DAC-BD34301 説明書（Rev.1.0）

概要

DAC-BD34301はローム独自の音質設計を導入した32-bitステレオオーディオDAC BD34301EKVを搭載したDAC基板です．PCM，DSDそれぞれ768kHz，22.4MHzまでのサンプリングレートに対応しています．
BD34301EKVのレジスタを設定する為のマイコン等は載せていません．外部からMCU等でI2Cバスを通してレジスタを設定してください．

特徴

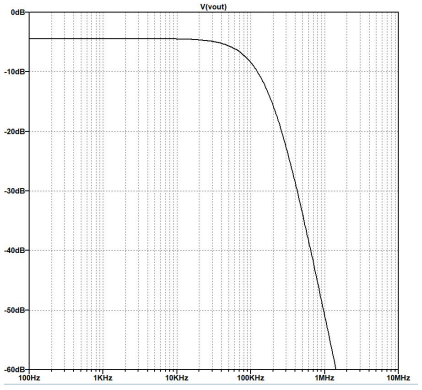
- 最大768kHzのPCM入力と22.5MHzのDSD入力に対応
- デジタルFIRフィルタはSharp Roll-Off，Slow Roll-Offの２種類を切り替え可能
- ステレオモード（2 ch），モノモード（1 ch）対応
- マイコンと通信を行うI2Cバスのスレーブアドレスは4通り（1Ch，1Dh，1Eh，1Fh）の設定が可能
- BD34301EKVの電源にはOS-CON（合計17個）を採用
- I-V変換用オペアンプとLPF用オペアンプは超低ノイズ，超低歪，高帯域のOPA1612， OPA1611を採用
- LPFにはPanasonicのフィルムコンデンサECHUシリーズを採用
- オペアンプはソケット対応
- 2個のGrove 4pin コネクタ
- 基板外形寸法：90mm x 100 x 20mm

仕様

項目		仕様	備考
電源	+5V	+4.75～5.25V 200mA	
	VCC	+11.40～15.75V 200mA	
	VEE	-11.40～15.75V 100mA	
出力電圧	PCM	2.1Vrms	0dB@1kHz
	DSD	0.8Vrms	-3dB@1kHz
サンプリング周波数	PCM	Up to 32 bit 768kHz	
	DSD	DSD 64/128/256/512	
基板外形寸法		90mm x 100mm x 20mm	

アナログLPF回路

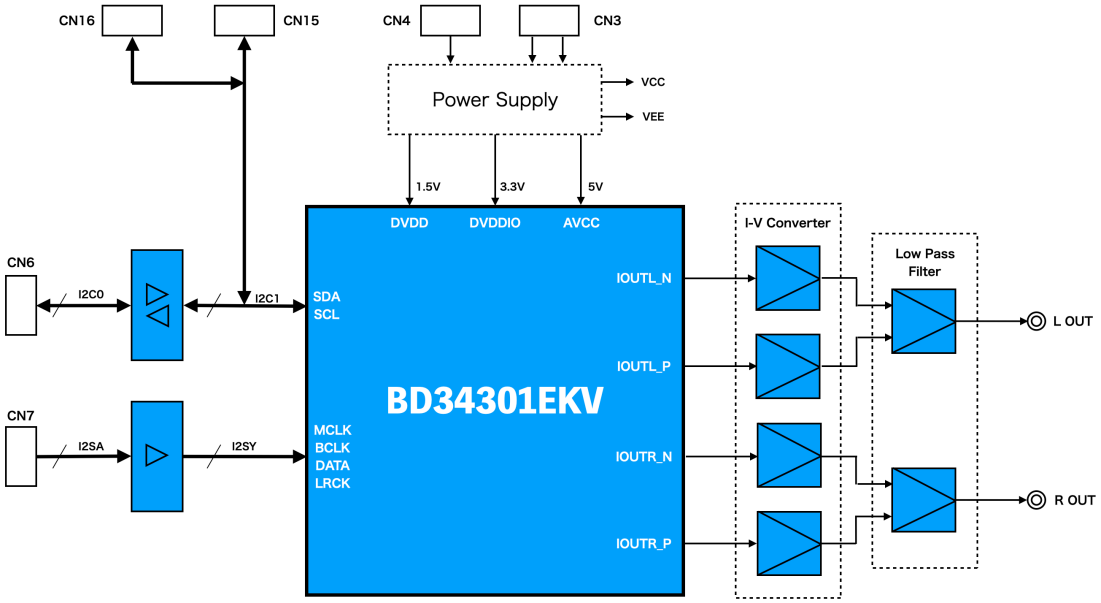
アナログLPFのオペアンプは高帯域オペアンプOPA1611で構成し，カットオフ周波数は85kHzで設計されています．



