * velo_preset * velo_main);
        prevDigiPins[12] = LOW;
    //      Serial.print("ModeValue= ");
    //      Serial.print(", ");
          Serial.print(ModeValue);
          Serial.print(", ");
          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();


         P3_count = P3_count +1; // for percussion xfr velocity detect
         }
  else if (prevDigiPins[12] == LOW && DigiPins[12] == HIGH)  //ast two percussion
        {
         noteOn(3, percussion_note3, velocity_0);
         prevDigiPins[12] = HIGH;
   //       Serial.print("ModeValue= ");
  //        Serial.print(", ");
          Serial.print(ModeValue);
          Serial.print(", ");
          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();

        }
  else if (prevDigiPins[12] == LOW && DigiPins[12] == LOW)  //ast two percussion
        {
         prevDigiPins[12] = LOW;
        }
 //==================END OF ======= PERC 3 XMIT ROUTINES ==== P3
==========================

 //================================ PERC 4 XMIT ROUTINES ===== pin 13 ============== P4
Button
  if (prevDigiPins[13] == HIGH && DigiPins[13] == HIGH)  //ast two percussion
        {
      //   noteOff(0, percussion_note4, velocity_0);
      //   prevDigiPins[13] = HIGH;
         P4_count = P4_count +1; // for percussion xfr velocity detect
        }
  else if (prevDigiPins[13] ==  HIGH && DigiPins[13] == LOW)  //ast two percussion
        {
```

```
      noteOn(4, percussion_note4, velocity_125 * velo_preset * velo_main);
      prevDigiPins[13] = LOW;
 //       Serial.print("ModeValue= ");
 //       Serial.print(", ");
        Serial.print(ModeValue);
        Serial.print(", ");
        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();

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

      }
 else if (prevDigiPins[13] == LOW && DigiPins[13] == HIGH)  //ast two percussion
      {
      noteOff(4, percussion_note4, velocity_0);
      prevDigiPins[13] = HIGH;
  //       Serial.print("ModeValue= ");
 //       Serial.print(", ");
        Serial.print(ModeValue);
        Serial.print(", ");
        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();

      }
else if (prevDigiPins[13] == LOW && DigiPins[13] == LOW)  //ast two percussion
      {
```

```
        prevDigiPins[13] = LOW;
        }
 //==================END OF ======= PERC 4 XMIT ROUTINES ==== P4
========================


   }

else if (ModeValue ==2) // Chords, Std Metro
  {

  // ======== 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] ;
    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_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[10] = LOW;
         Serial.print(ModeValue);
         Serial.print(", ");
         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();

   //delay(50);
```

```
 }

 else if (prevDigiPins[10] == LOW && DigiPins[10] == 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[10] = HIGH;
         Serial.print(ModeValue);
         Serial.print(", ");
         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();
  }

  else if (prevDigiPins[10] == LOW && DigiPins[10] == LOW)  //ast two percussion
  {
   prevDigiPins[10] = LOW;
  }
  //======================END OF ======= A CHORD
================================================

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

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

  else if (prevDigiPins[11] == HIGH && DigiPins[11] == 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[11] = LOW;
        Serial.print(ModeValue);
        Serial.print(", ");
        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();

