```
// Cleo7_IR_control_with_lcd_bigcrystal_v1a - Doede Douma start project 2020-06-08

// History v1a,b,c,d,e,f - First Drafts
// History v2a - working version with the new "China" remote control (see for details the Splan File)
// History v2b - cleaned working version, ready for any new ideas
// History v2c - Added external refence of 2,5 Volt and adjusted thecode to the real measuerd values of the TEST voltages
// History v2d - Debugging small stuff and adding real volt measurement when Mouse is pressed (next to % bias) - also did calibration and added separate corrections
for L & R bias reading


/*
=============================================================================
  This Project controls input & output selections and volume control
  Contrast and tness control for LCD is included
  Power supply voltages can be checked to display % of bias point
  Code and pin layout is for Arduino MEGA 2560 r3
=============================================================================
  Remote Control China version:

  <OnOff>       . Mute              - code  1FE817E
  <Menu>        . LCD Menu          - code  1FEC13E
  <Home>        . Reset LCD in Menu - code  1FECE31
  <Up>          . Tape Select +     - code  1FE1CE3
  <Down>        . Tape Select -     - code  1FE02FD
  <Left>        . Source Select +   - code  1FEEC13
  <Right>       . Source Select -   - code  1FE9C63
  <OK>          . Mute              - code  1FEC837
  <Return>      . Big Small toggle  - code  1FEE41B
  <VolumeDown>  . Volume Down       - code  1FE916E
  <VolumeUp>    . Volume Up         - code  1FEE11E
  <Mouse>       . BIAS Check        - code  1FE12ED

  During the "Menu" there are sub menus
  Left / Right    Contrast
  Up/Down         Brightness
  Home           Back to initial values (contrast 100 and Brightness 50%)
=============================================================================
  LCD circuit:
  LCD "RW" pin to      GND
  LCD "Vo" pin to      digital pin 06 (via 1 kOhm and 47 uF LP filter)
  LCD "RS" pin to      digital pin 07
  LCD "E"  pin to      digital pin 08
  LCD D4   pin to      digital pin 09
  LCD D5   pin to      digital pin 10
  LCD D6   pin to      digital pin 11
  LCD D7   pin to      digital pin 12
  LCD LED+ pin to      digital pin 13 (via 100 Ohm)
=============================================================================
  Power supply testing
```

```
 Ub   must be between 420V and 500V 100% equals 460V
 Uout must be between 180V and 240V 100% equals 210V
 Ugk  must be between   7V and  10V 100% equals 8.5V
 Ub   A0-A1 L-R
 Uout A2-A3 L-R
 Ugk  A4-A5 L-R
=============================================================================
 20/20 Rotary encoder is used to do up down volume
 24 relays are driven by outputs to switch corresponding volume
 PIN C & E connect to logal GND
 PIN A        digital pin 2
 PIN B        digital pin 3
 PIN D       digital pin 4 - Push button function of rotary: active LOW
=============================================================================
 Two toggle switch at Front:
 Switches are active "LOW"
 Source up    digital pin 14
 Source down  digital pin 14
 Tape   up    digital pin 14
 Source down  digital pin 14
=============================================================================
 Source IN selected by <left> & <right>:
 DDDAC 1794 digital pin  48 - High = select Relay switches on via ULN2003AN
 Sony SACD  digital pin  49 - High = select Relay switches on via ULN2003AN
 Phonodude  digital pin  50 - High = select Relay switches on via ULN2003AN
 Revox B77  digital pin  51 - High = select Relay switches on via ULN2003AN
 Revox PR99 digital pin  52 - High = select Relay switches on via ULN2003AN
 Aux       digital pin  53 - High = select Relay switches on via ULN2003AN

 Tape OUT  selected by <Up> & <Down>:
 DDDAC 1794 digital pin A10 - High = select Relay switches on via ULN2003AN
 Sony SACD  digital pin A11 - High = select Relay switches on via ULN2003AN
 Phonodude  digital pin A12 - High = select Relay switches on via ULN2003AN
 Revox B77  digital pin A13 - High = select Relay switches on via ULN2003AN
 Revox PR99 digital pin A14 - High = select Relay switches on via ULN2003AN
 Aux       digital pin A15 - High = select Relay switches on via ULN2003AN
=============================================================================
 Volume control is by 2x 24 relays for parallel attenuator
 Digital pins D24-D47
=============================================================================
 Display example 20x4
01234567890123456789
 Volume : - 58,0 dB
 Source : abcdefghijk
 Tape   : DDDAC 1794
          Sony SACD
          PhonoDude
          Revox B77
          Revox PR99
          Aux Input
          None Select
  Select : Source In
          Tape Out
============================================================================
*/


// Include Libraries being used and settings for it
#include <IRremote.h>                     // Use IR Library
#include <BigCrystal.h>                   // Special Library to print BIG Characters
#include <LiquidCrystal.h>                // include the LCD library code:
LiquidCrystal lcd(7, 8, 9, 10, 11, 12);   // My Pin Layout on MEGA 2560
BigCrystal bigCrystal(&lcd);              // Embed bigCrystal in the LCD Library
IRrecv irrecv(22);                        // IR Pin Setting for TSOP 31238 (PIN 22) - Use IR Function
decode_results results;                   // Function decoding IR string

// set pin numbers and all variables for all external connections and calculations and If statements.................