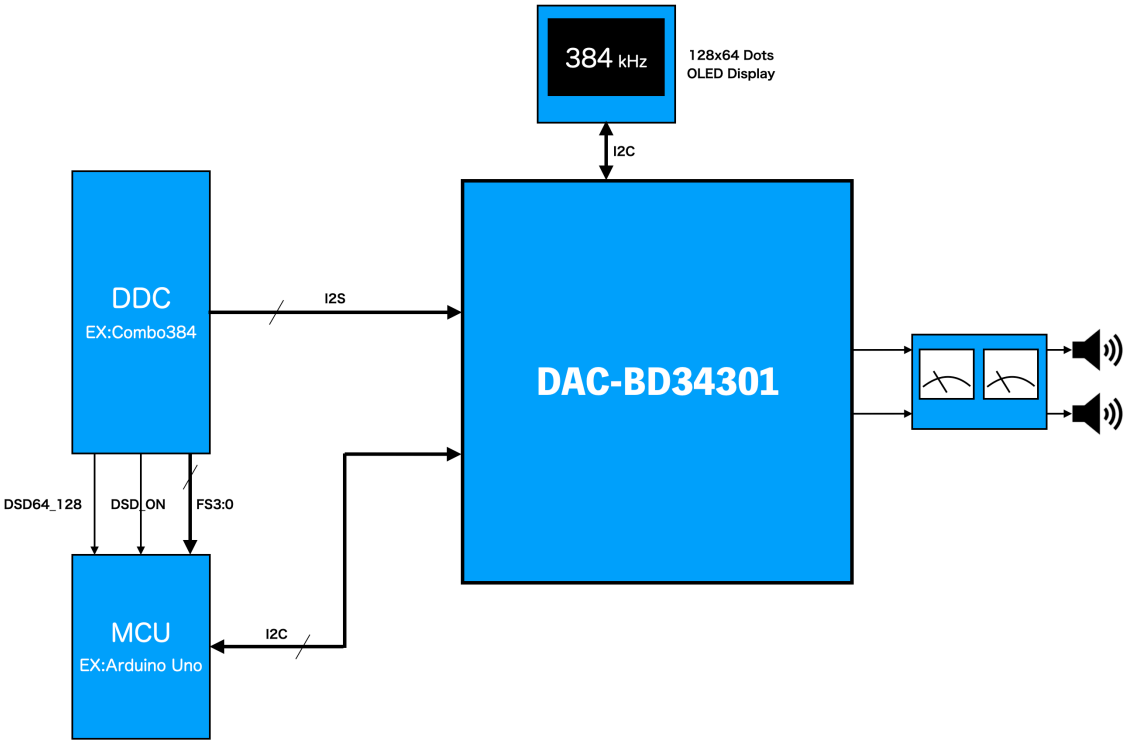
デジタル入力回路

MCU（Microcontroller Unit）やDDC（Digital to Dgital Converter）に接続されるI2CバスとI2S信号はそれぞれI2Cバスリピーターとバスバッファで受け，信号品質を高め動作の安定を図っています.