 // delay(50);
  }
else if (prevDigiPins[11] == LOW && DigiPins[11] == 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[11] = HIGH;
        Serial.print(ModeValue);
        Serial.print(", ");
        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();
```

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

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


  //====================== BEGIN ======= D CHORD  ===== pin 12
=========================================

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

  else if (prevDigiPins[12] == HIGH && DigiPins[12] == 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[12] = LOW;
         Serial.print(ModeValue);
         Serial.print(", ");
         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();

  // delay(50);
   }
```

```
 else if (prevDigiPins[12] == LOW && DigiPins[12] == 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[12] = HIGH;
        Serial.print(ModeValue);
        Serial.print(", ");
        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();

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

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


 //====================== BEGIN ======= G CHORD  === pin 13
============================================

 if (prevDigiPins[13] == HIGH && DigiPins[13] == HIGH)  //ast two percussion
   {
  prevDigiPins[13] = HIGH;
   }

 else if (prevDigiPins[13] == HIGH && DigiPins[13] == 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[13] = LOW;
        Serial.print(ModeValue);
        Serial.print(", ");
        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();

 // delay(50);
  }
else if (prevDigiPins[13] == LOW && DigiPins[13] == 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[13] = HIGH;
        Serial.print(ModeValue);
        Serial.print(", ");
        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();

    }

 else if (prevDigiPins[13] == LOW && DigiPins[13] == LOW)  //ast two percussion
    {
    prevDigiPins[13] = LOW;
    }
  //====================== END ======= G CHORD
==============================================
  // ===========END OFF ====== 4 Chord Xmit Routines ================================


  }

else if (ModeValue ==3) // Tones plus Std Metro
  {
  // ======== Begin  ========= 4 Tone Xmit Routines ================================

  //============================= Tone C ================================
  //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] ;
    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, note60, velocity_125 * velo_preset * velo_main);
    prevDigiPins[10] = LOW;
        Serial.print(ModeValue);
        Serial.print(", ");
        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();

    }

  else if (prevDigiPins[10] == LOW && DigiPins[10] == HIGH)  //ast two percussion
    {
```

```
  ChordNoteOn(0x90, note60, velocity_0);
   prevDigiPins[10] = HIGH;
        Serial.print(ModeValue);
        Serial.print(", ");
        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();
   }

 else if (prevDigiPins[10] == LOW && DigiPins[10] == LOW)  //ast two percussion
   {
  prevDigiPins[10] = LOW;
   }
 //======================END OF Tone C  ============================================

 //======================BEGIN ======= Tone B
==============================================

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

   }

 else if (prevDigiPins[11] == HIGH && DigiPins[11] == LOW)  //ast two percussion
   {
  ChordNoteOn(0x90, note59, velocity_125 * velo_preset * velo_main);
  prevDigiPins[11] = LOW;
        Serial.print(ModeValue);
        Serial.print(", ");
        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();

   }
  else if (prevDigiPins[11] == LOW && DigiPins[11] == HIGH)  //ast two percussion
   {
   ChordNoteOn(0x90, note59, velocity_0);
    prevDigiPins[11] = HIGH;
        Serial.print(ModeValue);
        Serial.print(", ");
        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();