// pins for LCD and IR are set already above in the library settings
const byte Ub_test_L          =  A0;   // Analog input to read Power supply. Connect to  Vb
const byte Ub_test_R          =  A1;   // Analog input to read Power supply. Connect to  Vb
const byte Uout_test_L        =  A2;   // Analog input to read Power supply. Connect to  Vout
const byte Uout_test_R        =  A3;   // Analog input to read Power supply. Connect to  Vout
const byte Ugk_test_L         =  A4;   // Analog input to read Power supply. Connect to  Vgk
const byte Ugk_test_R         =  A5;   // Analog input to read Power supply. Connect to  Vgk
const byte tape_dddac         = A10;   // digital output to drive Relay and select for tape  out this source
const byte tape_sacd          = A11;   // digital output to drive Relay and select for tape  out this source
const byte tape_phono         = A12;   // digital output to drive Relay and select for tape  out this source
const byte tape_B77           = A13;   // digital output to drive Relay and select for tape  out this source
const byte tape_PR99          = A14;   // digital output to drive Relay and select for tape  out this source
const byte tape_aux           = A15;   // digital output to drive Relay and select for tape  out this source
const byte source_dddac       =  53;   // digital output to drive Relay and select for source out this source
const byte source_sacd        =  52;   // digital output to drive Relay and select for source out this source
const byte source_phono       =  51;   // digital output to drive Relay and select for source out this source
const byte source_B77         =  50;   // digital output to drive Relay and select for source out this source
const byte source_PR99        =  49;   // digital output to drive Relay and select for source out this source
const byte source_aux         =  48;   // digital output to drive Relay and select for source out this source
const byte enc_B              =   2;   // Encoder Pin B  20/20 Encoder ALPS
const byte enc_A              =   3;   // Encoder pin A  20/20 Encoder ALPS
const byte enc_push           =   4;   // Encoder Pin D  Push Button active LOW
const byte brightness_out     =   6;   // PWM output brightness via 1k/47uF LP Filter (on mainboard)
const byte contrast_out       =  13;   // PWM output brightness via 100 Ohm (on mainboard)
const byte select_source_up   =  14;   // Input:  connected to panel switch select "source" up   - action is push to LOW
const byte select_source_down =  15;   // Input:  connected to panel switch select "source" down - action is push to LOW
const byte select_tape_up     =  16;   // Input:  connected to panel switch select "source" up   - action is push to LOW
const byte select_tape_down   =  17;   // Input:  connected to panel switch select "source" down - action is push to LOW
const byte volume_step_01     =  26;   // Output Volume Step  1 (min ouput)
const byte volume_step_02     =  29;   // Output Volume Step  2
const byte volume_step_03     =  32;   // Output Volume Step  3
const byte volume_step_04     =  35;   // Output Volume Step  4
const byte volume_step_05     =  38;   // Output Volume Step  5
const byte volume_step_06     =  41;   // Output Volume Step  6
const byte volume_step_07     =  44;   // Output Volume Step  7
const byte volume_step_08     =  47;   // Output Volume Step  8
const byte volume_step_09     =  25;   // Output Volume Step  9
const byte volume_step_10     =  28;   // Output Volume Step 10
const byte volume_step_11     =  31;   // Output Volume Step 11
const byte volume_step_12     =  34;   // Output Volume Step 12
const byte volume_step_13     =  37;   // Output Volume Step 13
const byte volume_step_14     =  40;   // Output Volume Step 14
const byte volume_step_15     =  43;   // Output Volume Step 15
const byte volume_step_16     =  46;   // Output Volume Step 16
const byte volume_step_17     =  24;   // Output Volume Step 17
const byte volume_step_18     =  27;   // Output Volume Step 18
const byte volume_step_19     =  30;   // Output Volume Step 19
const byte volume_step_20     =  33;   // Output Volume Step 20
const byte volume_step_21     =  36;   // Output Volume Step 21
const byte volume_step_22     =  39;   // Output Volume Step 22
const byte volume_step_23     =  42;   // Output Volume Step 23
const byte volume_step_24     =  45;   // Output Volume Step 24 (max ouput)
const long bias_check_Ub_L    = 372;   // Real test voltage is  930mV -> Code from 0-1023 will be - This is to calculate % performance of voltage measurement - Aref
= 2.503 Volt
const long bias_check_Uout_L  = 365;   // Real test voltage is  915mV -> Code from 0-1023 will be - This is to calculate % performance of voltage measurement
const long bias_check_Ugk_L   = 503;   // 499Real test voltage is 1215mV -> Code from 0-1023 will be - This is to calculate % performance of voltage measurement
const long bias_check_Ub_R    = 375;   // Same for Right channel
const long bias_check_Uout_R  = 362;
const long bias_check_Ugk_R   = 497;

// Setting codes for the China remote by the kind of button on the remote
const long OnOff       = 0x1FE817E;    // Mute               IR_Value =  0
const long Menu        = 0x1FEC13E;    // LCD Menu           IR_Value =  1
const long Home        = 0x1FECE31;    // Reset LCD in Menu  IR_Value =  2
const long Up          = 0x1FE1CE3;    // Tape Select +      IR_Value =  3
const long Down        = 0x1FE02FD;    // Tape Select -      IR_Value =  4
const long Left        = 0x1FEEC13;    // Source Select +    IR_Value =  5
const long Right       = 0x1FE9C63;    // Source Select -    IR_Value =  6
const long OK          = 0x1FEC837;    // Mute               IR_Value =  7
const long Return      = 0x1FEE41B;    // Big Small toggle   IR_Value =  8
const long VolumeDown  = 0x1FE916E;    // Volume Down        IR_Value =  9
const long VolumeUp    = 0x1FEE11E;    // Volume Up          IR_Value = 10
const long Mous        = 0x1FE12ED;    // Run Bias check     IR_Value = 11
const long Repeat      = 0xFFFFFFF;    // Repeat function

// set some Global variables
byte    mute_hard_status =   0;        // Mute status: 0 is "off"  1 is "on" for -99dB
byte    menu_status      =   0;        // Help variable so that the code knows if the "Menu" is active 0= not active, 1= active
int     source_status    =   0;        // Start with "nothing" 0 = nothing 1 = DDDAC, 2 = SACD etc etc 7 = "nothing" again
int     tape_status      =   0;        // Start with "nothing" 0 = nothing 1 = DDDAC, 2 = SACD etc etc
```

```
byte    big_status       =   0;        // big_status = 1 means big num display is active - Normal display = 0
long    bias_counter     =   0;        // Help variable for counting till it is time to do a bias check (every 5 or 10 minutes or so)
byte    bias_is_off      =   0;        // Help variable which tells if any of the Bias is wrong?
byte    IR_Value         =  99;        // Help variable for remembering IR code - start with "reset code"
int     volume_old       =   1;        // Volume start with 1= minimum -99dB
int     volume_new       =   1;        // Value in these variables are the step number
int     volume_change    =   0;        // Increments for volume volume starts 0 - can be also 1 or -1
float   brightness       = 250;        // Variable to set analog output = start value = 50% (half of value of 200)
float   contrast         =  40;        // (10 times higher) Variable to set analoge output = normal start value = 520, - JUMBO version starts much lower
byte    push_input       =   9;        // Front pannel push button encoder
byte    push_source_up   =   9;        // Front pannel push button source up
byte    push_source_down =   9;        // Front pannel push button source down
byte    push_tape_up     =   9;        // Front pannel push button tape up
byte    push_tape_down   =   9;        // Front pannel push button tape down
long    read_Ub_L;                     // Read Ub   voltage 0-1024 value
long    read_Uout_L;                   // Read Uout voltage
long    read_Ugk_L;                    // Read Ugk  voltage
long    read_Ub_R;                     // Same for right channel:
long    read_Uout_R;
long    read_Ugk_R;
int     percent_Ub_L;                  // calculate the percentage of where voltage is versus 100% bias point
int     percent_Uout_L;
int     percent_Ugk_L;
int     percent_Ub_R;                  // Same for right channel
int     percent_Uout_R;
int     percent_Ugk_R;
char    * BIG_Char;                    // This is the Character Array which contains the text or numbers to print in the BIG function
String  String_for_CHAR;               // This is used to convert a constructed String for the BIG function with: BIG_Char = String_for_CHAR.c_str();
byte    big_row_position;              // Defines if being printed on row 0 or row 1
String  lcd_vol     = " - xx.x  dB ";  // String variable for LCD volume in dB start at minimum level
String  lcd_vol_big = "-xx.x";         // Help variable for printing BIG nums - Short version of lcd_vol
String  lcd_source  = "None       ";   // Help variable for printing source Info
String  lcd_tape    = "None       ";   // Help variable for printing source Info
String  lcd_bias    = "Warming Up ";   // Help variable for printing source Info
unsigned char encoder_A;               // Help Variable for Rotary Encoder
unsigned char encoder_B;               // Help Variable for Rotary Encoder
unsigned char encoder_A_prev = 0;      // Help Variable for Rotary Encoder



//*********************************************setup*****************************************************
