Sistemas Microprocessados - 2017.2
DETI/UFC - Cursos de Eng. de Computação e Eng. de Telecomunicações
Elaborada por Danilo Coutinho, Jorge Reis e Jardel Silveira

PRÁTICA 12 - Reprodução de som gravado em formato wav (DA e Timer)

Objetivos:

- Utilizar a Bluepill para reproduzir som em uma caixa amplificada de som, através de um conversor R2R;
- Utilizar a biblioteca de *timer interrupts* da HAL da STM32 para gerar uma frequência precisa de reprodução das amostras do arquivo wav de som;
- Compreender noções básicas sobre o format wav de arquivos de som.

Softwares utilizados: Audacity, STM32CubeMX, GCC/Atollic e Strip .
Materiais utilizados: Bluepill, PMOD R2R, Jumpers, Microfone, Caixa de som Amplificada e PC .

Montagem do Hardware: Repita a montagem realizada na Prática 10: https://docs.google.com/document/d/1KX4M7UERRx5KIghHFZAo8sCw8HvympNMzNm84B4BIPc/edit

TIMERS

Timers possuem alguns parâmetros básicos, são eles:

1. Prescaler: É o número de pulsos de clock necessário para aumentar em um o valor de Counter Period. Este parâmetro é útil para dividir o clock de entrada em um número mais fácil de trabalhar.

2. Counter Period: Número a ser contado. Após o término da contagem o contador retorna a zero e a contagem recomeça.

3. Counter Mode: A forma que será realizada a contagem, por exemplo: de 0 até Counter Period ou de Counter Period até 0.

Definindo parâmetros: Prescaler e Counter Period

O prescaler é por definição um divisor de frequência. Em um contador, os pulsos de clock gerados **são mandados primeiro ao prescaler** que, ao terminar a contagem, gera um incremento no counter period.

Como exemplo prático, queremos contar 1 segundo a partir do clock de uma placa utilizando contadores. Considerando o STM32Cube define 84MHz de clock para os contadores dela , precisamos então trabalhar sobre o número $84 * 10^6$

Primeiro, definimos o prescaler. Vamos supor um prescaler de 8400.

