Audio peripherals, PWM

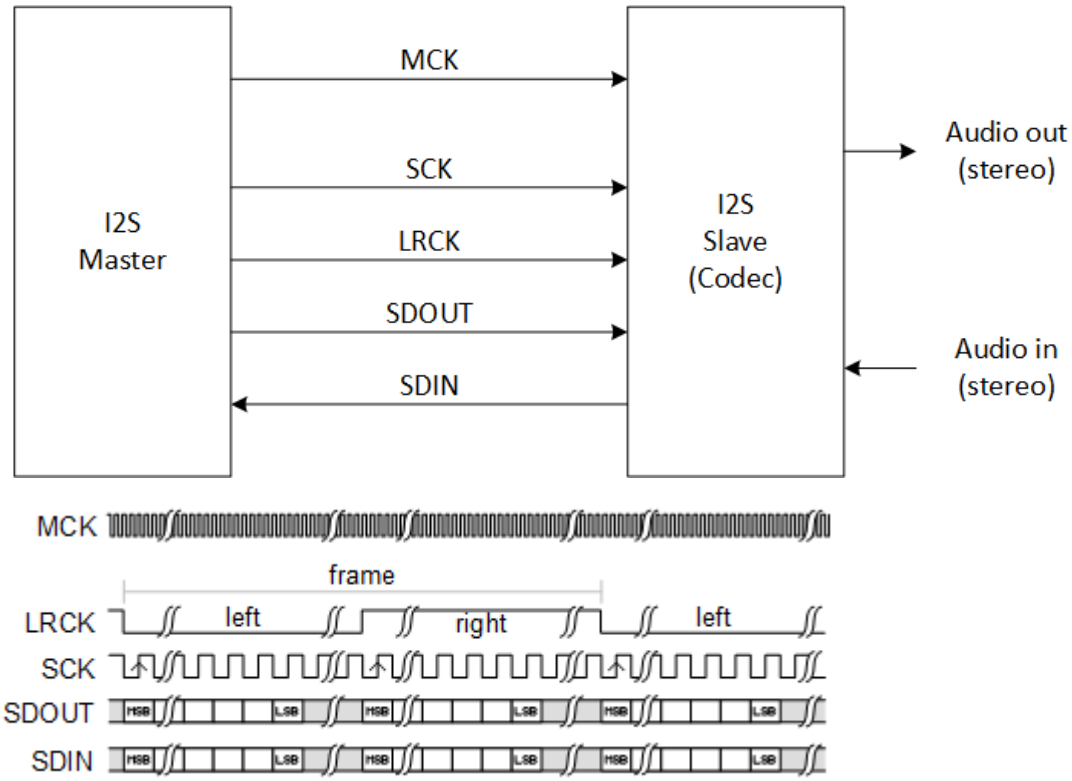
nRF52 Global Tech Tour

Outline

- I2S
- PDM
- PWM
- Demo

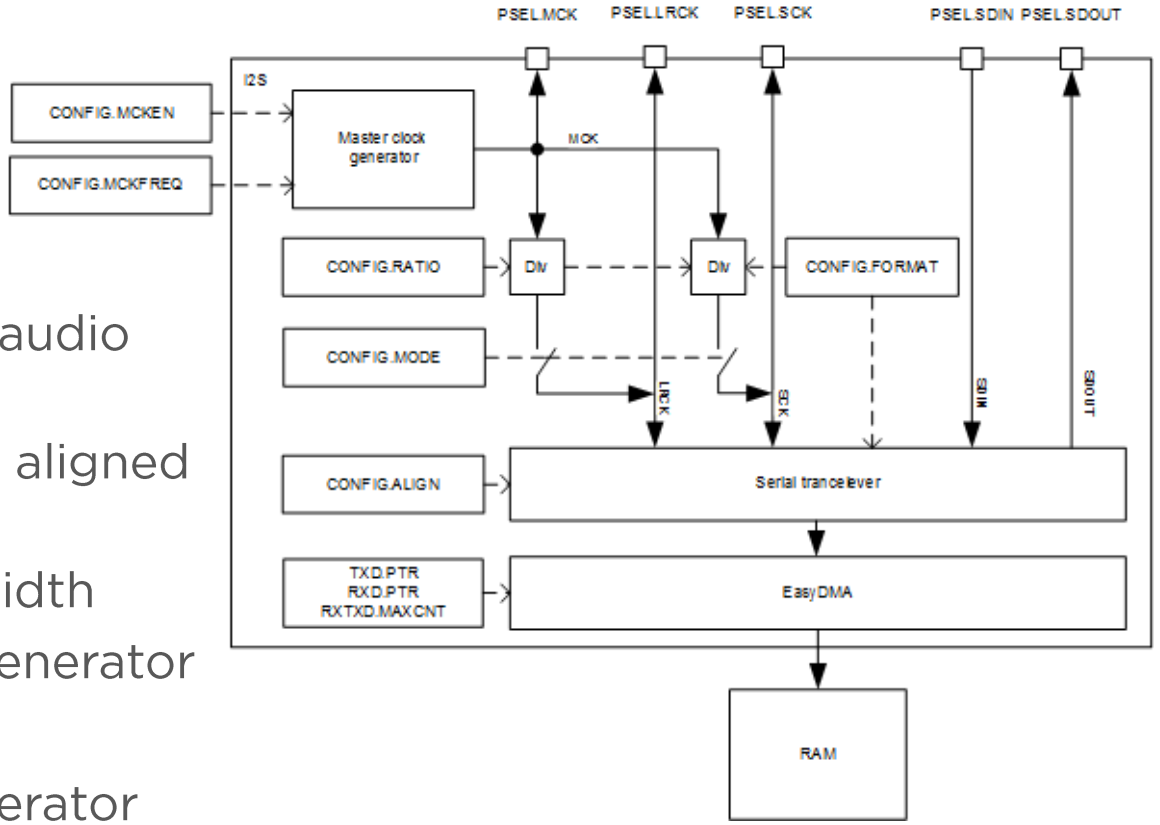
I2S interface

- 4-wire full duplex synchronous peer to peer interface
- Industry standard for interfacing codecs



I2S General Operation

- Master/slave operation
- Simultaneous TX and RX audio streaming
- Original I2S and left/right aligned format
- 8, 16 and 24-bit sample width
- Low-jitter Master Clock generator
- Various sample rates
- Local SCK and LRCK generator