//*********************************************setup*****************************************************
//*********************************************setup*****************************************************
//*********************************************setup*****************************************************



void setup()
{
 // Setup & set pin modes INPUT or OUTPUT and write Initial conditions LOW or HIGH
  irrecv.enableIRIn();                  // Start the IR Receiver
  bigCrystal.begin(20, 4);              // Define LCD 20x4
  analogReference(     EXTERNAL);      // use AREF for reference voltage // Use 2,500 Volt reference to the Aref pin - see also the value of bias check - example:
bias_check_Ub
  pinMode(Ub_test_L,       INPUT);     // Set PinModes for Voltage readings of the MU-Stage
  pinMode(Ub_test_R,       INPUT);
  pinMode(Uout_test_L,     INPUT);
  pinMode(Uout_test_R,     INPUT);
  pinMode(Ugk_test_L,      INPUT);
  pinMode(Ugk_test_R,      INPUT);
  pinMode(tape_dddac,     OUTPUT);     // Set Pinmodes for the outputs driving the Tape relays
  pinMode(tape_sacd,      OUTPUT);
  pinMode(tape_phono,     OUTPUT);
  pinMode(tape_B77,       OUTPUT);
  pinMode(tape_PR99,      OUTPUT);
  pinMode(tape_aux,       OUTPUT);
  pinMode(source_dddac,   OUTPUT);     // Set Pinmodes for the outputs driving the Source relays
  pinMode(source_sacd,    OUTPUT);
  pinMode(source_phono,   OUTPUT);
  pinMode(source_B77,     OUTPUT);
  pinMode(source_PR99,    OUTPUT);
  pinMode(source_aux,     OUTPUT);
  pinMode(brightness_out, OUTPUT);     // Set up brightness output
  pinMode(contrast_out,   OUTPUT);     // Set up contrast output
  pinMode(volume_step_01, OUTPUT);     // Set up 24x volume Digital I/O as outputs to drive Relays
 pinMode(volume_step_02, OUTPUT);
 pinMode(volume_step_03, OUTPUT);
 pinMode(volume_step_04, OUTPUT);
 pinMode(volume_step_05, OUTPUT);
 pinMode(volume_step_06, OUTPUT);
 pinMode(volume_step_07, OUTPUT);
 pinMode(volume_step_08, OUTPUT);
 pinMode(volume_step_09, OUTPUT);
 pinMode(volume_step_10, OUTPUT);
 pinMode(volume_step_11, OUTPUT);
 pinMode(volume_step_12, OUTPUT);
 pinMode(volume_step_13, OUTPUT);
 pinMode(volume_step_14, OUTPUT);
 pinMode(volume_step_15, OUTPUT);
 pinMode(volume_step_16, OUTPUT);
 pinMode(volume_step_17, OUTPUT);
 pinMode(volume_step_18, OUTPUT);
 pinMode(volume_step_19, OUTPUT);
 pinMode(volume_step_20, OUTPUT);
 pinMode(volume_step_21, OUTPUT);
 pinMode(volume_step_22, OUTPUT);
 pinMode(volume_step_23, OUTPUT);
 pinMode(volume_step_24, OUTPUT);
```

```
digitalWrite(tape_dddac,      LOW);  // Start with no tape output
digitalWrite(tape_sacd,       LOW);
digitalWrite(tape_phono,      LOW);
digitalWrite(tape_B77,        LOW);
digitalWrite(tape_PR99,       LOW);
digitalWrite(tape_aux,        LOW);
digitalWrite(source_dddac,    LOW);  // Start with no  source selected
digitalWrite(source_sacd,     LOW);
digitalWrite(source_phono,    LOW);
digitalWrite(source_B77,      LOW);
digitalWrite(source_PR99,     LOW);
digitalWrite(source_aux,      LOW);
digitalWrite(volume_step_01, HIGH);  // Start with Volume STEP 1 (-99dB)
digitalWrite(volume_step_02, LOW);
digitalWrite(volume_step_03, LOW);
digitalWrite(volume_step_04, LOW);
digitalWrite(volume_step_05, LOW);
digitalWrite(volume_step_06, LOW);
digitalWrite(volume_step_07, LOW);
digitalWrite(volume_step_08, LOW);
digitalWrite(volume_step_09, LOW);
digitalWrite(volume_step_10, LOW);
digitalWrite(volume_step_11, LOW);
digitalWrite(volume_step_12, LOW);
digitalWrite(volume_step_13, LOW);
digitalWrite(volume_step_14, LOW);
digitalWrite(volume_step_15, LOW);
digitalWrite(volume_step_16, LOW);
digitalWrite(volume_step_17, LOW);
digitalWrite(volume_step_18, LOW);
digitalWrite(volume_step_19, LOW);
digitalWrite(volume_step_20, LOW);
digitalWrite(volume_step_21, LOW);
digitalWrite(volume_step_22, LOW);
digitalWrite(volume_step_23, LOW);
digitalWrite(volume_step_24, LOW);

pinMode(enc_B,              INPUT_PULLUP);     // Set Encoder pin B as Input
pinMode(enc_A,              INPUT_PULLUP);     // Set Encoder pin A as Input
pinMode(enc_push,           INPUT_PULLUP);     // Set Encoder Push Button as Input, active is LOW
pinMode(select_source_up,   INPUT_PULLUP);     // Set Encoder Push Button as Input, active is LOW
pinMode(select_source_down, INPUT_PULLUP);     // Set Encoder Push Button as Input, active is LOW
pinMode(select_tape_up,     INPUT_PULLUP);     // Set Encoder Push Button as Input, active is LOW
pinMode(select_tape_down,   INPUT_PULLUP);     // Set Encoder Push Button as Input, active is LOW
analogWrite(contrast_out, contrast  / 10);     // Start with initial value for LCD Contrast (52 = 520/10)

lcd.print("      CLEO 7      "); lcd.setCursor(0, 1);
lcd.print(" Tube Pre Amplifier "); lcd.setCursor(0, 2);
```

```
  lcd.print(" Arduino Controlled "); lcd.setCursor(0, 3);
  lcd.print(" 2020 DDDAC Design  ");
  for (int n = 27; n < 250; n++)                 // Start display from 20% (dark) to 95%
  {
    brightness = n;
   analogWrite(brightness_out, brightness);
    delay(10);
  }                                             // So basically start with initial value 250 = 95% for LCD Brightness
  delay(500);
  lcd.clear();
}                                                // End of void Setup


//**************************************************************************************************************
//***************************************************loop***********************************************************
//***************************************************loop***********************************************************
//***************************************************loop***********************************************************
//***************************************************loop***********************************************************
//**************************************************************************************************************



void loop()
{
  if (irrecv.decode(&results))                         // Check if remote is active and, than Read IR remote and depending on button clicked the IR_Value is set and
action taken
  {
    read_IR_Value_from_IR_Code();                      // This code is used in below routines to do stuff (volume, LCD, Selections etc etc)IR_Value has a value now
    if (menu_status == 0)                              // These actions are only allowed if the "Menu" is NOT active
    {
      IR_Volume_setting();                             // IR reading and setting the correct up down Volume level
      voltage_check();                                 // Run the voltage check
      mute_routine();                                  // IR Audio Mute Function
      BIG_small_toggle();                              // IR Toggles display format
      source_tape_select();                            // Drives 2 times 6 relays for signal inputs / outputs
    }
    menu_routine();                                    // IR Start the LCD menu routine or toggle back to normal mode
    irrecv.resume();                                   // Reset IR to be ready to receive the next value
  }
  if (menu_status == 0)                                // These actions are only allowed if the Menu is NOT active
  {
    read_encoder_for_volume();                         // Rotary 20-20 pulse encoder volume function
    front_panel_select();                              // Check fron pannel push-buttons (switches) and encoder push button
    volume_actions();                                  // Call Function to do VOLUME calculation and select relay step + set the LCD text for volume
    bias_check();                                      // Check Bias every 5 minutes and display if all good or not
    lcd_small_volume_and_info_printing();              // Fill in 4 rows display with all existing active info when NOT BIG and when NOT in settings menu and hard
mute
    print_big_volume();                                // BIG printing Volume on row 2
  }
```

```
}


//**************************************************functions************************************************************
//**************************************************functions************************************************************
//**************************************************functions************************************************************
//**************************************************functions************************************************************
//**************************************************functions************************************************************

















// Read IR remote and depending on button clicked assign a value to the variable "IR_Value"
voidread_IR_Value_from_IR_Code()
{
  if (results.value == OnOff)      IR_Value =  0 ;    // IR_value is being used for if functions in the code and take the corrects actions
  if (results.value == Menu)       IR_Value =  1 ;    // See introduction text at the top for declarations
  if (results.value == Home)       IR_Value =  2 ;
  if (results.value == Up)         IR_Value =  3 ;
  if (results.value == Down)       IR_Value =  4 ;
  if (results.value == Left)       IR_Value =  5 ;
  if (results.value == Right)      IR_Value =  6 ;
  if (results.value == OK)         IR_Value =  7 ;
  if (results.value == Return)     IR_Value =  8 ;
  if (results.value == VolumeDown) IR_Value =  9 ;
  if (results.value == VolumeUp)   IR_Value = 10 ;
  if (results.value == Mous)       IR_Value = 11 ;
}



//IRVolumefunction...........................................................................................................