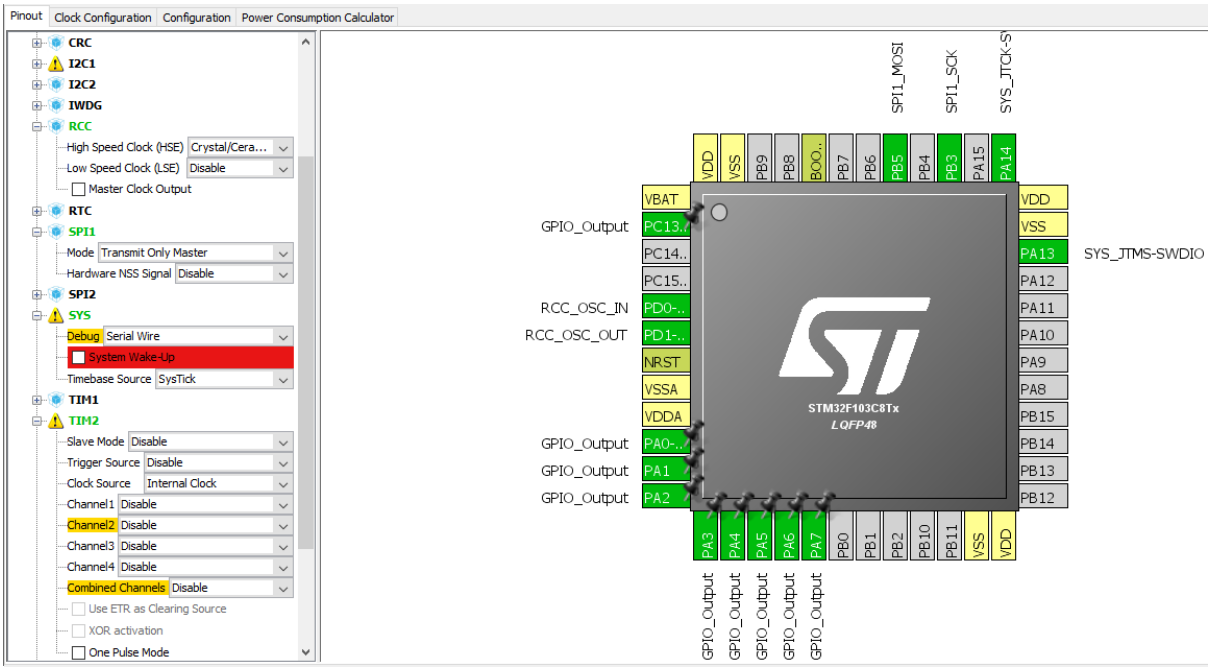
$$(84 * 10^6)/(8400) = 10 * 10^3 Hz$$

Ou seja, a cada 1/10 kHz = 0,1ms o valor do counter period é incrementado.

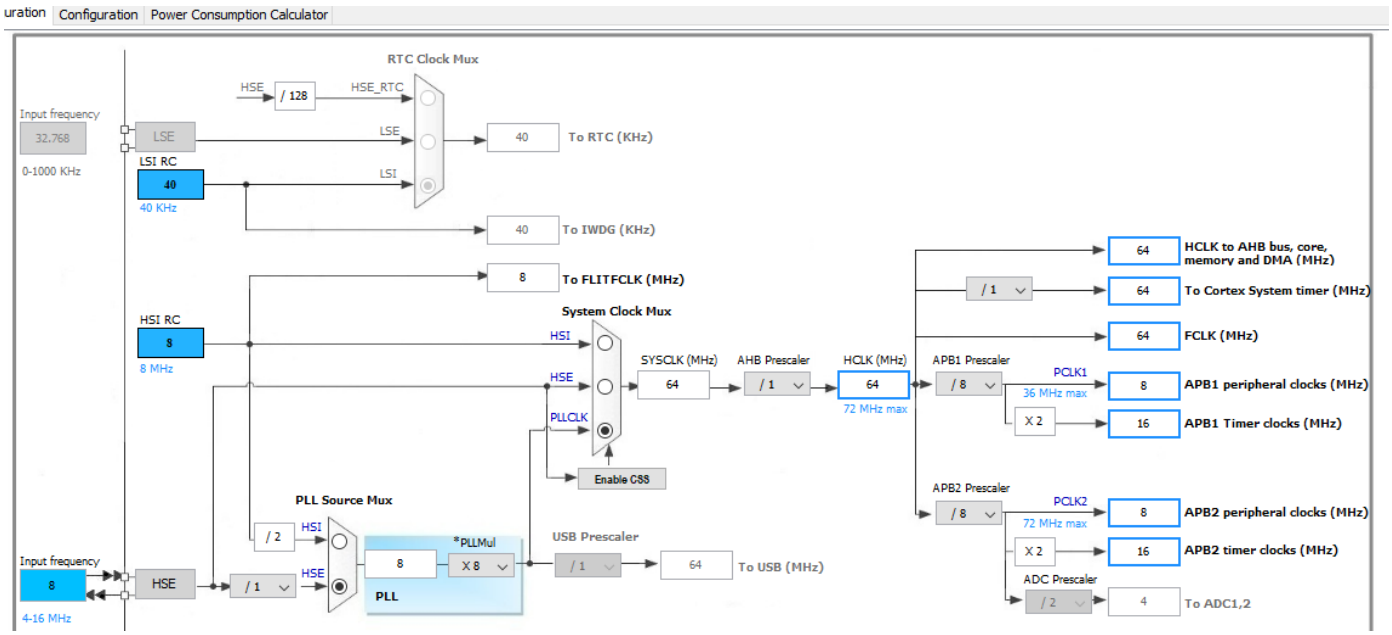
Portanto, se desejamos o valor de 1 segundo, devemos fazer com que o counter period conte 10000 unidades, pois 10^4 * 0,1*10^{-3}s= 1s. Logo o valor ideal será 9999, pois ao incrementar mais um no contador ele retornará a zero.