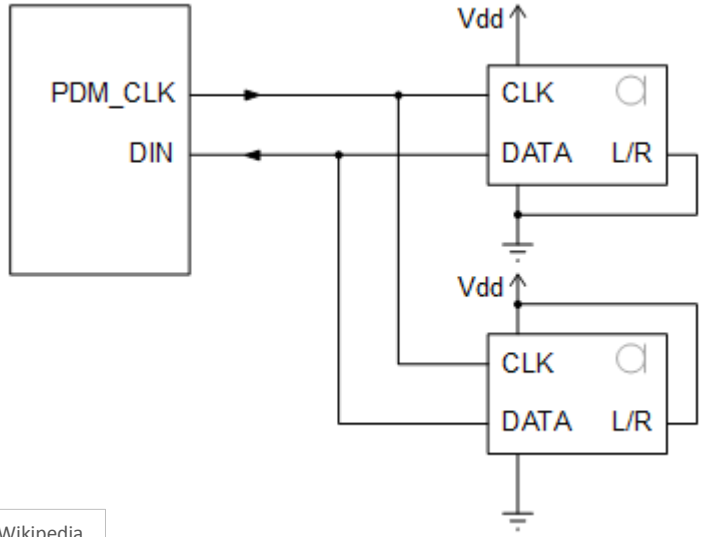


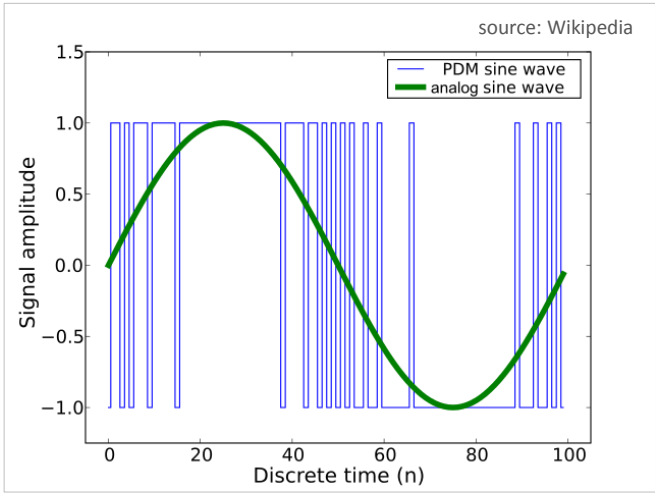
Outline

- I2S
- **PDM**
- PWM
- Demo

PDM Interface

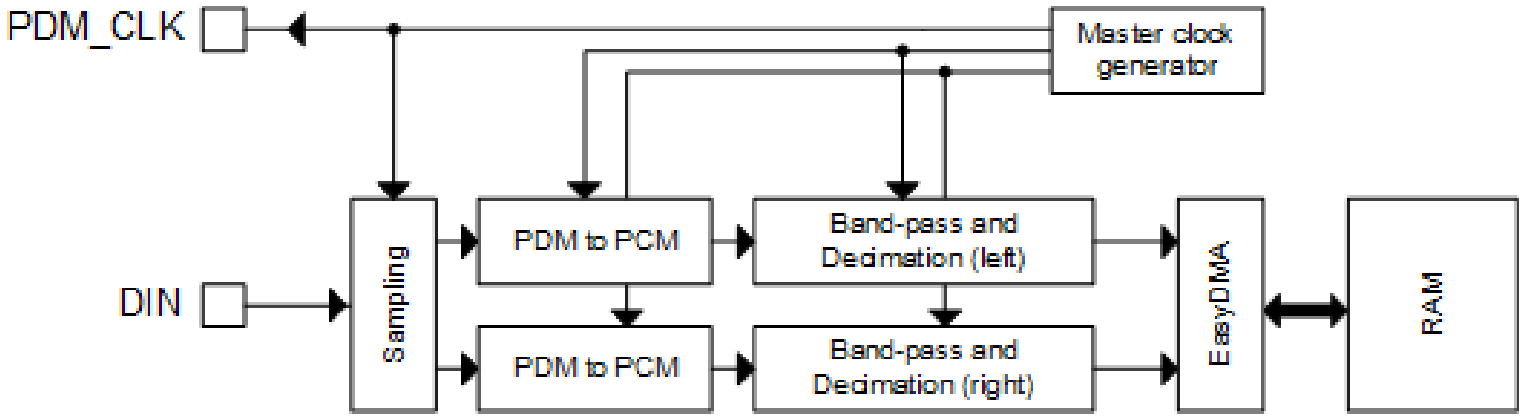
- Two wire digital audio interface
- Industry de-facto interface to digital microphones
- Up to two microphones on two wires
- Requires transforming Pulse Density Modulation into PCM samples (digital filter)





PDM General Operation

- Master operation
- PDM clock generated by the nRF52 – no additional xtal
- up to two PDM microphones on the same pair of wires
- built-in digital PDM-to-PCM filter (incl. decimation)
- outputs 16k samples per second through EasyDMA
  - Analog performance (SNR) dominated by microphone
  - M4 DSP is available for EQ and BLE compression