```

```
void IR_Volume_setting()
{
  if (mute_hard_status == 0)                                          // Volume only works when hard mute is "0"
  {
    if (IR_Value == 10) volume_change =  1;                           // Volume 1 step UP
    if (IR_Value ==  9) volume_change = -1;                           // Volume 1 step DOWN
    if (results.value == Repeat && IR_Value == 10) volume_change =  1;   // Volume repeat function "UP"
    if (results.value == Repeat && IR_Value ==  9) volume_change = -1;   // Volume repeat function "DOWN"
  }
}



// Voltage Check - Subroutine function to check Power supply status of 3 power supplies and show on LCD-----------------
void voltage_check()
{
  if (IR_Value == 11)                                 // Run this when the mouse button is pressed
  {
    lcd.clear();
    lcd.setCursor(0, 1);
    lcd.print("     Please Wait    ");                // Print start message on LCD
    read_Ub_L   = 0;                                  // "Reset" all reading values
    read_Uout_L = 0;
    read_Ugk_L  = 0;
    read_Ub_R   = 0;
    read_Uout_R = 0;
    read_Ugk_R  = 0;
    for (int n = 0; n < 100; n++)
    {
      read_Ub_L   += analogRead(Ub_test_L);           // Read voltage 100 times and add together LEFT
      read_Uout_L += analogRead(Uout_test_L);
      read_Ugk_L  += analogRead(Ugk_test_L);
      read_Ub_R   += analogRead(Ub_test_R);           // Read voltage 100 times and add together RIGHT
      read_Uout_R += analogRead(Uout_test_R);
      read_Ugk_R  += analogRead(Ugk_test_R);
      delay(15);                                      // Total time is ~ 2 second
    }
    percent_Ub_L   = read_Ub_L   / bias_check_Ub_L;     // Calculate performence % - Average the readings (100) this will make automatically % out of it  ;-)
    percent_Uout_L = read_Uout_L / bias_check_Uout_L;   // calculate the percentage of where voltage is versus 100% bias point
    percent_Ugk_L  = read_Ugk_L  / bias_check_Ugk_L;    // calculate the percentage of where voltage is versus 100% bias point
    percent_Ub_R   = read_Ub_R   / bias_check_Ub_R;     // calculate the percentage of where voltage is versus 100% bias point
    percent_Uout_R = read_Uout_R / bias_check_Uout_R;   // calculate the percentage of where voltage is versus 100% bias point
    percent_Ugk_R  = read_Ugk_R  / bias_check_Ugk_R;    // calculate the percentage of where voltage is versus 100% bias point
    lcd.setCursor(0, 0);
    lcd.print("V-Check   L     R   ");                // Print start message on LCD
    lcd.setCursor(0, 1);
    lcd.print("Ub Powr             ");                // Result comes later at position 9 (L) and 15 (R)
    lcd.setCursor(0, 2);
    lcd.print("Uo Bias             ");
    lcd.setCursor(0, 3);
    lcd.print("Uk Bias             ");
    lcd.setCursor(9, 1);
    if (percent_Ub_L < 100)    lcd.print (" ");          // formatting the readout if it is below 100%
    lcd.print(percent_Ub_L);   lcd.print("%");
    lcd.setCursor(9, 2);
    if (percent_Uout_L < 100)  lcd.print (" ");
    lcd.print(percent_Uout_L); lcd.print("%");
    lcd.setCursor(9, 3);
    if (percent_Ugk_L < 100)   lcd.print (" ");
    lcd.print(percent_Ugk_L);  lcd.print("%");
    lcd.setCursor(15, 1);                                // Same for Right channel
    if (percent_Ub_R < 100)    lcd.print (" ");          // formatting the readout if it is below 100%
    lcd.print(percent_Ub_R);   lcd.print ("%");
    lcd.setCursor(15, 2);
    if (percent_Uout_R < 100)  lcd.print (" ");
    lcd.print(percent_Uout_R); lcd.print ("%");
    lcd.setCursor(15, 3);
    if (percent_Ugk_R < 100)   lcd.print (" ");
    lcd.print(percent_Ugk_R);  lcd.print ("%");
    delay(4000);                                         // Keep screen up for 4 seconds
    lcd.clear();                                         // Secondly  show real voltages
    lcd.setCursor(0, 0);
    lcd.print("Volt    L       R   ");                   // Print start message on LCD and calculate the reading back to real original volts
    lcd.setCursor(0, 1);
    lcd.print("Vb   ");    lcd.print(read_Ub_L * 1.18 / 100, 2)      ; lcd.print("  ")  ;  lcd.print(read_Ub_R * 1.17 / 100, 2);
    lcd.setCursor(0, 2);
    lcd.print("Uo   ");    lcd.print(read_Uout_L * 0.5381 / 100, 2)  ; lcd.print("  ")  ;  lcd.print(read_Uout_R * 0.5424 / 100, 2);
    lcd.setCursor(0, 3);
    lcd.print("Ugk    ");    lcd.print(read_Ugk_L * 0.01691 / 100, 2); lcd.print("    ");  lcd.print(read_Ugk_R * 0.01681 / 100, 2);
    delay(6500);                                         // Show voltages 6,5 seconds
    lcd.clear();
    IR_Value = 99;                                       // Reset IR value
  }
}



// HARD Mute function by (IR) Remote only..........................................................................