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

  //====================== END OF ======= B
==============================================


  //====================== BEGIN ======= D CHORD  ===== pin 12
=========================================

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

  else if (prevDigiPins[12] == HIGH && DigiPins[12] == LOW)  //ast two percussion
   {
   ChordNoteOn(0x90, note57, velocity_75 * velo_preset * velo_main);
   prevDigiPins[12] = LOW;
        Serial.print(ModeValue);
        Serial.print(", ");
```

```
        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();
  }

 else if (prevDigiPins[12] == LOW && DigiPins[12] == HIGH)  //ast two percussion
   {
  ChordNoteOn(0x90, note57, velocity_0);
   prevDigiPins[12] = HIGH;
        Serial.print(ModeValue);
        Serial.print(", ");
        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();

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

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


 //====================== BEGIN ======= G CHORD  === pin 13
==========================================
```

```
  if (prevDigiPins[13] == HIGH && DigiPins[13] == HIGH)  //ast two percussion
    {
    prevDigiPins[13] = HIGH;
    }
// Note C
  else if (prevDigiPins[13] == HIGH && DigiPins[13] == LOW)  //ast two percussion
    {
    ChordNoteOn(0x90, note55, velocity_125 * velo_preset * velo_main);
    prevDigiPins[13] = LOW;
          Serial.print(ModeValue);
          Serial.print(", ");
          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();

    }
  else if (prevDigiPins[13] == LOW && DigiPins[13] == HIGH)  //ast two percussion
    {
    ChordNoteOn(0x90, note55, velocity_0);
    prevDigiPins[13] = HIGH;
          Serial.print(ModeValue);
          Serial.print(", ");
          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();

    }
```

```
  else if (prevDigiPins[13] == LOW && DigiPins[13] == LOW)  //ast two percussion
    {
      prevDigiPins[13] = LOW;
    }
  //====================== END ======= G CHORD
================================================
  // ===========END OFF ====== 4 Tone Xmit Routines ==================================

  }//end of ModeValue3



else if (ModeValue ==4) // percussion plus inverted metronme Tock Tock Tock Tick
  {

// Straight Metronome routine
    // ======== Begin  ========= 4 Tone Xmit Routines ================================

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

  else if (prevDigiPins[10] == HIGH && DigiPins[10] == LOW)  //ast two percussion
    {
    ChordNoteOn(0x90, note60, velocity_125 * velo_preset * velo_main);
    prevDigiPins[10] = LOW;
          Serial.print(ModeValue);
          Serial.print(", ");
          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();

    }

  else if (prevDigiPins[10] == LOW && DigiPins[10] == HIGH)  //ast two percussion
    {
    ChordNoteOn(0x90, note60, velocity_0);
```

```
   prevDigiPins[10] = HIGH;
        Serial.print(ModeValue);
        Serial.print(", ");
        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();
   }

 else if (prevDigiPins[10] == LOW && DigiPins[10] == LOW)  //ast two percussion
   {
  prevDigiPins[10] = LOW;
   }
 //======================END OF Tone C  ================================================

 //======================BEGIN ======= Tone B
==================================================

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

   }

 else if (prevDigiPins[11] == HIGH && DigiPins[11] == LOW)  //ast two percussion
   {
  ChordNoteOn(0x90, note59, velocity_125 * velo_preset * velo_main);
  prevDigiPins[11] = LOW;
        Serial.print(ModeValue);
        Serial.print(", ");
        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();

   }
  else if (prevDigiPins[11] == LOW && DigiPins[11] == HIGH)  //ast two percussion
   {
   ChordNoteOn(0x90, note59, velocity_0);
    prevDigiPins[11] = HIGH;
        Serial.print(ModeValue);
        Serial.print(", ");
        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();

