

プロンプト

“pico2のadc_fifo_get()でAMラジオの検波を想定している。

中波AMラジオからの信号をアナログ回路のヘテロダイン回路でIF(中間周波数)=150kHzにする。

IF出力をpico2のアナログ入力とする。

adc_fifo_get()でAD変換する

検波は低周波FIRフィルターを用いる。タップ数=21

AM検波として、まず整流(絶対値)を施し、

低周波フィルターFIRフィルタでエンベロープ抽出

FIRフィルターはライブラリを用いずにfir_filterとして関数定義する。

FIRフィルター関数は値を返さない。結果は出力パラメータ(結果変数への参照)を変更する

低周波カット周波数は5kHz

FIRフィルター内部ではスケールファクター128で計算する

FIRフィルターでスケールダウンするときは割り算「/」ではなくビットシフトする

FIRフィルター関数ではオーバーフローを検知する。

FIRに関する変数、定期、定数宣言はなるべく関数内部で定義する

オーバーフローは符号付き整数12bitを基準に厳密に反映する。

以下を用いて係数を計算する：

係数はhamming窓、LPF 5kHz, サンプリング周波数=300kHz, scale=128

オーバーフロー計算で以下は間違いなので使わないこと

if (acc > (INT16_MAX * scale_factor) || acc < (INT16_MIN * scale_factor))

パラメータとして入力されたbool変数(の参照)へオーバーフロー結果を書く。

オーバーフローありtrue, なしfalse

検波結果をPWMでDA変換してGPIOへ出力。

(12-bit解像度相当)

pico2のクロック周波数150MHzを考慮。

FIRフィルターのクロックを一定にするためにIRQ、DMAを用いる。

FIR関数はハンドラから呼び出す。

割り込みハンドラとFIRフィルター関数は__not_in_flash_funcにする。

割り込みハンドルから呼ばれる関数はすべて、`__not_in_flash_tunc`にする。

PWMからの音声出力を平滑化するためのアナログ回路例を示す。

平滑化の回路は2段RC回路とLC回路の例を示し、簡単にメリットを説明する。

平滑回路は音声向けとして典型的なカットオフ周波数とする。

AAでRC回路とLC回路の回路図を別々に示す。

AAの回路図中に素子定数を示す。

平滑回路の周波数特性をグラフ（画像）で示す。

平滑回路を使ったRC回路とLC回路の周波数特性をグラフに含める。

2つの周波数特性の曲線を1つのグラフの同じ座標上に含める。

周波数の範囲は1kHzから100kHz

グラフの横軸に周波数を値を含める。

C++コード例を示す。

C++文法はC++20最新の組み込み向けベストプラクティス。

コード内のコメントは英語とする。

C++ソースの先頭に今日の日付を書く（米式）

C++ソースの先頭にGrokをベースにした旨のコメントを書く。

C++ソースの先頭に"prompt by tommie.jp (GitHub:open-tommie)"と書く。

回答は日本語。

回答の冒頭に「プロンプト」と題して、このプロンプト全体を引用する。

次に仕様のまとめ。

次にコード例

次にPWM平滑化アナログ回路例

次に回路図

次に平滑回路の周波数特性

仕様のまとめ

このシステムは、Raspberry Pi Pico 2 (RP2350) を使用して、中波AMラジオ信号をヘテロダイ

・回路設計・回路構成・回路分析・回路実装・回路評価・回路改修・回路検証・回路最適化

ン回路 (FIFO=150KHz)に変換し、PICO Zのノアログ入力に接続します。ADCを300KHzサンプリングで使用し、FIFOからデータを取得。AM検波として、まずADC値を符号付き12bitに変換後、絶対値（整流）を取り、タップ数21の低周波FIRフィルタ（カットオフ5kHz、Hamming窓、scale=128）でエンベロープを抽出します。FIRフィルタは自作関数で実装し、ビットシフトでスケーリング、オーバーフローを12bit符号付き整数基準で検知（出力超過でtrue）。処理タイミングを一定にするため、ADC FIFO IRQを使用（DMA未使用でシンプルに）。割り込みハンドラからFIRを呼び、すべてRAM実行（__not_in_flash_func）。検波結果をPWMで12bit相当のDA変換（wrap=4095、f_pwm≈36.6kHz）し、GPIO出力。PWM出力はアナログ平滑回路で音声化。