void mute_routine()
{
  if ((mute_hard_status) == 0 && ((IR_Value == 0) || (IR_Value == 7)))     // When <OnOff> button is active "Hard Mute" toggles
  {
    digitalWrite(volume_step_01, HIGH);
    mute_hard_status = 1;
    IR_Value = 99;                              // "RESET" IR_Value to wait for next round of reading the IR Signal
    if (big_status == 0)
    {
      lcd.setCursor(8, 0);
    lcd.print(" Mute      ");                // Mute text is printed
   }
   if (big_status == 1)                      // different display
   {
    BIG_Char = "mute ";                      // Print MUTE  in BIG
    big_row_position = 2;
    bigprint0();
   }
  }
 if ((mute_hard_status) == 1 && ((IR_Value == 0) || (IR_Value == 7)))    // When Enter button is active "Hard Mute" toggles back to normal situation
 {
  digitalWrite(volume_step_01, LOW);
  mute_hard_status = 0;
  IR_Value = 99;                             // "RESET" IR_Value to wait for next round of reading the IR Signal
 }
}



// <PLAY> Display function by (IR) Remote button.................................................................
void BIG_small_toggle()
{
 if ((big_status == 0) && ((IR_Value == 8) || (IR_Value == 2)))       // When <Enter> button is active display toggles between normal and BIG
 {
  big_status = 1;                            // Now display is using BIG Char
  IR_Value = 99;                             // "RESET" IR_Value to wait for next round of reading the IR Signal
  print_input_selection_big();               // Print the current existing selection when Big starts
  String_for_CHAR = lcd_vol_big;             // Transfer Volume string to big print String
  BIG_Char = String_for_CHAR.c_str();        // convert the string into Char
  big_row_position = 2;
  bigprint0();
  lcd.setCursor(15, 3);
  lcd.print("    ");                         // Clean last character in row 3 (from kHz...)
 }
 if ((big_status == 1) && ((IR_Value == 8) || (IR_Value == 2)))       // When Menu button is active and we are in big num, we go back to normal
 {
  big_status = 0;
  IR_Value = 99;                             // "RESET" IR_Value to wait for next round of reading the IR Signal
  lcd.clear();                               // one time full clear of the display
 }
}



// Source and tape Selection (key buttons on remote and front panel up down switches)
void source_tape_select()
{
 if (menu_status == 0)                              // Selection only works when menu is in-active "0"
 {
```

```
if (IR_Value == 5 || IR_Value == 6)               // Change the source up/down through the list so to speak the table
 {
  digitalWrite(source_dddac, LOW);                 // start with switching everything OFF before the new one comes "online"
  digitalWrite(source_sacd,  LOW);
  digitalWrite(source_phono, LOW);
  digitalWrite(source_B77,   LOW);
  digitalWrite(source_PR99,  LOW);
  digitalWrite(source_aux,   LOW);
  if (IR_Value == 6)                               // Change the source up, so DOWN through the list so to speak the table
  {
    source_status = source_status + 1;
    if (source_status > 6) source_status = 0;      // We are rotating the selections
     IR_Value = 99;                                // RESET IR
  }
  if (IR_Value == 5)                               // Change the source UP through the list so to speak the table
  {
    source_status = source_status - 1;
    if (source_status < 0) source_status = 6;      // We are rotating the selections
     IR_Value = 99;                                // RESET IR
  }
  switch (source_status)                           // Drive the approviate Relay for the source input
  {
    case 1:
     digitalWrite(source_dddac, HIGH);
      lcd_source = "DDDAC 1794 ";
      break;
    case 2:
     digitalWrite(source_sacd,  HIGH);
      lcd_source = "Sony SACD  ";
      break;
    case 3:
     digitalWrite(source_phono, HIGH);
      lcd_source = "PhonoDude  ";
      break;
    case 4:
     digitalWrite(source_B77,   HIGH);
      lcd_source = "Revox B77  ";
      break;
    case 5:
     digitalWrite(source_PR99,  HIGH);
      lcd_source = "Revox PR99 ";
      break;
    case 6:
     digitalWrite(source_aux,   HIGH);
      lcd_source = "Aux Input  ";
      break;
    case 0:
      lcd_source = "None       ";
      break;
    }
  if (big_status == 1) print_input_selection_big(); // If BIG nums are used, print the new selection on LCD
 }

 if (IR_Value == 3 || IR_Value == 4)              // Change the source up/down through the list so to speak the table
 {
  digitalWrite(tape_dddac, LOW);                  // start with switching everything OFF before the new one comes "online"
  digitalWrite(tape_sacd,  LOW);
  digitalWrite(tape_phono, LOW);
  digitalWrite(tape_B77,   LOW);
  digitalWrite(tape_PR99,  LOW);
  digitalWrite(tape_aux,   LOW);
  if (IR_Value == 3)                              // Change the tape DOWN through the list so to speak the table
  {
    tape_status = tape_status + 1;
    if (tape_status > 6) tape_status = 0;         // We are rotating the selections
    IR_Value = 99;                                // RESET IR
  }
  if (IR_Value == 4)                              // Change the tape DOWN through the list so to speak the table
  {
    tape_status = tape_status - 1;
    if (tape_status < 0) tape_status = 6;         // We are rotating the selections
    IR_Value = 99;                                // RESET IR
  }
  switch (tape_status)                            // Drive the approviate Relay for the tape output
  {
    case 1:
     digitalWrite(tape_dddac, HIGH);
     lcd_tape = "DDDAC 1794 ";
     break;
    case 2:
     digitalWrite(tape_sacd,  HIGH);
     lcd_tape = "Sony SACD  ";
     break;
    case 3:
     digitalWrite(tape_phono, HIGH);
     lcd_tape = "PhonoDude  ";
     break;
    case 4:
     digitalWrite(tape_B77,   HIGH);
     lcd_tape = "Revox B77  ";
     break;
    case 5:
     digitalWrite(tape_PR99,  HIGH);
     lcd_tape = "Revox PR99 ";
     break;
    case 6:
     digitalWrite(tape_aux,   HIGH);
     lcd_tape = "Aux Input  ";
```

```
          break;
        case 0:
          lcd_tape = "None        ";
          break;
      }
    }
  }
}



// Menu LCD Screen functions by (IR) Remote only..............................................................................
void menu_routine()
{
  if (IR_Value == 1 && menu_status == 0)      // When <Menu> button is active LCD Menu starts when in normal mode (menu_status = 0 ...)