PRÁTICA

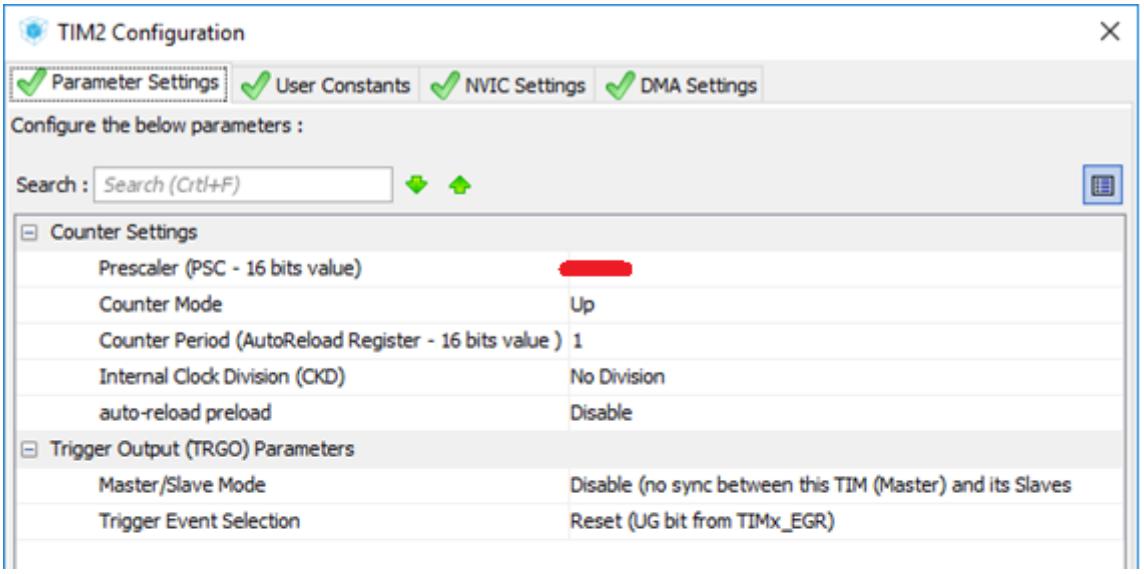
Crie um novo projeto no STM32CubeMX, e configure seu projeto tal como mostrado na figura a seguir. Feito isto, gere o projeto do atollic.



Após configurar os pinos vá para a aba Clock Configuration. A frequência que desejamos para o clock dos periféricos é de 8MHz pois é uma frequência em que conseguimos escutar bem o som sem uma taxa de amostragem muito alta, o que não irá gerar um arquivo .wav muito grande.