コード例

cpp

× 折りたたむ ≡ 広げる ▶ 実行 Ⓛ コピー

```
// 10/19/2025
// Based on Grok by xAI
// prompt by tommie.jp (GitHub:open-tommie)

#include <cstdlib> // For std::abs
#include <cstdint> // For fixed-width integers
#include "pico/stdlib.h"
#include "hardware/adc.h"
#include "hardware/irq.h"
#include "hardware/pwm.h"

constexpr uint ADC_PIN = 26; // GPIO26 as analog input (ADC0)
constexpr uint PWM_PIN = 0; // GPIO0 for PWM output
constexpr float SYS_CLK_MHZ = 150.0f;
constexpr float SAMPLE_RATE_KHZ = 300.0f;
constexpr uint16_t PWM_WRAP = 4095; // For 12-bit resolution equivalent

// FIR filter function (called from handler)
__not_in_flash_func(void fir_filter)(int16_t new_sample, int16_t* output,
bool& overflow) {
    // Define constants and variables inside the function as much as
    // possible
    constexpr int NUM_TAPS = 21;

    constexpr
    int SCALE_FACTOR = 128;
```

```

constexpr int16_t COEFFS[NUM_TAPS] = {1, 1, 2, 3, 5, 6, 8, 10, 11, 12,
12, 11, 10, 8, 6, 5, 3, 2, 1, 1};
static int16_t buffer[NUM_TAPS] = {0}; // History buffer

// Shift buffer to make room for new sample
for (int i = NUM_TAPS - 1; i > 0; --i) {
    buffer[i] = buffer[i - 1];
}
buffer[0] = new_sample;

// Compute FIR: accumulate with scaling
int32_t acc = 0;
for (int i = 0; i < NUM_TAPS; ++i) {
    acc += static_cast<int32_t>(buffer[i]) * COEFFS[i];
}

// Scale down using bit shift
*output = static_cast<int16_t>(acc >> 7); // Divide by 128

// Detect overflow based on signed 12-bit range (-2048 to 2047)
constexpr int16_t INT12_MAX = 2047;
constexpr int16_t INT12_MIN = -2048;
overflow = (*output > INT12_MAX) || (*output < INT12_MIN);
}

// ADC IRQ handler
__not_in_flash_func(void adc_handler)() {
    // Get ADC sample from FIFO
    uint16_t raw_sample = adc_fifo_get();

    // Convert to signed 12-bit
    int16_t sample = static_cast<int16_t>(raw_sample - 2048);

    // Rectify (absolute value)
    int16_t rectified = std::abs(sample);

    // Apply FIR filter
    int16_t filtered_output;
    bool overflow_detected = false;
    fir_filter(rectified, &filtered_output, overflow_detected);

    // Handle overflow if needed (e.g., clip)
    if (overflow_detected) {
        // Clip or handle overflow
    }
}

```

```

    if (overflow_detected) {
        filtered_output = (filtered_output > 2047) ? 2047 : -2048;
    }

    // Convert to unsigned for PWM (0-4095 range)
    uint16_t pwm_level = static_cast<uint16_t>(filtered_output + 2048) * 2;
    // Scale to 12-bit unsigned equivalent

    // Set PWM level
    uint pwm_slice = pwm_gpio_to_slice_num(PWM_PIN);
    uint pwm_chan = pwm_gpio_to_channel(PWM_PIN);
    pwm_set_chan_level(pwm_slice, pwm_chan, pwm_level);

    // Clear IRQ
    adc_hw->ints = ADC_FIFO_BITS;
}

int main() {
    // Initialize stdio
    stdio_init_all();

    // Set system clock to 150 MHz
    set_sys_clock_khz(static_cast<uint>(SYS_CLK_MHZ * 1000), true);

    // Initialize ADC
    adc_init();
    adc_gpio_init(ADC_PIN);
    adc_select_input(0); // ADC0
    adc_fifo_setup(true, false, 1, false, false); // FIFO enabled, IRQ on
threshold 1

    // Set ADC clock divider for ~300 kHz sampling
    float adc_clkdiv = (SYS_CLK_MHZ * 1e6) / SAMPLE_RATE_KHZ / 1000 - 1.0f;
    adc_set_clkdiv(adc_clkdiv);

    // Enable ADC FIFO IRQ
    irq_set_exclusive_handler(ADC_IRQ_FIFO, adc_handler);
    irq_set_enabled(ADC_IRQ_FIFO, true);
    adc_hw->inte = ADC_FIFO_BITS;

    // Start ADC in free-running mode
    adc_run(true);
}