  {
    menu_status = 1;                          // Flag status to 1 - to avoid other actions
    lcd.setCursor(0, 0);
    lcd.print ( "      LCD MENU      " );
    lcd.setCursor(0, 1);
    lcd.print ( " ------------------ " );
    lcd.setCursor(0, 2);
    lcd.print ( "Arrows  LCD settings" );
    lcd.setCursor(0, 3);
    lcd.print ( "   Home reset LCD   " );
    IR_Value = 99;                            // Reset IR value
  }
  if (IR_Value == 1 && menu_status == 1)                  // Toggle back to normal mode
  {
    menu_status = 0;
    IR_Value = 99;                                      // "RESET" IR_Value to wait for next round of reading the IR Signal
  }
  if (menu_status == 1)
  {
    // BRIGHTNESS control (IR)
    if ((results.value == Repeat && IR_Value == 3) || (IR_Value == 3))       // Check if Up button is pressed or repeat pressed
    {
      brightness += 23;                                 // During hard mute UP and DOWN set brightness
      if (brightness > 250) brightness = 250;           // Stop scrolling around by setting max and min levels
      if (brightness < 20) brightness = 20;
      lcd.setCursor(0, 0);                              // Display brigntness message
      lcd.print("Brightness   ");
      lcd.setCursor(0, 1);                              // Display brigntness message
      lcd.print("                    ");
      lcd.setCursor(0, 2);
      lcd.print("Use <Up> & <Down>   ");
      lcd.setCursor(0, 3);
      lcd.print("Use <Home> to reset ");
      lcd.setCursor(13, 0);
```

```
   int b_percent = map (brightness, 20, 250, 0, 100);  // Make brightness scale 0-100%
   lcd.print(b_percent);
   lcd.print(" %  ");
   analogWrite(brightness_out, brightness);            // Output LED voltage for LCD
  }
 if ((results.value == Repeat && IR_Value == 4) || (IR_Value == 4))      // Check if down button is pressed or repeat pressed
  {
   brightness -= 23;                                   // During hard mute UP and DOWN set brightness
   if (brightness > 250) brightness = 250;             // Stop scrolling around by setting max and min levels
   if (brightness < 20) brightness = 20;
   lcd.setCursor(0, 0);                                // Display brightness message
   lcd.print("Brightness   ");
   lcd.setCursor(0, 1);                                // Display brigntness message
   lcd.print("                    ");
   lcd.setCursor(0, 2);
   lcd.print("Use <Up> & <Down>   ");
   lcd.setCursor(0, 3);
   lcd.print("Use <Home> to reset ");
   lcd.setCursor(13, 0);
   int b_percent = map (brightness, 20, 250, 0, 100);  // Make brightness scale 0-100%
   lcd.print(b_percent);
   lcd.print(" %  ");
   analogWrite(brightness_out, brightness);            // Output LED voltage for LCD
  }
// CONTRAST control (IR) during HARD MUTE......................................................................
 if ((results.value == Repeat && IR_Value == 6) || (IR_Value == 6))      // Check if Right button is pressed or repeat pressed
  {
   contrast += 4;
   if (contrast > 80) contrast = 80;                   // Stop scrolling around by setting max and min levels
   if (contrast < 0) contrast = 0;
   lcd.setCursor(0, 0);                                // Display contrast message
   lcd.print("Contrast      %    ");
   int c_percent = map (contrast, 0, 80, 0, 100);      // Make contrast scale 0 - 100
   lcd.setCursor(12, 0);
   lcd.print(c_percent);
   lcd.setCursor(0, 1);
   lcd.print("                    ");
   lcd.setCursor(0, 2);
   lcd.print("Use <Left> & <Right>");
   lcd.setCursor(0, 3);
   lcd.print("Use <Home> to reset ");
   analogWrite(contrast_out, contrast / 10);           // Output Contrast Voltage
  }
 if ((results.value == Repeat && IR_Value == 5) || (IR_Value == 5))      // Check if Left button is pressed or repeat pressed
  {
   contrast -= 4 ;
   if (contrast > 80) contrast = 80;                   // Stop scrolling around by setting max and min levels
   if (contrast < 0) contrast = 00;
   lcd.setCursor(0, 0);                                // Display contrast message
```

```
      lcd.print("Contrast       %    ");
      int c_percent = map (contrast, 0, 80, 0, 100);     // Make contrast scale 0 - 100
      lcd.setCursor(12, 0);
      lcd.print(c_percent);
      lcd.setCursor(0, 1);
      lcd.print("                    ");
      lcd.setCursor(0, 2);
      lcd.print("Use <Left> & <Right>");
      lcd.setCursor(0, 3);
      lcd.print("Use <Home> to reset ");
      analogWrite(contrast_out, contrast / 10);          // Output Contrast Voltage
    }
    // RESET Brightness and Contrast with <Home> button
    if  (IR_Value == 2)                                  // Check if Menu button is pressed
    {
      contrast = 40;                                     // Back to "mid" value
      brightness = 125;                                  // Back to "mid" value
      analogWrite(contrast_out, contrast / 10);          // Output Contrast level
      analogWrite(brightness_out, brightness);           // Output brightness level
      lcd.setCursor(0, 0);                              // Display brigntness message (arbitrary choice... Contrast will do also)
      lcd.print("Brightness  50 %     ");
      lcd.setCursor(0, 1);
      lcd.print("Contrast    50 %     ");
      lcd.setCursor(0, 2);
      lcd.print("                    ");
      lcd.setCursor(0, 3);
      lcd.print("Value Reset done    ");
    }                                                    // Confirmation of reset on LCD
    IR_Value = 99;
  }
}



// Rotary 20-20 pulse encoder volume function ...........................................................................
void read_encoder_for_volume()                // Rotary 20-20 pulse encoder volume function
.......................................................................................
{
  for (int n = 0; n < 4000; n++)              // ENCODER VOLUME - loop n=x times to not miss any encoder pulses is basically defining the speed of fast volume
loop  by remote
  {
    encoder_A = digitalRead(enc_A);            // Read encoder pins
    encoder_B = digitalRead(enc_B);            // Read encoder pins
    if ((!encoder_A) && (encoder_A_prev))      // A has gone from high to low
    {
      if (encoder_B) volume_change = 1;        // B is high so clockwise, so volume up 1
      else volume_change = -1;                // B is low so counter-clockwise, so volume down -1
    }
    encoder_A_prev = encoder_A;                // Store value of A for next time
  }
}



// Front panel switches and encoder PUSH button.............................................................................
void front_panel_select()
{
  push_input = digitalRead (enc_push);                    // Read if encoder push button has gone active LOW
  if (push_input == 0 && mute_hard_status == 0)           // Set up to toggle status on/off
  {
    mute_hard_status = 1;                                 // Toggle status to "on" (1)
    digitalWrite(volume_step_01, HIGH);                   // Switch on relay -99dB
    delay(500);
    if (push_input == 0)                                 // Check if Pushbutton is stil pushed in (push_input == 0)- if yes, Act as if <PLAY> Button is pressed and
toggle BIG - small LCD Text
    {
      delay (500);                                       // wait another 500ms, so only after 1 full sec the toggle function will work...
      push_input = digitalRead (enc_push);                // Read if encoder push button is still LOW
      if (push_input == 0)
      {
        IR_Value = 8;                                     // Act as <Enter> button
        BIG_small_toggle();                               // Run soubroutine for BIG small LCD
        delay(500);
      }
    }
    push_input = 9;                                       // Kind of "Reset" the push indicator and wait for next one
  }                                                      // Jump out of code as otherwise status = 1 will be executed immediately
  if (push_input == 0 && mute_hard_status == 1)           // Set up to toggle status on/off
  {
    mute_hard_status = 0;                                 // Toggle status to "off" (0)
    digitalWrite(volume_step_01, LOW);                    // Switch off relay -99dB
    delay(500);
    if (push_input == 0)                                 // Check if Pushbutton is stil pushed in (push_input == 0)- if yes, Act as if <PLAY> Button is pressed and
toggle BIG - small LCD Text
    {
      delay (500);                                       // wait another 500ms, so only after 1 full sec the toggle function will work...
