SBMI

CRONÓMETRO

Sandro Augusto Costa Magalhães, up201304932
Sérgio António Moreira Fernandes, up201305659

Grupo 9
Turma 6

Universidade do Porto
Faculdade de Engenharia

Dezembro, 2015

1 Introdução

No âmbito da UC de Sistemas Baseados em Microprocessadores, fizemos um cronómetro recorrendo a dois botões de pressão (um para começar a contagem – *start* –, outro para parar a contagem – *stop*) ligados a um microprocessador ATMega328P, programado em linguagem C. Em primeira instância a comunicação deverá ser feita por porta série, de forma a poder mostrar o tempo medido. Neste estado, o nosso projecto é de dificuldade 1.

De forma adicional acrescentou-se ao circuito um *lcd* 16 × 2 e três *displays* de 7 segmentos. Transformamos, assim, o nosso projecto num de dificuldade 3.

2 Material Utilizado

Para que este projecto fosse construído nas melhores condições, foi necessário:

- 1 *BreadBoard*
- 1 Microcontrolador ATMega328P
- Programa Eclipse e *Arduino*
- 1 LCD[1] 16 × 2
- 3 *Displays* de 7 segmentos CPS03631BR
- 2 Botões de pressão
- 1 *Shift Register* 74HC595N

A Microcontrolador

Este é, certamente, o componente mais importante da nossa montagem. Para este projecto foi utilizado o microcomputador *Arduino Uno* V3 que possui um microcontrolador ATMega328P [1].

Este dispositivo permite-nos configurar as suas porta como saídas ou entradas que possuem uma resistência de *pull-up* interno configurável. Além disso, também possui um conjunto de registos que nos permitem configurar os *counters/timers* e as interfaces de comunicação, importantes para o nosso projecto.

O ATMega328P possui três conjuntos de portas I/O (portas B, C e D), das quais 16 foram configuradas, sendo do que entre as quais as PD0 e PD1 funcionam como Rx e o Tx do microcontrolador.

B LCD 16 × 2

O LCD de duas linhas (16 × 2) [2] permite-nos mostrar um conjunto de caracteres em duas linhas, cada uma com uma capacidade máxima de 16 caracteres.

Embora ele permita o uso de vários tipos de montagens, foi utilizada uma montagem em *4-bits*.

C Display 7 segmentos

Também foi utilizado um conjunto de *displays* de 7 segmentos CPS03631BR [3]. Como a sua montagem exigia um elevado número de portas que não possuíamos, tivemos que recorrer a um shift register que nos permitiu um uso apenas de três portas.

[1] LCD – *Liquid Cristals Display* – Mostrador de cristais líquidos

D Shift Register

O *shift register* 74HC595N [4] é, tal como o nome indica, um deslocador de registos de 8 bits no modo de *serial-in parallell-out* com *latch* na saída em três estados. Ele é constituído por 3 portas principais mais as portas de saída: *Input (serial)*, *Store* e *Clock*.

O seu modo de funcionamento consiste em deslocar os bits à medida que estes são escritos pela *serial*, do menos significativo para o mais significativo. Os bits são deslocados quando ocorre uma transição positiva no sinal de *clock*. Eles são escritos na saída do *shift register* quando ocorre uma trasição, também positiva, do sinal de *store*.

3 Metodologia

A Entradas do Microcontrolador

Para inicializar e parar a contagem, usou-se, respectivamente, um botão de *start* e um botão de *stop* (ligados às portas PC0 e PC1), ambos com o ponto de funcionamento a zero. Pretendemos que a corrente no Arduino seja 0,1mA, por isso, através da equação 1, calculamos que $R = 47k\Omega$ na série E12.

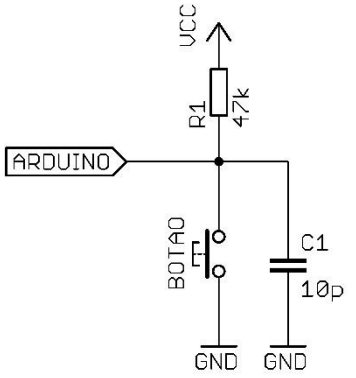

Figura 1 – Esquema utilizado para o cálculo da resistência dos botões

$$R = \frac{V}{I} \quad (1)$$

$$R = 50k\Omega \quad (2)$$

Usamos para esse efeito a resistência de *pull-up* interno do Arduino (montagem dos botões como o anexo 1), que é aproximadamente igual à resistência calculada.

No microcontrolador estas portas foram, portanto, configuradas como entradas, activando o registo de *pull-up* interno [1], como mostra a função *define_ports* do ficheio "main.c" (anexo 2).