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

  //====================== END OF ======= B
================================================


  //====================== BEGIN ======= D CHORD  ===== pin 12
=========================================

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

  else if (prevDigiPins[12] == HIGH && DigiPins[12] == LOW)  //ast two percussion
   {
   ChordNoteOn(0x90, note57, velocity_75 * velo_preset * velo_main);
   prevDigiPins[12] = LOW;
        Serial.print(ModeValue);
        Serial.print(", ");
        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();
  }

 else if (prevDigiPins[12] == LOW && DigiPins[12] == HIGH)  //ast two percussion
   {
  ChordNoteOn(0x90, note57, velocity_0);
   prevDigiPins[12] = HIGH;
        Serial.print(ModeValue);
        Serial.print(", ");
        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();

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

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


 //====================== BEGIN ======= G CHORD  === pin 13
==========================================

 if (prevDigiPins[13] == HIGH && DigiPins[13] == HIGH)  //ast two percussion
```

```
    {
    prevDigiPins[13] = HIGH;
    }
// Note C
  else if (prevDigiPins[13] == HIGH && DigiPins[13] == LOW)  //ast two percussion
    {
    ChordNoteOn(0x90, note55, velocity_125 * velo_preset * velo_main);
    prevDigiPins[13] = LOW;
          Serial.print(ModeValue);
          Serial.print(", ");
          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();

    }
  else if (prevDigiPins[13] == LOW && DigiPins[13] == HIGH)  //ast two percussion
    {
    ChordNoteOn(0x90, note55, velocity_0);
    prevDigiPins[13] = HIGH;
          Serial.print(ModeValue);
          Serial.print(", ");
          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();

  }

  else if (prevDigiPins[13] == LOW && DigiPins[13] == LOW)  //ast two percussion
```

```
  {
   prevDigiPins[13] = LOW;
  }
  //====================== END ======= G CHORD
================================================
  // ===========END OFF ====== 4 Tone Xmit Routines ================================


  } //  End of Mode Value 4


// ============= Metronome Xmit Routine ====================================
/*  for ref
int iSecond = 0;
int iPrevSecond = 0;
int iBeatRate = 0;
int iBeatCount =0;
int iBeatCountLast=0;
int iRate = 40 ; // range on a metrnome is 40 to 208, this will need to be prorated via a 10K linear pot
int iInterval = 500;// default 500 mSec actual 60000mSec/iRate
int iTimeSig = 0;// As a deafult, then 2,3,4  switch postion ~ for future coulb be OFF (no metro) One, simple betat,
them 1:2, 3/4, 4/4
int iSigCount =0; //for calculting time to "tocks"
int iSigCountLast = 0;

*/

// detect second casnge to figure out the RateP1
    iSecond =millis()/1000;
  if (iSecond - iPrevSecond >1)
     {
       iPrevSecond = iSecond;
       RateP1 = P1_count-P1_Prev_count;
       P1_Prev_count = P1_count;
     }

  iTick=HIGH; //Rest tick tocks
  iTock=HIGH;
  valTempo = analogRead(3);
  iBeatCount =  millis();// check how much tim ehas gone past
  iRate = 30+(valTempo/3); // for test purpose
//  iTimeSig= 2;   // 0, no tck, 1 gives alt tick tock 2 give 3/4, 3 gives 4/4 time  (1, 2, 3, 4 are derived from rotary switch
                 // th ezero is derived from the switch set both swithc O/P high regardles of position

  iInterval = 60000/iRate; // determin etim einterval fo reach beatthis let su sknow many mille seconds goes by before a
"tick" is sent out.

  if (iBeatCount-iBeatCountLast >= iInterval)
  {
    iBeatCountLast = iBeatCount;
    iTick = LOW; //  coul dbe seapped oubt with a reverse pattern switch
    iSigCount = iSigCount+1; // for time signature calculations
     Serial.print("A: iBeatCount= ");
     Serial.print(iBeatCount);
```

```
    Serial.print(" iSigCount= ");
    Serial.print(iSigCount);
    Serial.print(" iTick= ");
    Serial.print(iTick);
    Serial.print(" iTock= ");