      push_input = digitalRead (enc_push);                // Read if encoder push button is still LOW
      if (push_input == 0)
      {
        IR_Value = 8;                                     // Act as <Enter> button
        BIG_small_toggle();                               // Run soubroutine for BIG small LCD
        delay(500);
      }
    }
    push_input = 9;                                       // Kind of "Reset" the push indicator and wait for next one
  }
  // Read front panel switches for source and tape select and take actions afterwards
  push_source_up   = digitalRead (select_source_up);      // Read toggle switches
  push_source_down = digitalRead (select_source_down);    // Read toggle switches
  push_tape_up     = digitalRead (select_tape_up);        // Read toggle switches
  push_tape_down   = digitalRead (select_tape_down);      // Read toggle switches
  if (push_source_up == 0)
  {
    delay(300);                                           // debounce delay
    IR_Value = 6;                                        // This is same as a Button pressed on China remote - 6= up / 5 = down
    source_tape_select();                                 // Jump to this code as if China was used
  }
  if (push_source_down == 0)
  {
    delay(300);                                           // debounce delay
    IR_Value = 5;                                        // This is same as a Button pressed on China remote - 6= up / 5 = down
    source_tape_select();                                 // Jump to this code as if China was used
  }
  if (push_tape_up == 0)
  {
    delay(300);                                           // debounce delay
    IR_Value = 3;                                        // This is same as a Button pressed on China remote - 3= up / 4 = down
    source_tape_select();                                 // Jump to this code as if China was used
  }
  if (push_tape_down == 0)
  {
    delay(300);                                           // debounce delay
    IR_Value = 4;                                        // This is same as a Button pressed on China remote - 3= up / 4 = down
    source_tape_select();                                 // Jump to this code as if China was used
  }
}



// Function to do VOLUME calculation and select volume step / output to corresponding volume relais---------------------------- cleo done
void volume_actions()
{
  if (volume_change != 0 && mute_hard_status == 0);  //  Only do all this code when there is a need (volume delta = 1/-1 AND && Hard Mute is OFF
  {
    volume_new = volume_old + volume_change;        // In- or decrease the volume level value (1-24)
    if (volume_new > 24) volume_new = 24;           // Max volume level
    if (volume_new < 1)  volume_new =  1;           // Min volume level

    if (volume_new == 1) {
     digitalWrite(volume_step_01, HIGH);           // Activate new volume relais and set the string variable for LCD volume in dB
     lcd_vol_big = " -99.0";
     lcd_vol = " - 99.0 dB  ";
    }
    if (volume_new == 2) {
     digitalWrite(volume_step_02, HIGH);
     lcd_vol_big = " -58.0";
     lcd_vol = " - 58.0 dB  ";
```

```
}
if (volume_new == 3) {
 digitalWrite(volume_step_03, HIGH);
  lcd_vol_big = " -54.0";
  lcd_vol = " - 54.0 dB  ";
}
if (volume_new == 4) {
 digitalWrite(volume_step_04, HIGH);
  lcd_vol_big = " -50.0";
  lcd_vol = " - 50.0 dB  ";
}
if (volume_new == 5) {
 digitalWrite(volume_step_05, HIGH);
  lcd_vol_big = " -46.0";
  lcd_vol = " - 46.0 dB  ";
}
if (volume_new == 6) {
 digitalWrite(volume_step_06, HIGH);
  lcd_vol_big = " -42.0";
  lcd_vol = " - 42.0 dB  ";
}
if (volume_new == 7) {
 digitalWrite(volume_step_07, HIGH);
  lcd_vol_big = " -38.0";
  lcd_vol = " - 38.0 dB  ";
}
if (volume_new == 8) {
 digitalWrite(volume_step_08, HIGH);
  lcd_vol_big = " -34.0";
  lcd_vol = " - 34.0 dB  ";
}
if (volume_new == 9) {
 digitalWrite(volume_step_09, HIGH);
  lcd_vol_big = " -30.0";
  lcd_vol = " - 30.0 dB  ";
}
if (volume_new == 10) {
 digitalWrite(volume_step_10, HIGH);
  lcd_vol_big = " -28.0";
  lcd_vol = " - 28.0 dB  ";
}
if (volume_new == 11) {
 digitalWrite(volume_step_11, HIGH);
  lcd_vol_big = " -26.0";
  lcd_vol = " - 26.0 dB  ";
}
if (volume_new == 12) {
 digitalWrite(volume_step_12, HIGH);
  lcd_vol_big = " -24.0";
```

```
    lcd_vol = " - 24.0 dB  ";
  }
  if (volume_new == 13) {
   digitalWrite(volume_step_13, HIGH);
    lcd_vol_big = " -22.0";
    lcd_vol = " - 22.0 dB  ";
  }
  if (volume_new == 14) {
   digitalWrite(volume_step_14, HIGH);
    lcd_vol_big = " -20.0";
    lcd_vol = " - 20.0 dB  ";
  }
  if (volume_new == 15) {
   digitalWrite(volume_step_15, HIGH);
    lcd_vol_big = " -18.0";
    lcd_vol = " - 18.0 dB  ";
  }
  if (volume_new == 16) {
   digitalWrite(volume_step_16, HIGH);
    lcd_vol_big = " -16.0";
    lcd_vol = " - 16.0 dB  ";
  }
  if (volume_new == 17) {
   digitalWrite(volume_step_17, HIGH);
    lcd_vol_big = " -14.0";
    lcd_vol = " - 14.0 dB  ";
  }
  if (volume_new == 18) {
   digitalWrite(volume_step_18, HIGH);
    lcd_vol_big = " -12.0";
    lcd_vol = " - 12.0 dB  ";
  }
  if (volume_new == 19) {
   digitalWrite(volume_step_19, HIGH);
    lcd_vol_big = " -10.0";
    lcd_vol = " - 10.0 dB  ";
  }
  if (volume_new == 20) {
   digitalWrite(volume_step_20, HIGH);
    lcd_vol_big = " -  8.0";
    lcd_vol = " -  8.0 dB  ";
  }
  if (volume_new == 21) {
   digitalWrite(volume_step_21, HIGH);
    lcd_vol_big = " -  6.0";
    lcd_vol = " -  6.0 dB  ";
  }
  if (volume_new == 22) {
   digitalWrite(volume_step_22, HIGH);
```

```
      lcd_vol_big = " -  4.0";
      lcd_vol = " -  4.0 dB  ";
    }
    if (volume_new == 23) {
     digitalWrite(volume_step_23, HIGH);
      lcd_vol_big = " -  2.0";
      lcd_vol = " -  2.0 dB  ";
    }
    if (volume_new == 24) {
     digitalWrite(volume_step_24, HIGH);
      lcd_vol_big = " -  0.0";
      lcd_vol = "    0.0 dB  ";
    }
    if (volume_old != volume_new) {                               // Only switch off the relays when old volume is not new volume
      delay(5);                                                   // Give new volume relais time to settle before switching the old one off
     if (volume_old == 1)  digitalWrite(volume_step_01, LOW);     // Now switch OFF the old volume relais AFTER switching ON the new volume relais to avoid clicks
in sound
     if (volume_old == 2)  digitalWrite(volume_step_02, LOW);
     if (volume_old == 3)  digitalWrite(volume_step_03, LOW);
     if (volume_old == 4)  digitalWrite(volume_step_04, LOW);
     if (volume_old == 5)  digitalWrite(volume_step_05, LOW);
