



Universidade Estadual de Campinas
Faculdade de Engenharia Elétrica
e de Computação



EA871 - Laboratório de Programação Básica de Sistemas Digitais

Roteiro 12 - Módulo ADC: Voltímetro

Autores: Fernando Teodoro de Cillo e Rafael Silva Cirino

RA: 197029 223730

Campinas
Junho de 2022

Introdução:

O intuito deste experimento é compreender o funcionamento do conversor analógico-digital (ADC) por aproximações sucessivas (SAR). Para isso, desenvolveremos o projeto voltmetro.

Experimento:

1

1.a A inicialização do ADC é por hardware, enquanto PIT é software. Os tempos estão corretos, como visto na figura 1

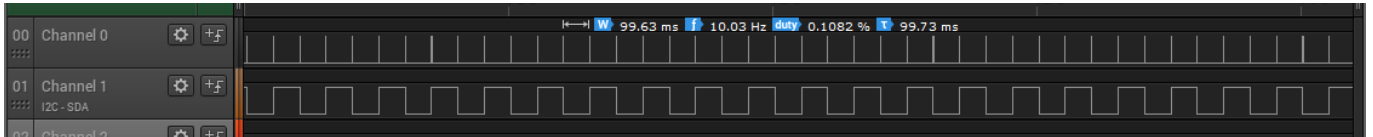


Figura 1: Analisador lógico

1.b A transição dos leds acontece de maneira muito mais lenta devido às alterações nas configurações.

1.c Mesmo reduzindo o delay ele fica com o tempo de amostragem mais longo, como vemos na figura 2, e isso era o que se esperava.

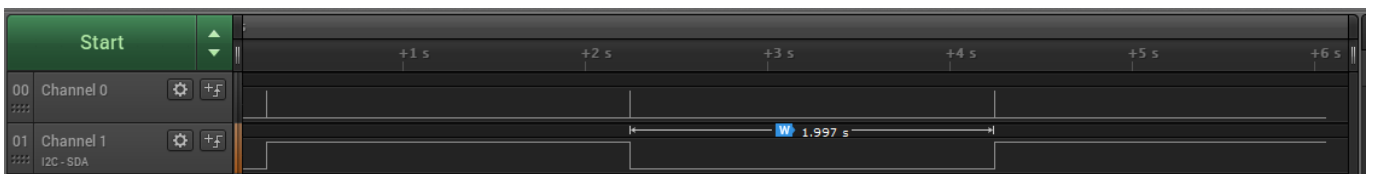


Figura 2: cfg1_adlsmp=1; cfg2_adhsc=1; e sc3_avgs=0b11

4

- adlsmp = 1
- adacken = 0
- adiclk=0b01
- avgs = 0b10
- mode = 0b01
- adlsmp = 1
- adlststs = 0b11
- adhsc = 0
- adiv=0b10
- adiclk=0b01

5

- $SFCadder = 3 \text{ ACDKs} + 5 \text{ bus clocks} = 3 \text{ ACDKs} + 10 \text{ ACDKs} = 13 \text{ ACDKs}$
- $AVGnum = 16$
- $LST = 2 \text{ ADCK}$
- $HSC = 0$
- $BCT = 20 \text{ ADCK}$
- $ADCK = 8$
- frequência ADCK: 1310720Hz

$$ADCK_n = SFCadder + AVGnum \cdot (LST + HSC + BCT) = 13 + 16 \cdot (22) = 365 \cdot ADCK$$

$$\text{Tempo de conversão} = \frac{ADCK_n}{1310720} = 278 \mu s$$

$$\text{Período apropriado} = \left(\frac{8}{1310720} + 278 \right) \cdot 20000 = 5.7s$$

$$\text{Valor de PIT} = 2.0971520 \cdot 3 = 6.3s$$

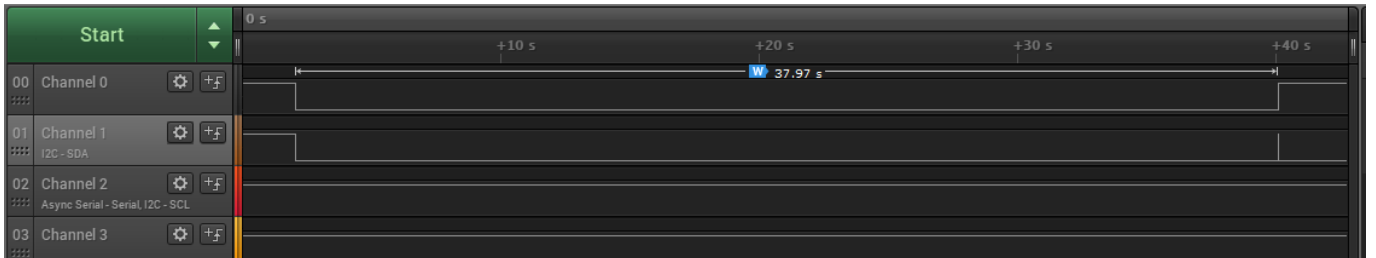


Figura 3: Sinal amostrado pelo analisador lógico

6

O mínimo da representação é 0 e será atrelado ao valor de referência para tensões baixas (0V). O valor máximo da representação é 4095 ($2^{12} - 1$) e será atrelado à tensão de referência para altas (3,27V).