ブロック図

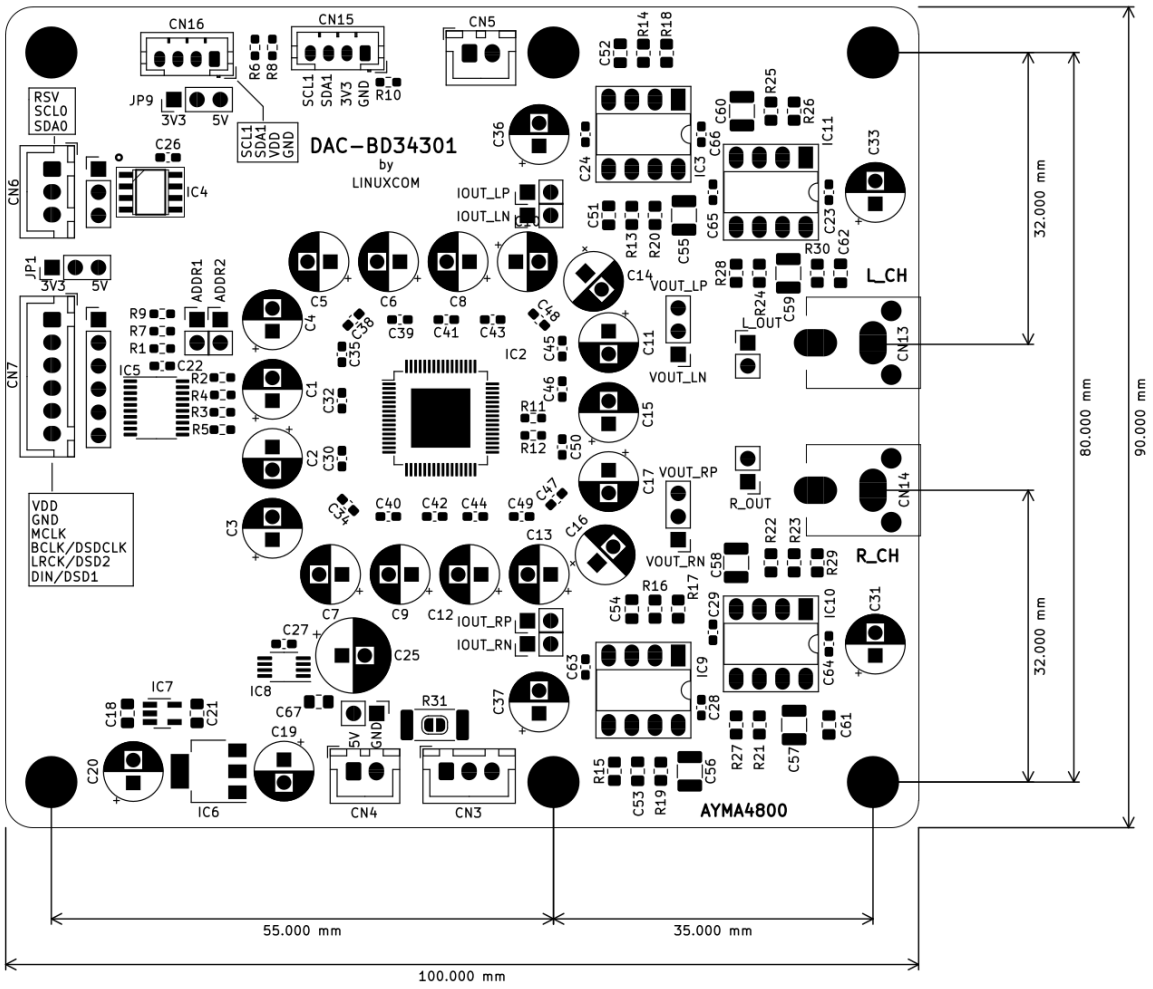


応用システム例



※I2Cバスのプルアップ抵抗はMCU側で行ってください

部品配置および基板外形図



ジャンパーピン設定

基板上のジャンパーピンの機能について説明します.

JPP1 アイソレーターIC電源電圧の設定

I2SインターフェースコネクタCN2の1番ピンはI2Sアイソレータ基板等で使用されるアイソレータICの電源になります. この電圧を選択します. アイソレータICを使用しない場合はオープンとしてください. デフォルトはオープンです.

1-2 : 3.3V

2-3 : 5V

JP2, JP3 スレーブアドレス設定

BD34301EKVのI2Cのスレーブアドレスを設定します. 以下の4通りが設定可能です.

JP3	JP2	スレーブアドレス
ショート	ショート	1Ch（デフォルト）
ショート	オープン	1Dh
オープン	ショート	1Eh
オープン	オープン	1Fh

JP4～JP7 DACの電流出力取り出しピン

BD34301EKVの電流出力をIV変換用のオペアンプに接続する為のショートピンです. 基板上のオペアンプを使用せずに外部で別途IV変換回路を組む場合などにオープンとすることで電流出力をとり出しやすくなります. デフォルトはショートです.

JP9 Grove-4Pコネクタ電源電圧の設定

Grove規格のI2Cモジュール接続用のコネクタの電源電圧を選択します.

1-2 : 3.3V

2-3 : 5V

尚, 5Vとした場合でもSDA, SCLは3.3V仕様となるようにしなければなりません. デフォルトは3.3Vです.