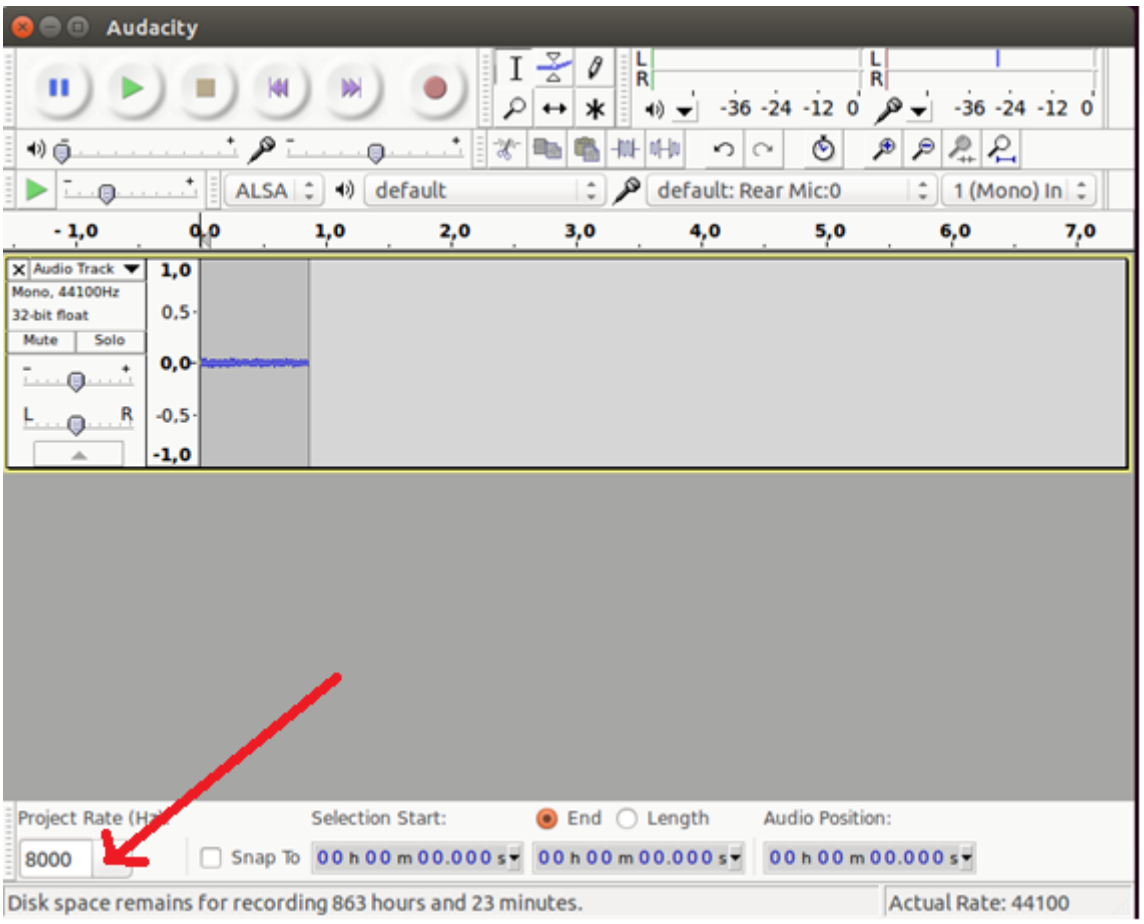


Baseado do clock de 8MHz configurado na janela anterior para o clock dos periféricos, escolha um valor de prescaler para o *timer* 2, de tal forma que a interrupção seja chamada sincronamente com o momento de apresentar as amostras de som para o circuito R2R, de acordo com a taxa de amostragem da gravação de 8000Hz a ser utilizada para gravação do seu áudio no audacity. Você pode manter o counter period como 1. Além disso, é necessário, clicar na aba nvic settings da janela abaixo e habilitar a interrupção geral da tim2.

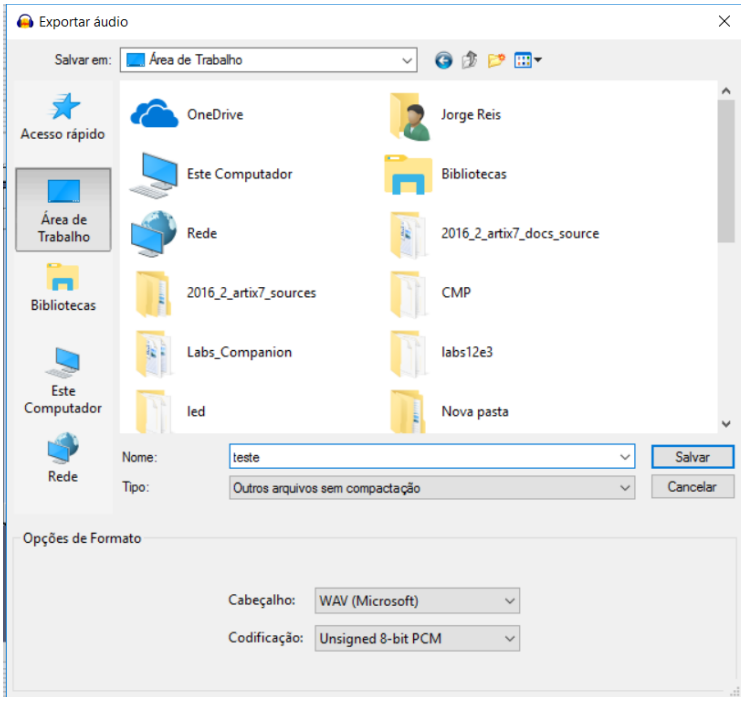


Baixe do SIGAA o arquivo .zip desta prática, e descompacte-o em uma pasta qualquer, de preferência na raiz C: . Execute um Prompt do DOS, vá para a pasta dos arquivos da prática, e deixe essa janela do DOS aberta.