$$VREFH = 3,27V, VREFL = 0V: (0, 4095) \rightarrow (0, 3.27V)$$

Name	Value	Location
valor	509	0x20002fae
mV	0x1ffff000	0x20002fa8
*mV	0.407352	0x1ffff000
prop	0.124328	0x20002fb4

(a) Valor que ainda é interpretado como BAIXO

Name	Value	Location
valor	2060	0x20002fae
mV	0x1ffff000	0x20002fa8
*mV	1.6374	0x1ffff000
prop	0.503175	0x20002fb4

(b) Valor que já é considerado ALTO (próximo de 50% do máximo)

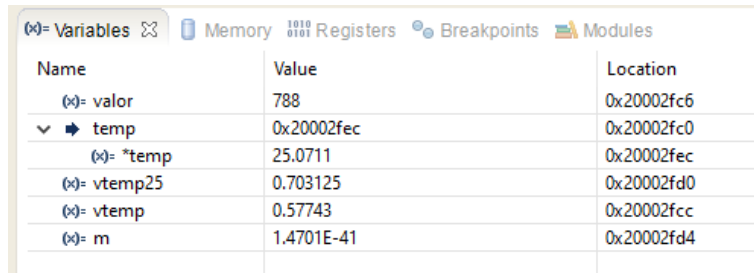
Name	Value	Location
valor	4094	0x20002fae
mV	0x1ffff000	0x20002fa8
*mV	3.27	0x1ffff000
prop	1.0	0x20002fb4

(c) Valor máximo amostrável

Figura 4: Valores amostrados de tensão (em binário) e convertidos para ponto flutuante

7

A função `void AN3031_temperatura (uint16_t valor, float *temperatura)` converte o código binário `valor` amostrado do sensor AN3031 (armazenado em `ADC0_RA`) na grandeza física `temperatura` em ponto flutuante, como mostra a figura 5.



Name	Value	Location
(x)= valor	788	0x20002fc6
▼ temp	0x20002fec	0x20002fc0
(x)= *temp	25.0711	0x20002fec
(x)= vtemp25	0.703125	0x20002fd0
(x)= vtemp	0.57743	0x20002fcc
(x)= m	1.4701E-41	0x20002fd4

Figura 5: Função `AN3031_temperatura`

10

INÍCIO

MAIN:

compara o estado atual com o anterior

se não forem iguais:

laço:

escolha (estado)

caso ESPERA

disparo do temporizador (a cada 0.5s)

desativa a interrupção da botoeira `IRQA5`

muda para o estado `AMOSTRA_TEMP`

caso `AMOSTRA_TEMP`

sensor AN3031 faz a amostragem da temperatura

muda para o estado `TEMPERATURA`

caso `AMOSTRA_VOLT`

amostra a tensão (valor binário)

converte o valor binário para ponto flutuante entre os valores de referência

muda para o estado `TENSAO`

caso `LEITURA`

passa

caso `TEMPERATURA`

lê a última temperatura amostrada

mostra o valor amostrado no LCD

atualiza o estado do canal vermelho do led RGB

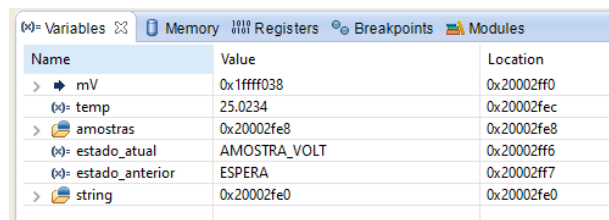
se `temperatura > 24`:

acende o led vermelho
 caso contrário:
 apaga o led vermelho
 ativa a interrupção da botoeira IRQA5
 muda para o estado ESPERA
 caso TENSAO
 lê a tensão amostrada
 mostra o valor amostrado no LCD
 resetar SysTick
 resetar o contador de períodos transcorridos no SysTick
 mudar para o estado LEITURA
 caso LIMPA_LINHA
 apaga o valor da tensão no LCD
 muda para o estado ESPERA
 fimsecolha

FIM

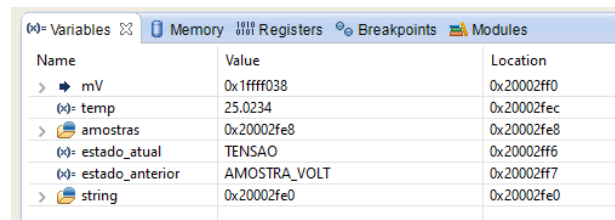
12

As figuras 6 e 7 mostram, respectivamente, os fluxos de execução da nossa máquina de estados para as funções de amostragem de tensão e de temperatura, da forma como fora projetado, seguindo o pseudo-código.



