

ダミー・データを作成し、マイク音声データとして
ホストに転送する

4チャンネルのマイク入力… audio_4_channel_mic

関本 健太郎



図1 audio_4_channel_micのシステム構成

ここからは、TinyUSBの個々のサンプル・プログラムについて解説します。(編集部)

動作

audio_4_channel_micは、4チャンネルのUSBマイクのサンプル・プログラムです。プログラムの中でダミー・データを作成し、そのデータをマイク音声データとしてホストに転送します(図1)。デフォルト

リスト1 メイン・ループ処理main.cのmain関数

```
int main(void)
{
    board_init();
    // TinyUSBデバイス初期化処理
    tud_init(BOARD_TUD_RHPORT);
    // サンプル・レートなどの変数の初期化
    sampFreq = AUDIO_SAMPLE_RATE;
    clkValid = 1;
    sampleFreqRng.wNumSubRanges = 1;
    sampleFreqRng.subrange[0].bMin =
        AUDIO_SAMPLE_RATE;
    sampleFreqRng.subrange[0].bMax =
        AUDIO_SAMPLE_RATE;
    sampleFreqRng.subrange[0].bRes = 0;
    while (1)
    {
        tud_task(); // TinyUSBデバイス・タスク処理
        省略
        audio_task(); // AUDIOクラス・タスク処理
    }
    return 0;
}
```

リスト2 何も処理を行わないmain.cのaudio_task関数

```
void audio_task(void)
{
    // 現時点ではなにも処理していません
}
```

で作成されるダミー・データは、0, 1, 2, …と非常に小さい値なので、ホストPCで確認するにはちょっとした工夫が必要です。

ディスクリプタ

USBディスクリプタ(図2)として、インターフェース・ディスクリプタを2つ持ちます。

1つはAudio Controlでエンドポイントはゼロ、つまりデフォルトのエンドポイントを使用します。もう1つはAudio Streamingでエンドポイントは1つ、IN方向(デバイスからホストへ)のアイソクロナス・エンドポイントを使っています。

その他に、インターフェース・アソシエーション・ディスクリプタが存在し、USB Audioクラスとして、2つのインターフェースを持つことを定義しています。

プログラム

プログラムはmain関数の先頭でTinyUSBのUSBデバイス初期化処理を行い、メイン・ループ処理を実行しています。メイン・ループ処理では、USBデバイスのタスク処理をtud_taskで実行し、USB CDCクラスに入力されたデータをcdc_taskで処理します(リスト1)。

audio_task関数では何も処理が行われていません(リスト2)。

その代わりに、ほとんどの処理がAudioクラスで実

リスト3 ホストPCへの音声データの送信を行うmain.cのtud_audio_tx_done_pre_load_cbコールバック関数

```
bool tud_audio_tx_done_pre_load_cb(uint8_t rhport,
uint8_t itf, uint8_t ep_in, uint8_t cur_alt_setting)
{
    省略
    for (uint8_t cnt=0; cnt <
        CFG_TUD_AUDIO_FUNC_1_N_TX_SUPP_SW_FIFO;
        cnt++)
    {
        tud_audio_write_support_ff(cnt,
            i2s_dummy_buffer[cnt], AUDIO_SAMPLE_RATE/1000
            * CFG_TUD_AUDIO_FUNC_1_N_BYTES_PER_SAMPLE_TX
            * CFG_TUD_AUDIO_FUNC_1_CHANNEL_PER_FIFO_TX);
    }
    return true;
}
```