```

```

// Initialize PWM
gpio_set_function(PWM_PIN, GPIO_FUNC_PWM);
uint pwm_slice = pwm_gpio_to_slice_num(PWM_PIN);
pwm_set_wrap(pwm_slice, PWM_WRAP);
pwm_set_chan_level(pwm_slice, pwm_gpio_to_channel(PWM_PIN), 0);
pwm_set_enabled(pwm_slice, true);

// Main loop (idle)
while (true) {
    tight_loop_contents();
}

return 0;
}

```

PWM平滑化アナログ回路例

PWM出力からの音声信号を平滑化するためのアナログ回路として、2段RC低域通過フィルタとLC低域通過フィルタの例を示します。これらはPWMのキャリア周波数（約36.6kHz）を除去し、音声帯域（典型的に20Hz～20kHz、ただしここではカットオフ約5kHz程度）を通過させるために使用します。音声向け典型的なカットオフ周波数は5～10kHzとして設計し、PWMの高周波成分を十分に減衰させます。

- ・**2段RC回路のメリット:** 部品が安価で入手しやすく、シンプルに実装可能。受動回路なので電源不要。ただし、減衰率が緩やか (-40dB/decade) で、高周波ノイズ除去がLCほど急峻でない。
- ・**LC回路のメリット:** 減衰率が急峻 (-40dB/decadeだが、共振特性によりシャープ) で、低損失。音声品質が良好だが、コイルが必要でコストが高く、サイズが大きい。

回路図

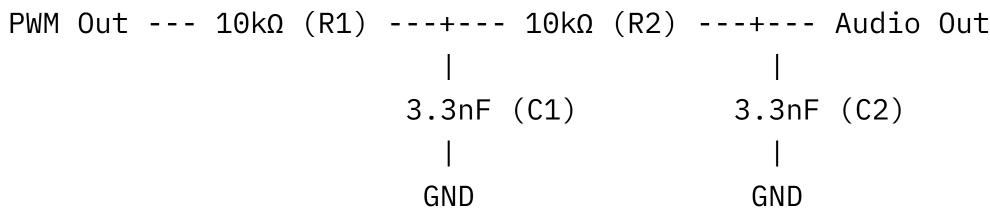
2段RC回路のAA回路図（カットオフ約5kHz、各段R=10kΩ, C=3.3nF）：

text

✗ 折りたたむ

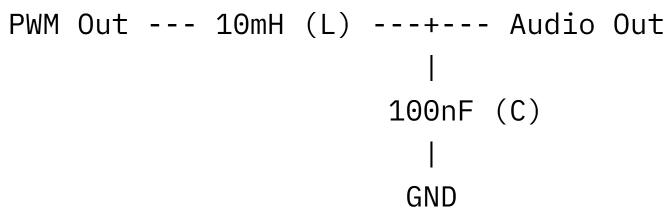
≡ 広げる

○ コピー



LC回路のAA回路図（カットオフ約5kHz, L=10mH, C=100nF）：

text
 折りたたむ
 広げる
 コピー



平滑回路の周波数特性

以下は、2段RC低域通過フィルタとLC低域通過フィルタの周波数特性を比較したグラフ（振幅 [dB] vs 周波数[Hz], ログスケール, 1kHz～100kHz範囲）です。両者の曲線を同じ座標上に示しています。このグラフは2次フィルタの典型的な応答を示しており、RCは緩やかなロールオフ、LCは共振近辺でシャープな特性が見られます。

2段RCとLC低域通過フィルタの周波数応答比較（2次フィルタ例）

↳ FIR係数計算の詳細

↳ ヘテロダイン回路設計

↳ コードの効率改善