    Serial.print(iTock);
    Serial.print(" iTock= ");
    Serial.print(iTock);
    Serial.println();
  }
 else
 {
   iTick = HIGH;
 }
  // End of Beat counting routin

 if (iSigCount-iSigCountLast >= iTimeSig) // Time sig routing
 {
  iSigCountLast = iSigCount;
  iTick = HIGH; //Knockout main beat when tOCK is sent

 if (iTimeSig>0)  // no tock if time sig = 0
   {
    iTock = LOW;  // no tick when a toack is being sent
    Serial.print("B: iBeatCount= ");
    Serial.print(iBeatCount);
    Serial.print(" iSigCount= ");
    Serial.print(iSigCount);
    Serial.print(" iTick= ");
    Serial.print(iTick);
    Serial.print(" iTock= ");
    Serial.print(iTock);
    Serial.println();
   }
   else
   {
    iTock = HIGH;    // end of sim esig routine
   }
 }
// Work out howfast P1 is being operated
//P1_count


// TEST Line needed to switch metronome function on and off

// ============== Tick Xmit  Routine ========================================


  if (prev_iTick == HIGH && iTick == 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
     //    prev_iTick = iTick;
```

```
    //    P1_count = P1_count +1; // for percussion xfr velocity detect
        }
  else if (prev_iTick ==  HIGH && iTick == LOW)  //ast two percussion
        {
         noteOn(5, metronome_note1, velocity_125 * velo_preset * velo_main); // need to chane percussion note to
metro note
         prev_iTick = LOW;

        }
  else if (prev_iTick == LOW && iTick == HIGH)  //ast two percussion
        {
         noteOn(5, metronome_note1, velocity_0);
         prev_iTick = HIGH;

        }
   else if (prev_iTick == LOW && iTick == LOW)  //ast two percussion
        {
        prev_iTick = LOW;
        }

// ========End of  Tick Xmit Routine ========================================

// ======= ====== Tock Xmit Routine ==========================================

  if (prev_iTock == HIGH && iTock == HIGH)  //ast two percussion
        {
      //    prev_iTock = HIGH;
         }
   else if (prev_iTock ==  HIGH && iTock == LOW)  //ast two percussion
        {
         noteOn(6, metronome_note2, velocity_125 * velo_preset * velo_main); // need to chane percussion note to
metro note
         prev_iTock = LOW;
        }
  else if (prev_iTock == LOW && iTock == HIGH)  //ast two percussion
        {
         noteOn(6, metronome_note2, velocity_0);  //NB '6' is th echannel number
         prev_iTock = HIGH;

        }
   else if (prev_iTock == LOW && iTock == LOW)  //ast two percussion
        {
        prev_iTock = LOW;
        }



// ============= End of Tock Transmit routine ================================



// ============== End of Metronome Transmit routine ========================

// ====================== Time Sig Select plus Metronome ON/OFF
```

```
 if (valbutton08 == HIGH && valbutton07 == HIGH )
 { iTimeSig = 4; } // Percussion Metronome is on TICK TOCK
 else if (valbutton08 == HIGH && valbutton07 == LOW)
 { iTimeSig = 3; } //  Percussion, invert to TOCK TICKInverted percussion, straight for time being
 else if (valbutton08 == LOW && valbutton07 == HIGH)
 { iTimeSig =2; } // Chords Std Metro
  else if(valbutton08 == LOW && valbutton07 == LOW)
 { iTimeSig =1; } // Tones, Chords for th emoment 2Aug 2014 Standard metro, think about chords pairs

 metroOffOn = valbutton09;

//if (ModeValue ==4)
 //{
 if (metroOffOn == HIGH) // check if metrnome function is switched on
    {
      iTimeSig = 0; //Turn off Metronome
      digitalWrite(ledTriRed, HIGH);  // No RGB OutPut if Metromoe is off.
      digitalWrite(ledTriGreen, HIGH);  //
      digitalWrite(ledTriBlue, HIGH);  //
    }
   else
      {// metrnome is but can be used in two ways

    if (ModeValue ==4) // As a standard metrnome in mode 4
      {
      //
        if (iRate < 60)
        {
         digitalWrite(ledTriRed, LOW);  // No RGB OutPut if Metromoe is off.
         digitalWrite(ledTriGreen, HIGH);  //
         digitalWrite(ledTriBlue, HIGH);  //
        } //
         else if (iRate > 200)
        {
         digitalWrite(ledTriRed, HIGH);  // No RGB OutPut if Metromoe is off.
