



Universidade Estadual de Campinas  
Faculdade de Engenharia Elétrica  
e de Computação



---

# **EA871 - Laboratório de Programação Básica de Sistemas Digitais**

**Roteiro 10 - TPM - PWM**

---

Autores: Fernando Teodoro de Cillo e Rafael Silva Cirino

RA: 197029    223730

Campinas  
Junho de 2022

# Introdução:

O intuito deste experimento é compreender a geração de sinais PWM no microcontrolador MKL25Z128. Outros tópicos abordados serão as diferenças entre aritmética de inteiros e de pontos flutuantes em C, conversão de pontos flutuantes para inteiros e problemas de processamento relacionados aos tipos de dados.

# Experimento:

## 1

- PIT:

- Intervalo de contagem:  $2 \cdot \frac{2097152}{20971520} = 0.2s$
- Intervalo de contagem:  $2 \cdot \frac{20971}{20971520} = 0.002s = 2ms$

- TPM:

- Contagem máxima 255 (256 unidades); PS = 32 => Período =  $\frac{256 \cdot 32}{20971520} \approx 0.39ms$

### 1.a

Na figura 1 vemos que o período medido pelo analisador lógico foi de 2.344 ms para o módulo PIT e na figura 2 que o período para o TPM foi de 0.3901 ms. Para ambos foram encontrados tempos próximos aos esperados, que eram 2 ms e 0.39 ms, respectivamente.

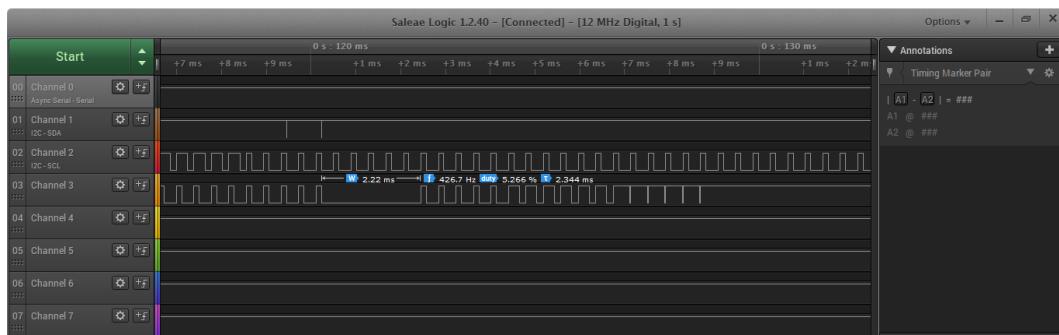


Figura 1: Analisador lógico do módulo PIT

### 1.b

O sinal varia a aproximadamente a cada 39 ms, alterando o valor de contagem para 2097152, temos um período de 0.2 segundos agora. No led foi possível notar a variação de cor entre azul e verde, que antes não era possível, pois ambos estavam piscando de forma muito rápida.

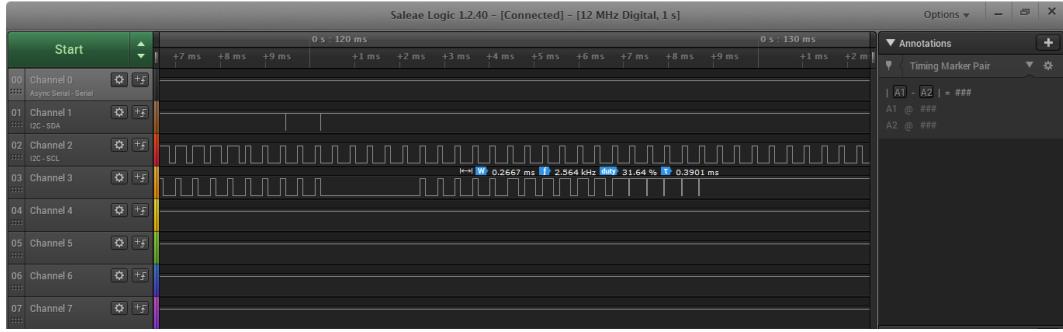


Figura 2: Analisador lógico do módulo TPM

### 1.c

Quando o valor  $< 0$ , o processador considera como 255, fazendo com que o seja ativada uma nova *flag*.