Execute o audacity e selecione 8000Hz na aba de seleção do canto esquerdo inferior, e faça sua curta gravação, uma única palavra. Veja o detalhe no canto superior direito da janela abaixo: **MONO** . A gravação deve ser feita em modo mono.



Após interromper a gravação, vá no menu File -> Export -> wav. No canto direito da janela que se abrirá, selecione *Other Uncompressed Files.* Então novamente selecione wav e unsigned 8 bit PCM. Salve seu wav na mesma pasta que contém o programa strip.exe (arquivos da prática) .

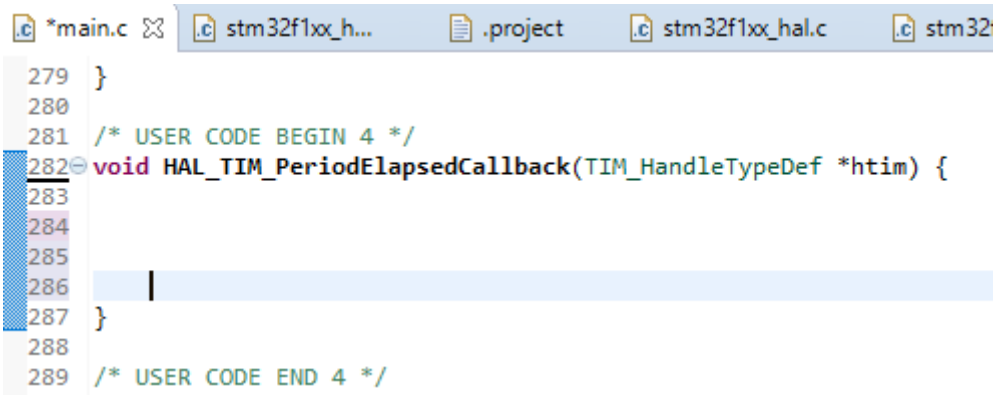


No prompt dos DOS, execute strip meusom.wav > meusom.c

Voltando agora ao Atollic, adicione o arquivo meusom.c ao seu projeto do Atollic.

Chamar na main: HAL_TIM_Base_Start_IT(&htim2);

Essa é a função de callback:



```c
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>
struct wavfile
{
    char        id[4];          // should always contain "RIFF"
    int     totallength;    // total file length minus 8
    char        wavefmt[8];     // should be "WAVEfmt "
    int     format;         // 16 for PCM format
    short     pcm;            // 1 for PCM format
    short     channels;       // channels
    int     frequency;      // sampling frequency
    int     bytes_per_second;
    short     bytes_by_capture;
    short     bits_per_sample;
    char        data[4];        // should always contain "data"
    int     bytes_in_data;
};
int main(int argc, char *argv[]) {
    char *filename=argv[1];
    int nSamples=0;
    FILE *wav = fopen(filename,"rb");
    struct wavfile header;

    if ( wav == NULL ) {
        fprintf(stderr,"Can't open input file %s", filename);
        exit(1);
    }
    // read header
    if ( fread(&header,sizeof(header),1,wav) < 1 )
    {
        fprintf(stderr,"Can't read file header\n");
        exit(1);
    }
    if (    header.id[0] != 'R' || header.id[1] != 'I'
         || header.id[2] != 'F' || header.id[3] != 'F' ) {
        fprintf(stderr,"ERROR: Not wav format\n");
        exit(1); }
    printf ("unsigned const char meusom[]={");
    // read data
    unsigned char value=0;
    unsigned short uvalue;
    fread(&value,sizeof(value),1,wav) ;
        printf("0x%x",value);
        nSamples++;
    while( fread(&value,sizeof(value),1,wav) ) {
        printf(", 0x%x",value);
        nSamples++;
    }
    printf ("};");
    fprintf(stderr,"\n%d samples",nSamples);
}
```

Código fonte do programa strip.c

Referências

http://store.digilentinc.com/pmod-amp2-audio-amplifier/

https://reference.digilentinc.com/reference/pmod/pmodamp2/start#example_projects

http://yannesposito.com/Scratch/en/blog/2010-10-14-Fun-with-wav/

Vieira, Caio Rodrigo de Almeida “GEA- Grupo de Estudos em ARM do CdH: Prática 12- Timers e Interrupção”