コネクタ機能説明

■ CN3 (B3B-XH-A(LF)(SN))

アナログ電源入力コネクタ.

Pin No.	I/O	信号名	説明
1	I	VCC	アナログ正電源（+12V～+15V）
2	-	GND	グラウンド
3	I	VEE	アナログ負電源（-12V～-15V）

■ CN4 (B2B-XH-A(LF)(SN))

デジタル電源入力コネクタ.

Pin No.	I/O	信号名	説明
1	I	+5V	+5V電源
2	-	GND	グラウンド

■CN5 (B2B-XH-A(LF)(SN))
デジタル電源3V3出力コネクタ.

Pin No.	I/O	信号名	説明
1	O	+3V3	+3.3V出力 外部3.3Vマイコン基板用
2	-	GND	グラウンド

■ CN6 (B3B-XH-A(LF)(SN))
I2Cインターフェースコネクタ.

Pin No.	I/O	信号名	説明
1	I	RST	BD34301EKVのリセット信号
2	IO	SCL0	I2Cクロック信号 LOBDAS等マイコン側（I2Cバス0）に接続される マイコン側でプルアップ抵抗が必要
3	IO	SDA0	I2Cデータ信号 LOBDAS等マイコン側（I2Cバス0）に接続される マイコン側でプルアップ抵抗が必要

■ CN7 (B6B-XH-A(LF)(SN))
I2Sインターフェースコネクタ.

Pin No.	I/O	信号名	説明
1	O	VDD	アイソレータIC電源 アイソレータICを使わない場合は未接続
2	-	GND	グラウンド
3	I	MCLK	マスタークロック LVCMOS
4	I	BCLK	オーディオシリアルデータクロック LVCMOS
5	I	DIN	オーディオシリアルデータ LVCMOS
6	I	LRCK	L/Rクロック LVCMOS

■ CN8（PH-1X3SG）
XLRバランス出力ローパスフィルター基板との接続用のLチャネルコネクタ.

Pin No.	I/O	信号名	説明
1	O	VOUT_LN	BD34301EKV LチャネルのIV変換後の電圧出力 差動の逆相出力
2	-	GND	グラウンド
3	O	VOUT_LP	BD34301EKV LチャネルIV変換後の電圧出力 差動の正相出力

■CN9（PH-1X3SG）
XLRバランス出力ローパスフィルター基板との接続用のRチャネルコネクタ.

Pin No.	I/O	信号名	説明
1	O	VOUT_RN	BD34301EKV RチャネルのIV変換後の電圧出力 差動の逆相出力
2	-	GND	グラウンド
3	O	VOUT_RP	BD34301EKV RチャネルIV変換後の電圧出力 差動の正相出力

■ CN11（PH-1X2SG）
Lチャネルアナログ出力コネクタ.

Pin No.	I/O	信号名	説明
1	O	L_OUT	Lチャネル出力
2	O	GND	グラウンド

■ CN12（PH-1X2SG）
Rチャネルアナログ出力コネクタ.

Pin No.	I/O	信号名	説明
1	O	R_OUT	Rチャネル出力
2	O	GND	グラウンド

■ CN13（HLR-3201VXB）
LチャネルRCA出力.

Pin No.	I/O	信号名	説明
1	O	GND	グラウンド
2	O	L_OUT	Lチャネル出力

■ CN14（HLR-3201VXB）
RチャネルRCA出力.

Pin No.	I/O	信号名	説明
1	O	GND	グラウンド
2	O	R_OUT	Rチャネル出力

■ CN15（Grove規格4pinコネクタ）
I2Cモジュール接続コネクタ.

Pin No.	I/O	信号名	説明
1	-	GND	グラウンド
2	O	3V3	+3.3V
3	IO	SDA1	I2Cデータ信号 BD34301EKV側（I2Cバス1）に接続される
4	IO	SCL1	I2Cクロック信号 BD34301EKV側（I2Cバス1）に接続される

■ CN16（Grove規格4pinコネクタ）
I2Cモジュール接続コネクタ.

Pin No.	I/O	信号名	説明
1	-	GND	グラウンド
2	O	3V3	+3.3V
3	IO	SDA1	I2Cデータ信号 BD34301EKV側（I2Cバス1）に接続される
4	IO	SCL1	I2Cクロック信号 BD34301EKV側（I2Cバス1）に接続される

5V トレラント信号

以下の信号は5Vトレラントです.