Outline

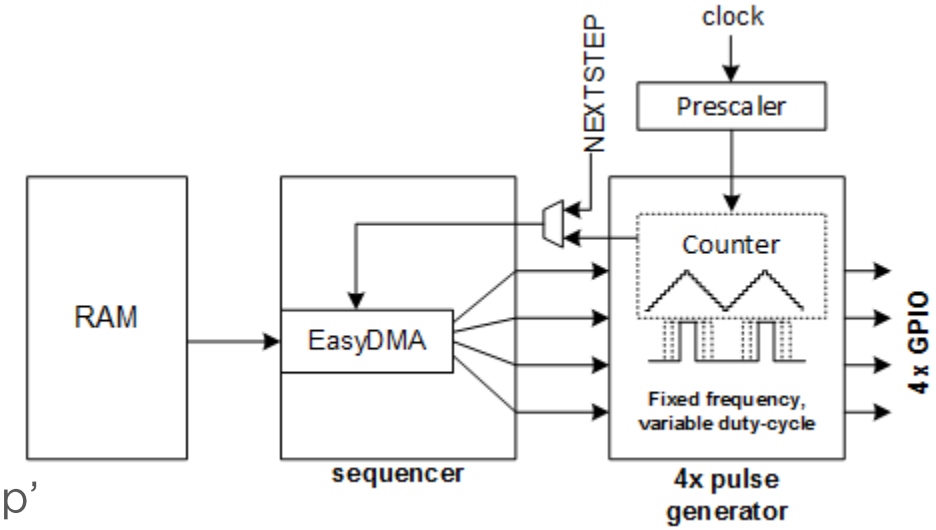
- I2S
- PDM
- **PWM**
- Demo

PWM

- Fully autonomous, glitch-free PWM generator
- Fixed PWM base frequency with programmable clock divider

- 4 synchronous PWM channels
- Individual duty-cycle and polarity
- Edge- or center aligned pulses

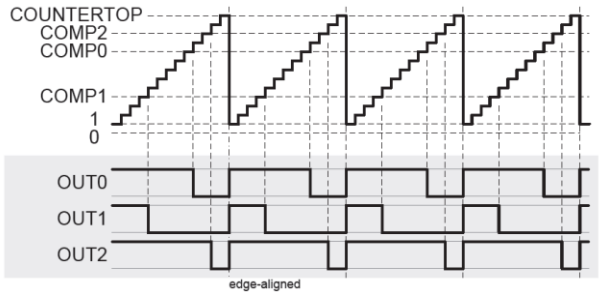
- Reads sequence tables from RAM
  - EasyDMA
  - Polarity and duty-cycle in every ‘step’
  - Sequences can be repeated or looped