### 1.d

Aumentou o período dos sinais gerados (de 0.3901 para 0.7799 ms, seguindo a mesma proporção do aumento do período), como se vê na figura 3, diminuindo assim a sua frequência. A relação entre a largura dos pulsos e o período dos sinais gerados também foi mantida aproximadamente constante.

$$\frac{0.3901}{0.7799} \approx \frac{255}{511} \approx \frac{0.2667}{0.5392} \approx 0.5$$

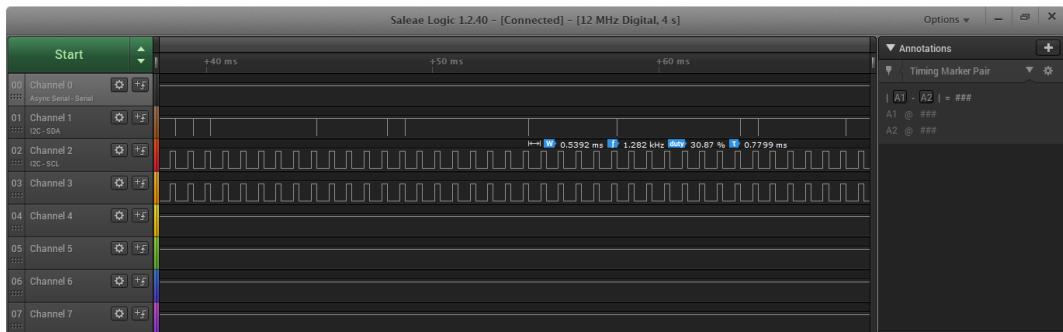


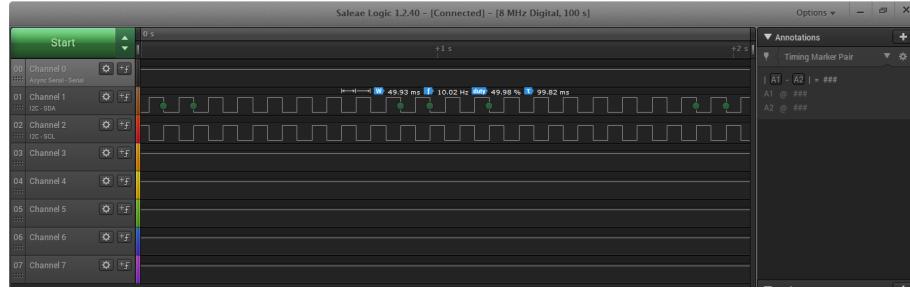
Figura 3: Analisador lógico do módulo TPM com o período aumentado

## 4

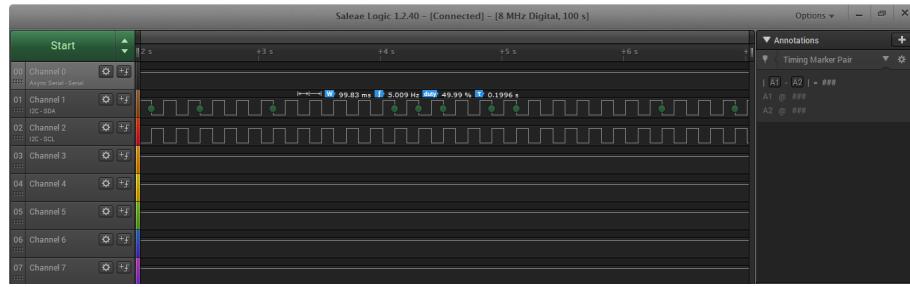
Conforme a tabela disponível no manual, para o PTB0 temos o registrador TPM1\_BASE\_PTR. Para se obter 0,1s, temos o período igual a 65525 e divisor de frequência (ps) igual  $0b101 = 25 = 32$ .