- RST
- SCL0

- SDA0
- MCLK
- BCLK
- DATA
- LRCK

プログラミングにおける注意点

パターンレイアウトの都合上、AudioOutpuPoralityレジスタ（14h）は0x01（Lch位相反転あり、Rch位相反転なし）としてください。

サンプルソースコード

以下はAmanero Combo384（又はその互換）基板とArduino Unoとのシステム構成で動作確認したサンプルソースコードです．Arduino UnoのIO(7:4)にCombo384のFS(3:0)を，IO3にDSD64_128を，IO2にDSD_ONを割り当てています．
表示はOLED 128x64ドットを使用し，サンプルレートのみ表示させています（U8g2ライブラリをインストールしてください）．またVolumeは0dB固定，デジタルフィルタはシャープロールオフ固定としています．そしてDSD256，DSD512の再生も可能ですが表示は2.8MHzとなります．
プログラミングの際はBD34301EKVのデータシートを参照ください．

```
/*
  Copyright (c) 2021/04/11 あおやまわたる

  以下に定める条件に従い，本ソフトウェアおよび関連文書のファイル（以下「ソフトウェア」）の複製を取得するすべての人に対し，ソフトウェアを無制限に扱うことを無償で許可します．これには，ソフトウェアの複製を使用，複写，変更，結合，掲載，頒布，サブライセンス，および/または販売する権利，およびソフトウェアを提供する相手に同じことを許可する権利も無制限に含まれます．

  上記の著作権表示および本許諾表示を，ソフトウェアのすべての複製または重要な部分に記載するものとします．

  ソフトウェアは「現状のまま」で，明示であるか暗黙であるかを問わず，何らの保証もなく提供されます．ここでいう保証とは，商品性，特定の目的への適合性，および権利非侵害についての保証も含みますが，それに限定されるものではありません．作者または著作権者は，契約行為，不法行為，またはそれ以外であろうと，ソフトウェアに起因または関連し，あるいはソフトウェアの使用またはその他の扱いによって生じる一切の請求，損害，その他の義務について何らの責任を負わないものとします．
*/
#include <Wire.h>
#include <U8g2lib.h>

#define BD34301_ADR 0x1C
#define SoftwareReset 0x00
#define ChipVersion 0x01
#define DigitalPower 0x02
#define AnalogPower 0x03
#define Clock1 0x04
#define Clock2 0x06
#define AudioIF1 0x10
#define AudioIF2 0x12
#define AudioIF3 0x13
#define AudioOutputPolarity 0x14
#define DSDFilter 0x16
#define AudioInputPolarity 0x17
#define VolumeTransitionTime 0x20
#define Volume1 0x21
#define Volume2 0x22
#define MuteTransitionTime 0x29
#define Mute 0x2A
#define RAMClear 0x2F
#define FIRFilter1 0x30
#define FIRFilter2 0x31
#define DeEmphasis1 0x33
#define DeEmphasis2 0x34
#define DeltaSigma 0x40
#define Setting1 0x41
#define Setting2 0x42
#define Setting3 0x43
#define Setting4 0x48
#define Setting5 0x60
#define Setting6 0x61
#define Boot1 0xD0
#define Boot2 0xD3

U8G2_SSD1306_128X64_NONAME_F_HW_I2C u8g2(U8G2_R0, /*SCL, SDA,*/ /* reset=*/ U8X8_PIN_NONE);

char pcm[] = "PCM ";
char dsd[] = "DSD ";
char fs32[] = "32";
char fs44[] = "44.1";
char fs48[] = "48";
char fs88[] = "88.2";
char fs96[] = "96";
char fs176[] = "176.4";
char fs192[] = "192";
char fs352[] = "352.8";
char fs384[] = "384";
char fs282[] = "2.8";
char fs564[] = "5.6";
char fs1128[] = "11.2";
char fs2256[] = "22.4";
char nofs[] = "";
uint8_t FS[] = {7, 6, 5, 4};
uint8_t DSD_ON = 2;
uint8_t DSD64_128 = 3;
uint8_t RST = 8;
uint8_t LED = 13;
uint8_t DAC_ON = 9;
uint8_t prevMode = 1;
uint16_t prevPcmRate = 0;
uint16_t prevDsdRate = 0;
boolean go = LOW;

void setup() {
  int i;
  for (i=0; i<=3; i++) pinMode(FS[i], INPUT);
  pinMode(DSD64_128, INPUT);
  pinMode(DSD_ON, INPUT);
  pinMode(DAC_ON, INPUT);
  pinMode(RST, OUTPUT);
  pinMode(LED, OUTPUT);

  Serial.begin(115200);
  Wire.begin();
  Wire.setClock(400000);
  while(go == LOW) {
    go = digitalRead(DAC_ON);
  }
  u8g2.begin();