         digitalWrite(ledTriGreen, HIGH);  //
         digitalWrite(ledTriBlue, LOW);  //
         }
         else
         {
         digitalWrite(ledTriRed, HIGH);  // No RGB OutPut if Metromoe is off.
         digitalWrite(ledTriGreen, LOW);  //
         digitalWrite(ledTriBlue, HIGH);  //
        }
      }// end of use as a standard metronome in mode 4

   else if (ModeValue == 1 || 2 || 3)
         {

          if (RateP1 <  0.05*iRate)
          {
          digitalWrite(ledTriRed, LOW);  // No RGB OutPut if Metromoe is off.
```

```
          digitalWrite(ledTriGreen, HIGH);  //
          digitalWrite(ledTriBlue, HIGH);
          }

         else if (RateP1 >  0.2*iRate)
          {
          digitalWrite(ledTriRed, HIGH);  // No RGB OutPut if Metromoe is off.
          digitalWrite(ledTriGreen, HIGH);  //
          digitalWrite(ledTriBlue, LOW);
          }

          else
          {
          digitalWrite(ledTriRed, HIGH);  // No RGB OutPut if Metromoe is off.
          digitalWrite(ledTriGreen, LOW);  //
          digitalWrite(ledTriBlue, HIGH);

          }
         }
 }
 // }// End of mode select wrt to RGB mode
  // ======================  End of Time Sig Select plus Metronome ON/OFF



}//  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);
}

void noteOnCh05(int cmd05, int pitch, int velocity)
{
  Serial.write(cmd05);
  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 )
{
  noteOnCh05(0x94, chordnote1, 0x00); // Fault Fix, clearnotes pasting from Ch04 corrected x93 to x94
  noteOnCh05(0x94, chordnote2, 0x00); // and Ch04 to Ch05  Clears "sticking notes
  noteOnCh05(0x94, chordnote3, 0x00);
  noteOnCh05(0x94, chordnote4, 0x00);
  noteOnCh05(0x94, chordnote5, 0x00);
  noteOnCh05(0x94, chordnote6, 0x00);
}

void scan_DigiPins () // get value of globle variables NB This new routine detached Inst Chord fromphysical Pin
Number
{

  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[18] = digitalRead(BUTTON18);
  DigiPins[19] = digitalRead(BUTTON19);
}

void scan_all_pins () // get value of glable variables
{
  //  valbutton06 = digitalRead(BUTTON6);// detect up to 4 chords ( or 6 notes, possible 6 string chord).
  valbutton07 = digitalRead(BUTTON7);// 7 & 8 give time sig, nono, 2/2/ 3/4. 4/4
  valbutton08 = digitalRead(BUTTON8);

  valbutton09 = digitalRead(BUTTON9);//metronome OFF/ON
```

```
  valbutton10 = digitalRead(BUTTON10);// four percs
  valbutton11 = digitalRead(BUTTON11);
  valbutton12 = digitalRead(BUTTON12);
  valbutton13 = digitalRead(BUTTON13);

  valbutton18 = digitalRead(BUTTON18);   //mode select
  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); remmed out 1Aug, causing complile error, not in use at present anyway
 } */

//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);
}



/*
============== TO BE DOEN BUGS and Improvements==========

 20 July 2014: Wedge-Medi design.Check level of use of inputss, emough for velocity sensing?

 TBD: Remap input and Outputs fot the Medi-Wedge box

 D0: Rx MIDI : TX is used for testing with Roland Sound Canvas
 D1: Tx MIDI : and also for Input to a standard MIDI keyboard.
```

D2: Blue LED Lead
D3: Music Sheild Tx-MIDI Inpt to Sparkfun
D4: Reset for for Sparkfun Music Sheild.

D5: Mode Seletc (Binary 00, 01, 10, 11)
D6: Mode Slect fro Percussion, Chords, Tones, ?

D7: Time Sig Slelct: Clock Tick: 4:4. 3:4, 6:8
D8: Time Sig Select; 4 Modes 00, 01, 10, 11

D9: Metronome ON/Off

D10: P1, P2, P3, P4: C1, C2, C3, C4, T1, T2, T3, T4 : Set circlip to limit to 3 modes.
D11: Ditto ~ P2
D12: Ditto ~ P3
D13: Ditto ~ P4

Analoque Sections
A0: Percusion [6 selctions]
A1: Instrument Select: [4] Limit on circlip
A2: Metronome Instrument: 4 paired ranges, might need addtionla channels, ie. ting, toch
A3: Metronome rate

A4 (D17):Red ~ Trti Colour LED, Below, On, Abov etarget rate.
A5 (D18): Green
D2: Blue
Reset: Reset

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, volum etuning 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 slect moved over to DigiPin[] arrays.

*/