Tabela 1: Variação do período

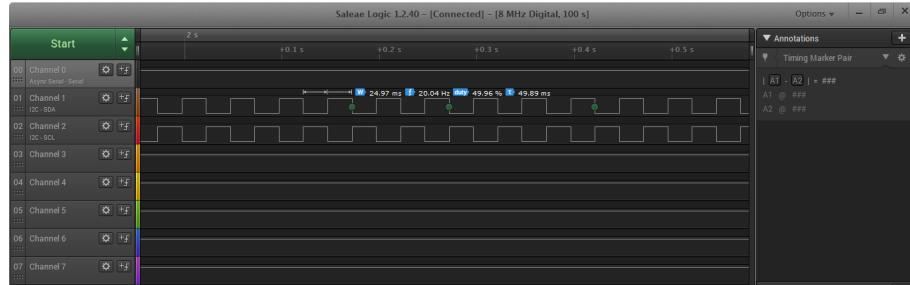
	período	ps	percentagem
0.1s	65525	0b101	50
0.2s	65525	0b110	50
0.05s	65525	0b100	50



(a) Versão original (0.1s)



(b) Alteração para obter 0.2s



(c) Alteração para obter 0.05s

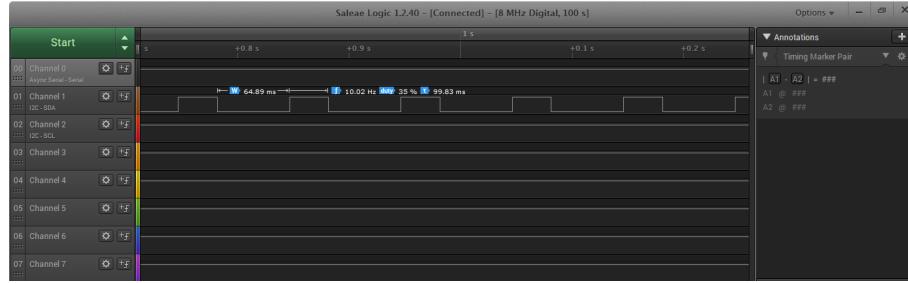
Figura 4: Analisador lógico

## 5

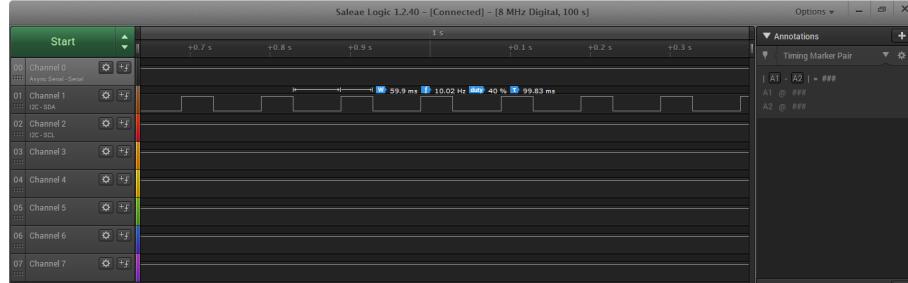
A tabela 2 e a figura 5 mostram os valores necessários para atingir percentagens da potência desejadas.

Tabela 2: Percentagens da potência máxima

	período	ps	percentagem
35%	65525	0b110	35
40%	65525	0b100	40
95%	65525	0b100	95



(a) Alteração para obter 35%



(b) Alteração para obter 40%



(c) Alteração para obter 95%

Figura 5: Analisador lógico mostrando a obtenção das potências desejadas

## 6

As figuras 6, 7 e 8 mostram um valor que foi arredondado "para cima", um que foi arredondado "para baixo" e outro que foi armazenado sem arredondamento, mostrando que o código faz corretamente o arredondamento para manter apenas duas casas decimais.