  u8g2.clearBuffer();
  delay(500);
  digitalWrite(RST, HIGH);
  bootUp();
  initOledDisplay();
  delay(4000);
}

void loop() {
  uint16_t FSR = detectFS();
  uint8_t DIGIFIL = 1;
  modeSwitch(FSR, DIGIFIL);
  displayOledFSR(FSR);
  delay(10);
}

uint8_t bd34301ReadRegister(uint8_t regadr) {
  Wire.beginTransmission(BD34301_ADR);
  Wire.write(regadr);
  Wire.endTransmission();
  Wire.requestFrom(BD34301_ADR, 1);
  return Wire.read();
}

void bd34301WriteRegister(uint8_t regadr, uint8_t wdata) {
  Wire.beginTransmission(BD34301_ADR);
  Wire.write(regadr);
  Wire.write(wdata);
  Wire.endTransmission();
}

uint16_t detectFS() {
  uint8_t pcmRate = 0x00;
  uint16_t FSR;
  boolean dsdRate, dsdOn;
  int i;
  for (i=0; i<=3; i++) {
    pcmRate = pcmRate << 1 | digitalRead(FS[i]);
  }
  dsdRate = digitalRead(DSD64_128);
  dsdOn = digitalRead(DSD_ON);
  if (dsdOn == LOW) {
    digitalWrite(LED, LOW);
    if (pcmRate == 0x00 ) FSR = 32;
    else if (pcmRate == 0x01) FSR = 44;
    else if (pcmRate == 0x02) FSR = 48;
    else if (pcmRate == 0x03) FSR = 88;
    else if (pcmRate == 0x04) FSR = 96;
    else if (pcmRate == 0x05) FSR = 176;
    else if (pcmRate == 0x06) FSR = 192;
    else if (pcmRate == 0x07) FSR = 352;
    else if (pcmRate == 0x08) FSR = 384;
    else FSR = 0;
  }
  else {
    digitalWrite(LED, HIGH);
    if (dsdRate == LOW) FSR = 2822;
    else FSR = 5644;
  }
  return(FSR);
}

void initOledDisplay() {
  u8g2.clearBuffer();
  u8g2.setFont(u8g2_font_helvB12_tr);
  u8g2.drawStr(10,14,"DAC-BD34301");
  u8g2.setFont(u8g2_font_helvR10_tr);
  u8g2.drawStr(20,36,"Designed by");
  u8g2.setFont(u8g2_font_helvB12_tr);
  u8g2.drawStr(20,60,"LINUXCOM");
  u8g2.sendBuffer();
}

void displayOledFSR(uint16_t FSR) {
```

```
  u8g2.clearBuffer();
  u8g2.setFont(u8g2_font_helvR24_tr);
  u8g2.setCursor(22, 48);
  if (FSR == 32) u8g2.print(fs32);
  else if (FSR == 44) u8g2.print(fs44);
  else if (FSR == 48) u8g2.print(fs48);
  else if (FSR == 88) u8g2.print(fs88);
  else if (FSR == 96) u8g2.print(fs96);
  else if (FSR == 176) u8g2.print(fs176);
  else if (FSR == 192) u8g2.print(fs192);
  else if (FSR == 352) u8g2.print(fs352);
  else if (FSR == 384) u8g2.print(fs384);
  else if (FSR == 2822) u8g2.print(fs282);
  else if (FSR == 5644) u8g2.print(fs564);
  else if (FSR == 1128) u8g2.print(fs1128);
  else if (FSR == 2256) u8g2.print(fs2256);
  else u8g2.print(nofs);
  u8g2.setFont(u8g2_font_helvR10_tr);
  if (FSR <=384) u8g2.print(" kHz");
  else if (FSR >=2822) u8g2.print(" MHz");
  u8g2.sendBuffer();
}

void bootUp() {
  initBD34301();

  delayMicroseconds(10);

  // ソフトリセット　オフ
  bd34301WriteRegister(SoftwareReset, 0x01);
  // デジタルパワー　オン
  bd34301WriteRegister(DigitalPower, 0x01);
  // ポップノイズ防止
  bd34301WriteRegister(Boot1, 0x6A);
  bd34301WriteRegister(Boot2, 0x10);