     if (volume_old == 6)  digitalWrite(volume_step_06, LOW);
     if (volume_old == 7)  digitalWrite(volume_step_07, LOW);
     if (volume_old == 8)  digitalWrite(volume_step_08, LOW);
     if (volume_old == 9)  digitalWrite(volume_step_09, LOW);
     if (volume_old == 10) digitalWrite(volume_step_10, LOW);
     if (volume_old == 11) digitalWrite(volume_step_11, LOW);
     if (volume_old == 12) digitalWrite(volume_step_12, LOW);
     if (volume_old == 13) digitalWrite(volume_step_13, LOW);
     if (volume_old == 14) digitalWrite(volume_step_14, LOW);
     if (volume_old == 15) digitalWrite(volume_step_15, LOW);
     if (volume_old == 16) digitalWrite(volume_step_16, LOW);
     if (volume_old == 17) digitalWrite(volume_step_17, LOW);
     if (volume_old == 18) digitalWrite(volume_step_18, LOW);
     if (volume_old == 19) digitalWrite(volume_step_19, LOW);
     if (volume_old == 20) digitalWrite(volume_step_20, LOW);
     if (volume_old == 21) digitalWrite(volume_step_21, LOW);
     if (volume_old == 22) digitalWrite(volume_step_22, LOW);
     if (volume_old == 23) digitalWrite(volume_step_23, LOW);
     if (volume_old == 24) digitalWrite(volume_step_24, LOW);

    }
    volume_old = volume_new;       // Reset volume values
    volume_change = 0;             // Reset volume values
    delay(50);
  }
}
```

```
// Bias Check - Subroutine function to check Power supply status of 3 power supplies and see if one is too much OOF  bias for LCD warning
void bias_check()
{
 bias_counter++;
  if (bias_counter > 2400)                              // 5 minutes past --- 100 is 15 secs
  {
    bias_counter = 0;                                   // reset counter
    read_Ub_L    = 0;                                   // "Reset" all reading values
    read_Uout_L  = 0;
    read_Ugk_L   = 0;
    read_Ub_R    = 0;
    read_Uout_R  = 0;
    read_Ugk_R   = 0;
    for (int n = 0; n < 100; n++)
    {
      read_Ub_L   += analogRead(Ub_test_L);             // Read voltage 100 times and add together LEFT
     read_Uout_L += analogRead(Uout_test_L);
     read_Ugk_L  += analogRead(Ugk_test_L);
      read_Ub_R   += analogRead(Ub_test_R);             // Read voltage 100 times and add together RIGHT
     read_Uout_R += analogRead(Uout_test_R);
     read_Ugk_R  += analogRead(Ugk_test_R);
      delay(5);                                         // Total time is ~ 500m second
    }
   percent_Ub_L   = read_Ub_L   / bias_check_Ub_L;      // Calculate performence % - Average the readings (100) this will make automatically % out of it  ;-)
   percent_Uout_L = read_Uout_L / bias_check_Uout_L;    // calculate the percentage of where voltage is versus 100% bias point
   percent_Ugk_L  = read_Ugk_L  / bias_check_Ugk_L;     // calculate the percentage of where voltage is versus 100% bias point
   percent_Ub_R   = read_Ub_R   / bias_check_Ub_R;      // calculate the percentage of where voltage is versus 100% bias point
   percent_Uout_R = read_Uout_R / bias_check_Uout_R;    // calculate the percentage of where voltage is versus 100% bias point
   percent_Ugk_R  = read_Ugk_R  / bias_check_Ugk_R;     // calculate the percentage of where voltage is versus 100% bias point

    bias_is_off = 0;                                                      // start with zero : "no problem"
    if (percent_Ub_L >   106 || percent_Ub_L   < 94) bias_is_off = 1;      // do the tests and warn if one of them is outof range
    if (percent_Ub_R >   106 || percent_Ub_L   < 94) bias_is_off = 1;      // do the tests and warn if one of them is outof range
    if (percent_Uout_L > 107 || percent_Uout_L < 87) bias_is_off = 1;      // do the tests and warn if one of them is outof range
    if (percent_Uout_R > 107 || percent_Uout_R < 87) bias_is_off = 1;      // do the tests and warn if one of them is outof range
    if (percent_Ugk_L >  110 || percent_Ugk_L  < 90) bias_is_off = 1;      // do the tests and warn if one of them is outof range
    if (percent_Ugk_R >  110 || percent_Ugk_R  < 90) bias_is_off = 1;      // do the tests and warn if one of them is outof range
    if (bias_is_off == 0) lcd_bias = "All good   ";                        // set the lcd text for bias OK
    if (bias_is_off == 1) lcd_bias = "Check pls  ";                        // set the lcd text for bias OK
  }
}



//LCDSmallprinting.........................................................................................
void lcd_small_volume_and_info_printing()
{
  if (mute_hard_status == 0 && big_status == 0)        // Only print lcd Volume printing, when hard MUTE = 0 - (no brightness routines runs),  and when BIG Num is
```

```
in-active (0)
  {
    lcd.setCursor(0, 0);
    lcd.print("Volume : ");                          // Start with Index printing - all vriable printing starts at position 9 (8 for volume)
    lcd.setCursor(0, 1);
    lcd.print("Source : ");
    lcd.setCursor(0, 2);
    lcd.print("Tape   : ");
    lcd.setCursor(0, 3);
    lcd.print("Bias   : ");
    lcd.setCursor(8, 0);
    lcd.print (lcd_vol);                             // Print the actual dB Volume number
    lcd.setCursor(9, 1);
    lcd.print (lcd_source);                          // Print the actual source IN
    lcd.setCursor(9, 2);
    lcd.print (lcd_tape);                            // Print the actual TAPE OUT source
    lcd.setCursor(9, 3);
    lcd.print (lcd_bias);                            // Print Bias message
  }
}



// go print the BIG Numbers on LCD when status = 1
void print_big_volume()
{
  if (big_status == 1 && mute_hard_status == 0)     // go print the BIG Numbers on LCD when status = 1
  {
    String_for_CHAR = lcd_vol_big;                  // Transfer Volume string to big print String
    BIG_Char = String_for_CHAR.c_str();             // convert the string into Char
    big_row_position = 2;
    bigprint0();
  }
}



// Print big what is inside the Character Array called: BIG_Char starting at Row 0---------------------------------------
void bigprint0()
{
  bigCrystal.printBig(BIG_Char, 0, big_row_position );
}



// Print Input selection in the BIG section
voidprint_input_selection_big()
{
  switch (source_status)            // Correct Text for the source input
  {
    case 0:
      BIG_Char = "NONE ";             // Print Source in BIG
      break;
    case 1:
      BIG_Char = "DDDAC";
      break;
    case 2:
      BIG_Char = "SACD  ";
      break;
    case 3:
      BIG_Char = "PHONO";
      break;
    case 4:
      BIG_Char = "B77    ";
      break;
    case 5:
      BIG_Char = "PR99   ";
      break;
    case 6:
      BIG_Char = "AUX   ";
      break;
  }
  big_row_position = 0;
  bigprint0();
}



// All Code ends here
```