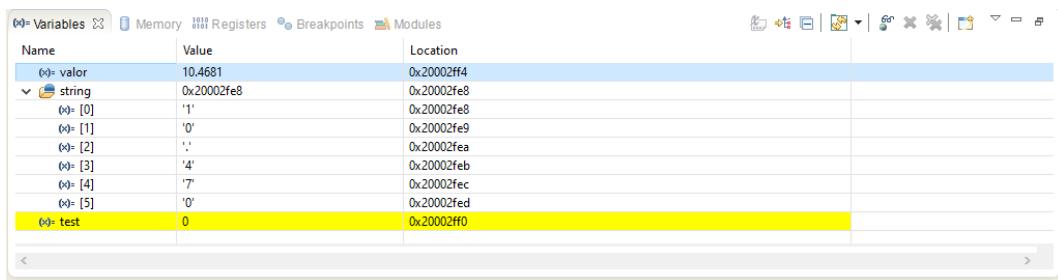


Figura 6: 104681

Name	Value	Location
(& valor	12.461	0x20002ff4
string	0x20002fe8	0x20002fe8
(& [0]	'1'	0x20002fe8
(& [1]	'2'	0x20002fe9
(& [2]	'.'	0x20002fea
(& [3]	'4'	0x20002feb
(& [4]	'6'	0x20002fec
(& [5]	'0'	0x20002fed
(& test	0	0x20002ff0

Figura 7: 12461

Name	Value	Location
(& valor	128.41	0x20002ff4
string	0x20002fe8	0x20002fe8
(& [0]	'1'	0x20002fe8
(& [1]	'2'	0x20002fe9
(& [2]	'8'	0x20002fea
(& [3]	'.'	0x20002feb
(& [4]	'4'	0x20002fec
(& [5]	'1'	0x20002fed
(& test	0	0x20002ff0

Figura 8: 12841

7

A figura 9 mostra os *bitmaps* criados para os módulos *incrementa*, *decrementa* e *desligado*.

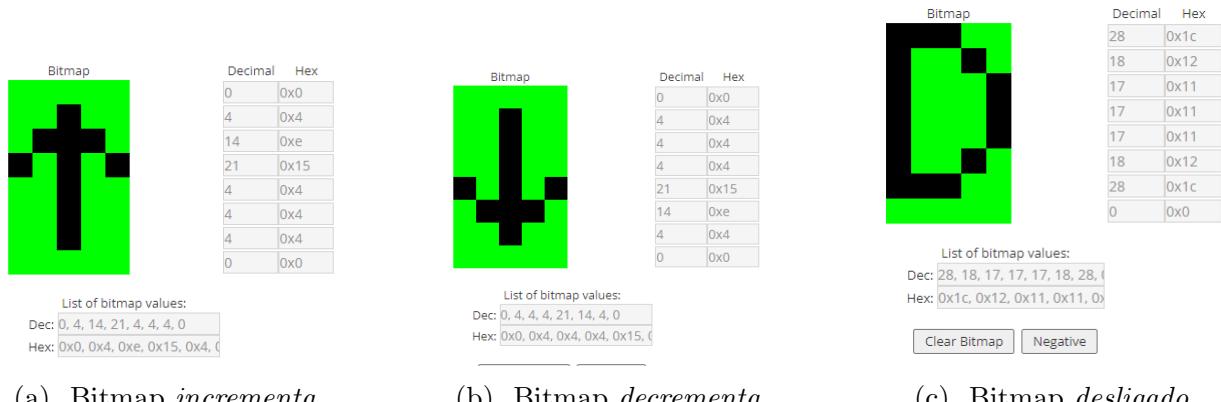


Figura 9: Bitmaps

9

## INÍCIO

main:

  inicialização dos módulos  
  inicialização dos bitmaps

  inicialização das variáveis e estados

loop:

    atualiza percentagem

    Se percentagem atual for diferente da anterior:

        atualiza estado

        atualiza duty cycle

        atualiza LCD

    Se estado atual for diferente do anterior:

        switch:

            caso DESLIGADO

                endereço=0

            caso RESET

                endereço=1

            caso INCREMENTA

                endereço=2

            caso DECREMENTA

                endereço=3

            escreve no LCD

PORTE\_IRQHandler:

    ativa interrupção do sistema

SysTick\_Handler:

    switch:

        caso DESLIGADO

            percentagem=0

        caso RESET

            percentagem=50

        caso INCREMENTA

            percentagem aumenta em 5

        caso DECREMENTA

            percentagem diminui em 5

    desativa interrupção de sistema

FIM

**11**

As tabelas 3 e 4 mostram duas sequências percorridas pela máquina de estados.