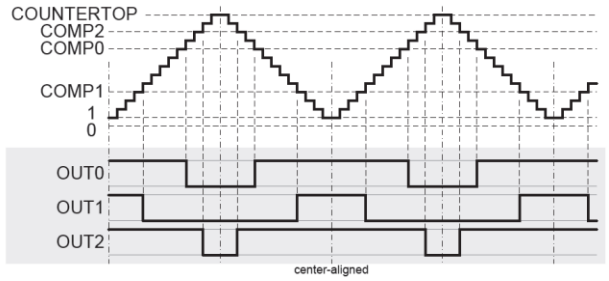


- 3 instances of 4 channels each in nRF52832

Wave Generator

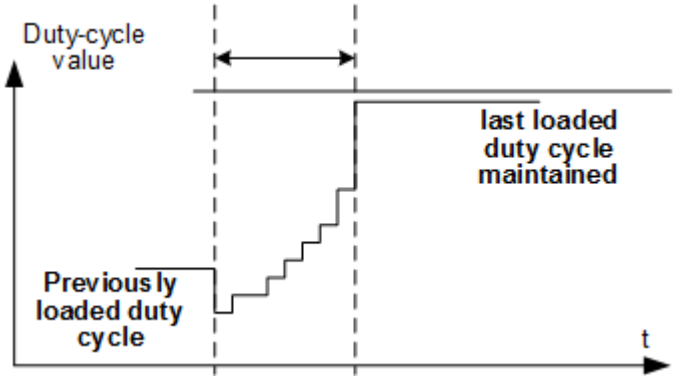
- Wave Generator built around a 15 bit counter
- programmable top counter value
  - Optionally updated in sync with duty cycles
- built-in prescaler
- up to 4 synchronous channels
- edge- or centered-aligned
- values loaded directly from RAM through EasyDMA
- polarity loaded along with the duty-cycle value (16th bit)
- glitch-free value updates



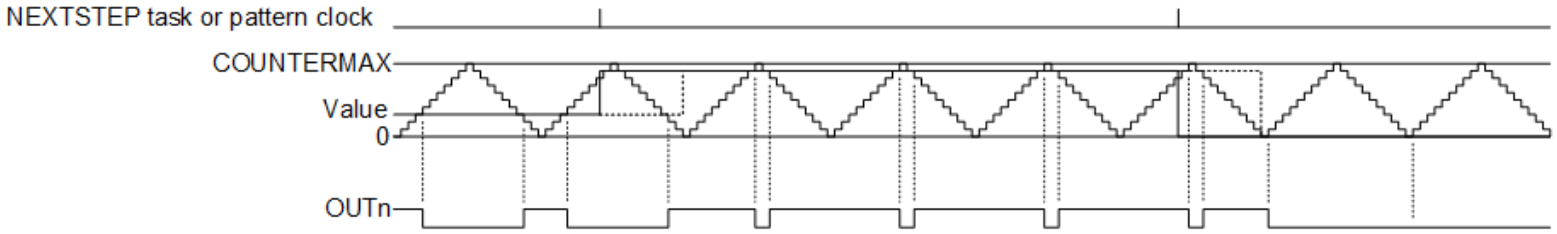


Sequence

- Successive duty-cycles values in RAM

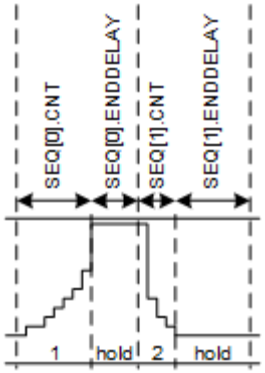


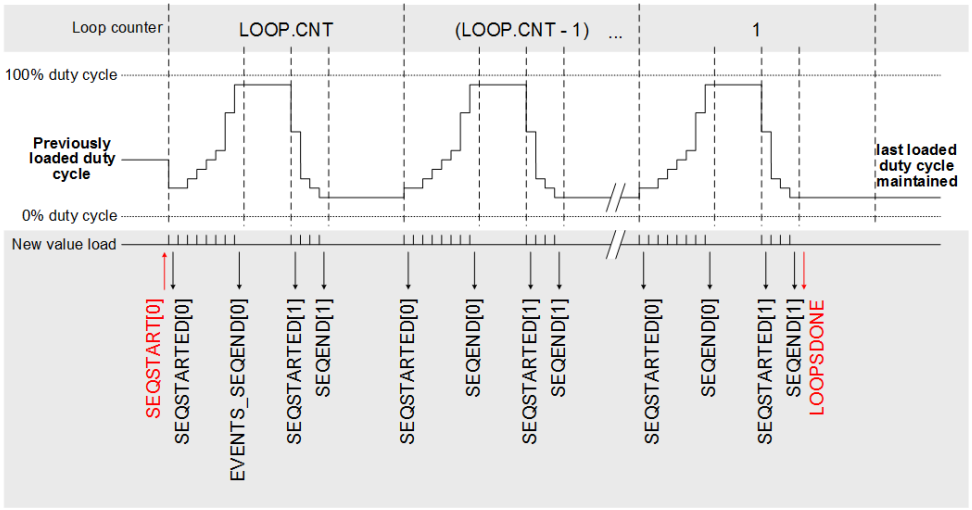
- EasyDMA loads successive values from RAM into the wave generator
- programmable rate (every n PWM period), or use PPI and task
- Grouped or individual value load (1-2-4)