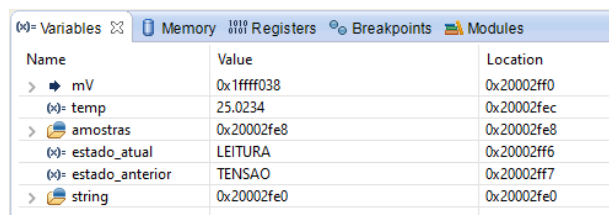
Name	Value	Location
mV	0x1ffff038	0x20002ff0
temp	25.0234	0x20002fec
amostras	0x20002fe8	0x20002fe8
estado_atual	AMOSTRA_VOLT	0x20002ff6
estado_anterior	ESPERA	0x20002ff7
string	0x20002fe0	0x20002fe0

(a) estado AMOSTRA_VOLT



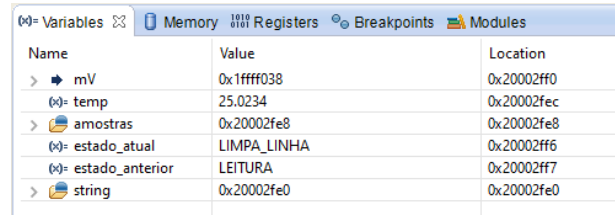
Name	Value	Location
mV	0x1ffff038	0x20002ff0
temp	25.0234	0x20002fec
amostras	0x20002fe8	0x20002fe8
estado_atual	TENSAO	0x20002ff6
estado_anterior	AMOSTRA_VOLT	0x20002ff7
string	0x20002fe0	0x20002fe0

(b) estado TENSAO



Name	Value	Location
mV	0x1ffff038	0x20002ff0
temp	25.0234	0x20002fec
amostras	0x20002fe8	0x20002fe8
estado_atual	LEITURA	0x20002ff6
estado_anterior	TENSAO	0x20002ff7
string	0x20002fe0	0x20002fe0

(c) estado LEITURA



Name	Value	Location
mV	0x1ffff038	0x20002ff0
temp	25.0234	0x20002fec
amostras	0x20002fe8	0x20002fe8
estado_atual	LIMPA_LINHA	0x20002ff6
estado_anterior	LEITURA	0x20002ff7
string	0x20002fe0	0x20002fe0

(d) estado LIMPA_LINHA

Figura 6: Máquina de estados durante a amostragem de tensão

Name	Value	Location
> mV	0x1ffff038	0x20002ff0
(v) temp	2.5393E-41	0x20002fec
> amostras	0x20002fe8	0x20002fe8
(v) estado_atual	AMOSTRA_TEMP	0x20002ff6
(v) estado_anterior	ESPERA	0x20002ff7
> string	0x20002fe0	0x20002fe0

(a) estado AMOSTRA_TEMP

Name	Value	Location
> mV	0x1ffff038	0x20002ff0
(v) temp	2.5393E-41	0x20002fec
> amostras	0x20002fe8	0x20002fe8
(v) estado_atual	TEMPERATURA	0x20002ff6
(v) estado_anterior	AMOSTRA_TEMP	0x20002ff7
> string	0x20002fe0	0x20002fe0

(b) estado TEMPERATURA

Figura 7: Máquina de estados durante a amostragem de temperatura

15

Na figura 9 temos o *print size* obtido no experimento 12, comparando com a figura 8, referente ao experimento 10, 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*.

```

CDT Build Console [calculadora]
'Executing target #12 calculadora.siz'
'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 calculadora.elf
text    data    bss     dec    hex filename
24696   88    2124   26908  691c calculadora.elf
'Finished building: calculadora.siz'

```

Figura 8: *Print size* do projeto calculadora

```

CDT Build Console [voltmetro]
'Executing target #14 voltmetro.siz'
'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 voltmetro.elf
text    data    bss     dec    hex filename
16524   44    2084   18652  48dc voltmetro.elf
'Finished building: voltmetro.siz'

```

Figura 9: *Print size* do projeto voltmetro