  bd34301WriteRegister(Boot2, 0x00);
  bd34301WriteRegister(Boot1, 0x00);

  delayMicroseconds(200);

  // アナログパワー　オン
  bd34301WriteRegister(AnalogPower, 0x01);
  // RAMクリア　オン
  bd34301WriteRegister(RAMClear, 0x80);
  // RAMクリア　オフ
  bd34301WriteRegister(RAMClear, 0x00);
  // Mute オフ
  bd34301WriteRegister(Mute, 0x03);
}

void initBD34301() {
  /**********************
   * BD34301EKV 初期設定
   **********************/
  // MCLK分週比 : 1/2
  // 出力位相調整 : なし
  // PCMモード，DSDミュート機能有効，I2S，32-bit
  // PCMステレオモード，DSDステレオモード
  // Lch，Rchスワップなし
  // Lch位相反転あり，Rch位相反転なし
  // -- DSDフィルター --
  // DSD2.8M:26kHz DSD5.6MHz:52kHz
  // DSD11.2MHz:104kHz DSD22.4MHz:208kHz
  // ------------------
  // 入力位相反転なし
  // ボリューム遷移時間:1024fs
  // Attenuationレベル:0dB(L,R)
  // ミュート遷移時間:1024fs
  // シャープロールオフ,高精度演算オフ
  // ディエンファシス オフ
  // オーバーサンプリングレート:8倍
  bd34301WriteRegister(Clock1, 0x02);
  bd34301WriteRegister(Clock2, 0x00);
  bd34301WriteRegister(AudioIF1, 0x0B);
  bd34301WriteRegister(AudioIF2, 0x00);
  bd34301WriteRegister(AudioIF3, 0x00);
  bd34301WriteRegister(AudioOutputPolarity, 0x01);
  bd34301WriteRegister(DSDFilter, 0x01);
  bd34301WriteRegister(AudioInputPolarity, 0x00);
  bd34301WriteRegister(VolumeTransitionTime, 0x4B);
  bd34301WriteRegister(Volume1, 0x00);
  bd34301WriteRegister(Volume2, 0x00);
  bd34301WriteRegister(MuteTransitionTime, 0x48);
  bd34301WriteRegister(FIRFilter1, 0x01);
  bd34301WriteRegister(FIRFilter2, 0x80);
  bd34301WriteRegister(DeEmphasis1, 0x00);
  bd34301WriteRegister(DeEmphasis2, 0x00);
  bd34301WriteRegister(DeltaSigma, 0x00);
  bd34301WriteRegister(Setting1, 0x00);
  bd34301WriteRegister(Setting2, 0x34);
  bd34301WriteRegister(Setting3, 0xB8);
  bd34301WriteRegister(Setting4, 0x0D);
  bd34301WriteRegister(Setting5, 0x16);
  bd34301WriteRegister(Setting6, 0x16);
}

void modeSwitch(uint16_t FS, uint8_t digiFil) {
  uint16_t DSD = digitalRead(DSD_ON);
  if ((prevMode == 0) && (DSD == 0)) {
    if (prevPcmRate != FS) {
      sequenceOne();
      sequenceTwo(FS, digiFil);
      sequenceFour();
    }
    prevPcmRate = FS;
  } else if ((prevMode == 1) && (DSD == 1)) {
    if (prevDsdRate != FS) {
      sequenceOne();
      sequenceThree(FS);
      sequenceFive();
    }
    prevDsdRate = FS;
  } else if ((prevMode == 0) && (DSD == 1)) {
    sequenceOne();
    sequenceThree(FS);
    sequenceFive();
    prevMode = DSD;
  } else if ((prevMode == 1) && (DSD == 0)) {
    sequenceOne();
    sequenceTwo(FS, digiFil);
    sequenceFour();
    prevMode = DSD;
  }
}

void sequenceOne() {
  bd34301WriteRegister(Mute, 0x00); // ミュート　オン
  bd34301WriteRegister(DigitalPower, 0x00);  // デジタル・パワー　オフ
  bd34301WriteRegister(SoftwareReset, 0x00); // ソフト・リセット　オン
}

// PCM モード時の設定
void sequenceTwo(uint16_t FS, uint8_t digiFil) {
  if (FS == 32) {
    bd34301WriteRegister(Clock1, 0x03);