## 13

As tabelas 5 e 6 mostram duas sequências percorridas pela máquina de estados. A figura 10 mostra como fica o display LCD do projeto.



Figura 10: Display LCD

## 14

Na figura 11 temos o print size obtido no experimento 10, comparando com a figura 12, referente ao experimento 9, podemos perceber que há um aumento nos três valores: `.txt` corresponde ao tamanho das instruções e dados constantes armazenados na memória flash, `.data` se refere a valores com inicialização armazenada na memória RAM e `.bss` variáveis sem inicialização armazenadas na memória RAM. Os 3 aumentos são esperados, visto que foram adicionadas variáveis globais ao programa e aumentou o número de funções que deve executar. Em detalhe, podemos comentar o grande aumento no valor para `.txt`, devido a nova biblioteca `string.h` adicionada.

```

CDT Build Console [cooler_multi_velocidade]
'Invoking: ARM Ltd Windows GNU Print Size'
"C:/Freescale/CW MCU v10.6/Cross_Tools/arm-none-eabi-gcc-4_7_3/bin/arm-none-eabi-size" --format=berkeley
cooler_multi_velocidade.elf
text    data     bss   dec   hex filename
15256      60    2100   17416   4408 cooler_multi_velocidade.elf
'Finished building: cooler_multi_velocidade.size'
'
../Sources/main.c: In function 'atualiza_lcd_pwm':

```

Figura 11: *Print size* do projeto `cooler_multi_velocidade`

```

CDT Build Console [relogio_digital]
'Executing target #13 relogio_digital.size'
'Invoking: ARM Ltd Windows GNU Print Size'
"C:/Freescale/CW MCU v10.6/Cross_Tools/arm-none-eabi-gcc-4_7_3/bin/arm-none-eabi-size" --format=berkeley relogio_digital.elf
text    data     bss   dec   hex filename
4136      24    2108   6268   187c relogio_digital.elf
'Finished building: relogio_digital.size'
'

```

Figura 12: *Print size* do projeto `relogio_digital`

Tabela 3: Primeira sequência

ESTADO	BOTÃO PRESSIONADO
DESLIGADO	
RESET	NMI
INCREMENTA	IRQA12
INCREMENTA	IRQA12
INCREMENTA	IRQA12
DECREMENTA	IRQA5
DESLIGADO	NMI

Tabela 4: Segunda sequência

ESTADO	BOTÃO PRESSIONADO
DESLIGADO	
RESET	NMI
DECREMENTA	IRQA5
DECREMENTA	IRQA5
DECREMENTA	IRQA5
RESET	IRQA5+IRQA12
INCREMENTA	IRQA12
INCREMENTA	IRQA12
DESLIGADO	NMI

Tabela 5: Primeira sequência

ESTADO	BOTÃO PRESSIONADO	%	ABSOLUTO	RAZÃO
DESLIGADO				
RESET	NMI	50	32763	0,5
INCREMENTA	IRQA12	55	36059	0,55
INCREMENTA	IRQA12	60	39315	0,6
INCREMENTA	IRQA12	65	42591	0,65
DECREMENTA	IRQA5	60	39315	0,6
DESLIGADO	NMI	0	0	0

Tabela 6: Segunda sequência

ESTADO	BOTÃO PRESSIONADO	%	ABSOLUTO	RAZÃO
DESLIGADO				
RESET	NMI	50	32763	0,5
DECREMENTA	IRQA5	45	29486	0,45
DECREMENTA	IRQA5	40	26210	0,4
DECREMENTA	IRQA5	35	22934	0,35
RESET	IRQA5+IRQA12	50	32763	0,5
INCREMENTA	IRQA12	55	36039	0,55
INCREMENTA	IRQA12	60	39315	0,6
DESLIGADO	NMI	0	0	0