Complex patterns

- Inspired from “fade-in, wait, fade-out, wait”
- Sequencer combining two different sequences
- Programmable pause between sequences
- Programmable amount of loops

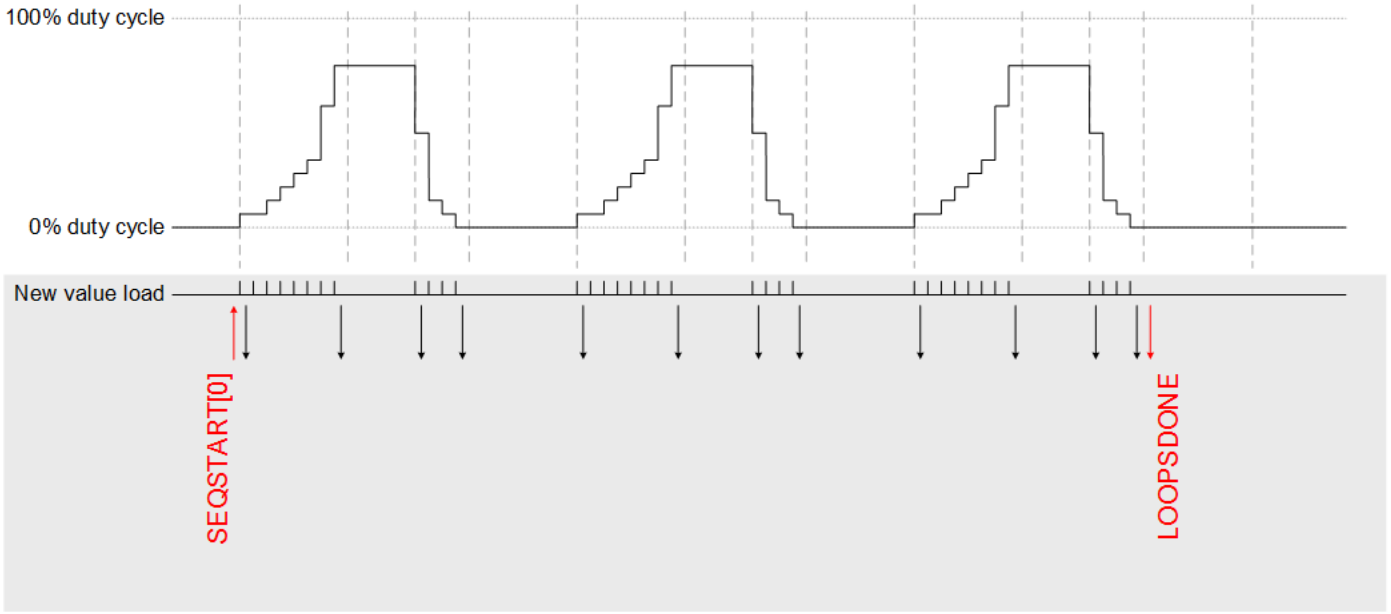




- With some help from the software, more than two sequences possible

Example: Fade LEDs in and out independently, end with disabled

- Actual use:
  - Send SEQSTART[0] - this starts the first complex pattern, i.e. fade-in
  - Do what you want, the CPU is all yours
  - Software will get notified when the last sequence has been played back.