    bd34301WriteRegister(FIRFilter1, 0x01);
    bd34301WriteRegister(DeltaSigma, 0x02);
    if (digiFil == 1) bd34301WriteRegister(FIRFilter2, 0x80);
    else bd34301WriteRegister(FIRFilter2, 0x83);
  } else if ((FS == 44) || (FS == 48)) {
    bd34301WriteRegister(Clock1, 0x02);
    bd34301WriteRegister(FIRFilter1, 0x01);
    bd34301WriteRegister(DeltaSigma, 0x02);
    if (digiFil == 1) bd34301WriteRegister(FIRFilter2, 0x80);
    else bd34301WriteRegister(FIRFilter2, 0x83);
  } else if ((FS == 88) || (FS == 96)) {
    bd34301WriteRegister(Clock1, 0x00);
    bd34301WriteRegister(FIRFilter1, 0x02);
    bd34301WriteRegister(DeltaSigma, 0x11);
    if (digiFil == 1) bd34301WriteRegister(FIRFilter2, 0x01);
    else bd34301WriteRegister(FIRFilter2, 0x04);
  } else if ((FS == 176) || (FS == 192)) {
    bd34301WriteRegister(Clock1, 0x00);
    bd34301WriteRegister(FIRFilter1, 0x04);
    bd34301WriteRegister(DeltaSigma, 0x11);
    if (digiFil == 1) bd34301WriteRegister(FIRFilter2, 0x02);
    else bd34301WriteRegister(FIRFilter2, 0x05);
  } else if ((FS == 352) || (FS == 384)) {
    bd34301WriteRegister(Clock1, 0x00);
    bd34301WriteRegister(FIRFilter1, 0x08);
    bd34301WriteRegister(DeltaSigma, 0x11);
    bd34301WriteRegister(FIRFilter2, 0x80);
  } else if ((FS == 705) || (FS == 768)) {
    bd34301WriteRegister(Clock1, 0x00);
    bd34301WriteRegister(FIRFilter1, 0x08);
    bd34301WriteRegister(DeltaSigma, 0x01);
    bd34301WriteRegister(FIRFilter2, 0x80);
  }
  bd34301WriteRegister(Clock2, 0x00);
  bd34301WriteRegister(AudioIF1, 0x0B);
  bd34301WriteRegister(Setting5, 0x16);
  bd34301WriteRegister(Setting6, 0x16);
}

// DSD モード時の設定
void sequenceThree(uint16_t FS) {
  bd34301WriteRegister(Clock1, 0x00);
  bd34301WriteRegister(Clock2, 0x00);
  bd34301WriteRegister(AudioIF1, 0x8B);
  if (FS == 2822) bd34301WriteRegister(DSDFilter, 0x01);       // fc = 26kHz
  else if (FS == 5644) bd34301WriteRegister(DSDFilter, 0x00);  // fc = 26kHz
  else if (FS == 11289) bd34301WriteRegister(DSDFilter, 0x00); // fc = 52kHz
  else if (FS == 22579) bd34301WriteRegister(DSDFilter, 0x00); // fc = 104kHz
  bd34301WriteRegister(DeltaSigma, 0x02);
  bd34301WriteRegister(Setting5, 0x9E);
  bd34301WriteRegister(Setting6, 0x1E);
}

void sequenceFour() {
  bd34301WriteRegister(SoftwareReset, 0x01);   // ソフト・リセット　オフ
  bd34301WriteRegister(DigitalPower, 0x01);    // デジタル・パワー　オン
  bd34301WriteRegister(RAMClear, 0x80);        // ラム・クリア　オン
  bd34301WriteRegister(RAMClear, 0x00);        // ラム・クリア　オフ
  bd34301WriteRegister(Mute, 0x03);            // ミュート オフ
}

void sequenceFive() {
  bd34301WriteRegister(SoftwareReset, 0x01);
  bd34301WriteRegister(DigitalPower, 0x01);
  bd34301WriteRegister(Mute, 0x03);
}
```

変更履歴

Date(Y/M/D)	リビジョン	改訂理由	ページ	改訂内容
2021/10/3	0.9	Draft		
2021/12/11	1.0	初版		