Outline

- I2S
- PDM
- PWM
- **Demo**

PWM initialization code

main.c* | arm_startup_nrf52.s | nrf52_bitfields.h | nrf_gpio.h | nrf_drv_gpiote.h | app_pwm.c | nrf51_to_nrf52.h | sinetable.h | nrf_delay.h | pca1(

```
1101
1102  void hp_pwm_config(void)
1103  {
1104    NRF_PWM0->PSEL.OUT[0] =   (MYHP << PWM_PSEL_OUT_PIN_Pos)
1105                            | (PWM_PSEL_OUT_CONNECT_Connected << PWM_PSEL_OUT_CONNECT_Pos);
1106    NRF_PWM0->PSEL.OUT[1] =   (0 << PWM_PSEL_OUT_PIN_Pos)
1107                            | (PWM_PSEL_OUT_CONNECT_Disconnected << PWM_PSEL_OUT_CONNECT_Pos);
1108    NRF_PWM0->PSEL.OUT[2] =   (1 << PWM_PSEL_OUT_PIN_Pos)
1109                            | (PWM_PSEL_OUT_CONNECT_Disconnected << PWM_PSEL_OUT_CONNECT_Pos);
1110    NRF_PWM0->PSEL.OUT[3] =   (2 << PWM_PSEL_OUT_PIN_Pos)
1111                            | (PWM_PSEL_OUT_CONNECT_Disconnected << PWM_PSEL_OUT_CONNECT_Pos);
1112
1113    NRF_PWM0->MODE = (PWM_MODE_UPDOWN_Up << PWM_MODE_UPDOWN_Pos);
1114    NRF_PWM0->PRESCALER = (PWM_PRESCALER_PRESCALER_DIV_1 << PWM_PRESCALER_PRESCALER_Pos);
1115    NRF_PWM0->COUNTERTOP = (256 << PWM_COUNTERTOP_COUNTERTOP_Pos);
1116    NRF_PWM0->DECODER =   (PWM_DECODER_LOAD_Common << PWM_DECODER_LOAD_Pos)
1117                        | (PWM_DECODER_MODE_RefreshCount << PWM_DECODER_MODE_Pos);
1118
1119    NRF_PWM0->SHORTS = (PWM_SHORTS_LOOPSDONE_SEQSTART0_Disabled << PWM_SHORTS_LOOPSDONE_SEQSTART0_Pos);
1120
1121    NRF_PWM0->ENABLE = (PWM_ENABLE_ENABLE_Enabled << PWM_ENABLE_ENABLE_Pos);
1122
1123    NRF_PWM0->SEQ[0].PTR = ((uint32_t)(tr707_bd) << PWM_SEQ_PTR_PTR_Pos);
1124    NRF_PWM0->SEQ[0].CNT = ((sizeof(tr707_bd) / sizeof(uint16_t)) << PWM_SEQ_CNT_CNT_Pos);
1125    NRF_PWM0->SEQ[0].REFRESH = REFRESHBD;
1126    NRF_PWM0->SEQ[0].ENDDELAY = REFRESHBD * ((HALFLOOPPERIOD / REFRESHBD) - ((sizeof(tr707_bd) / sizeof(uint16_t))));
1127
1128    NRF_PWM0->SEQ[1].PTR = ((uint32_t)(tr707_sd) << PWM_SEQ_PTR_PTR_Pos);
1129    NRF_PWM0->SEQ[1].CNT = ((sizeof(tr707_sd) / sizeof(uint16_t)) << PWM_SEQ_CNT_CNT_Pos);
1130    NRF_PWM0->SEQ[1].REFRESH = REFRESHSD;
1131    NRF_PWM0->SEQ[1].ENDDELAY =  REFRESHSD * ((HALFLOOPPERIOD / REFRESHSD) - ((sizeof(tr707_sd) / sizeof(uint16_t))));
1132
1133    NRF_PWM0->LOOP = (8 << PWM_LOOP_CNT_Pos);
1134  }
1135
1136  void led_pwm_config(void)
```

PWM start playing

```
case 1: //Bass drum
  NRF_PWM0->LOOP = (0 << PWM_LOOP_CNT_Pos);
  NRF_PWM0->TASKS_SEQSTART[0] = 1;
  break;
```