B Counter/Timer

Pretendíamos que o cronómetro funcionasse com uma base de tempo de 1 centésima de segundo. No entanto, para obter uma melhor qualidade nos displays de 7 segmentos, usamos um timer com uma base de tempo de 1ms. Portanto, pela equação 3, conseguimos calcular o número de impulsos a contar para obtermos a nossa base de tempo.

$$N = \frac{T \times f_x}{P} \quad (3)$$

Sabendo que apenas existem cinco pré-divisores de frequência e que temos uma frequência de entrada de 16MHz, rapidamente achamos que o *Prescaler* mais adequado é 64 [1]; portanto, temos de contar 250 impulsos. Definindo o *timer0* em modo normal, contamos de 5 a 255 (overflow) – função *init_timers* do "main.c" (anexo 2) .

A incrementação da variável do tempo irá funcionar por *overflow* do timer como mostra a função *ISR(TIMER0_OVF_vect) [5]*.

C Porta série

Primeiramente, para que pudessemos visualizar o tempo medido, utilizamos e configurámos a porta série [1] e imprimimos o tempo por uma função de *printf* reconstruída – *usart_put_str* na biblioteca "usart.h".

Para visualizar os valores impressos pelo microcontrolador recorremos ao programa *Arduino*.

D Saídas do Microcontrolador

LCD 16 × 2

Como primeira especificação adicional, para mostrar o valor da contagem, utilizamos um LCD 16 × 2. O LCD mostra a contagem em segundos com duas casas decimais e consegue contar até, aproximadamente, 11 minutos (660 segundos)

O LCD foi conectado às portas B (PB2 a PB7) do ATMega 328P, configuradas como saídas (anexo 1).

Para utilizar este componente, construiu-se uma biblioteca "lcd.h" baseada no código de Donald Weiman [6]. Para que os tempos de inicialização e os procedimentos do LCD não falhassem verificamos as regras de inicialização em *4-bits* – modo de utilização usado – na datasheet do lcd [2].

```
void lcd_write (uint8_t byte){
DATA_PORT &= ~((1<<DATA7) | (1<<DATA6) |
(1<<DATA5) | (1<<DATA4));
if (byte & 1<<7) DATA_PORT |= (1<<DATA7);
if (byte & 1<<6) DATA_PORT |= (1<<DATA6);
if (byte & 1<<5) DATA_PORT |= (1<<DATA5);
if (byte & 1<<4) DATA_PORT |= (1<<DATA4);
LCD_PORT |= (1<<LCD_E);
_delay_us(1);
LCD_PORT &= ~(1<<LCD_E);
_delay_us(1);
}
```

Código 1 – Código de escrita dos bits no lcd

É interessante verificar o código utilizado na escrita dos bits no LCD. Primeiramente, ele começa por colocar as saídas de dados todas a *LOW* e depois verificasse se cada um dos quatro bits mais significativos é 1 ou 0, caso seja um reescreve-se a saída a um. No final das quatro verificações dá-se a ordem de escrita dos dados (*clock* no *enable*) – cógido 1.

Neste momento passamos a ter um projecto de dificuldade 2.

Display 7 segmentos

Como segunda especificação adicional, montamos três displays de 7 segmentos que mostram a contagem em segundos com 1 casa decimal e apenas conseguem contar até 100 segundos, exclusivé.

Para a montagem destes três *displays* recorremos a um *shift register* que nos permite reduzir o número de portas necessárias para o funcionamento do circuito. Toda esta montagem foi feita segundo o esquema do anexo 1 e ligada à porta B do microprocessador.

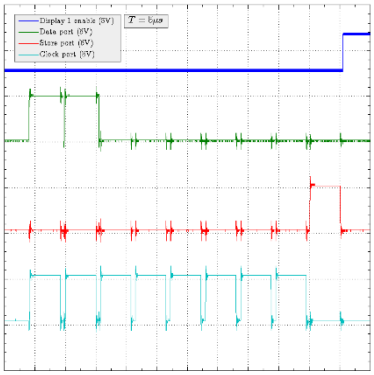

Figura 2 – Escrita do dígito '0' no display 1

Para que este circuito funcionasse, implementou-se uma biblioteca "display7.h" que utiliza, para escrever os *bytes* nos displays uma função semelhante à do código 1, sendo que desta vez escreve um *bit* de cada vez – sendo a ordem do bit menos significativo ao bit mais significativo.

Recorrendo à função de interrupção por *overflow* do *timer 0* [5], definiu-se que cada display está activo isoladamente a uma cadência de 1ms.

4 Análise de Resultados

Para verificar se o nosso cronómetro estava a contar bem o tempo, recorremos a um osciloscópio Rigol [7] e medimos o tempo que decorre desde que carregamos no botão de *start* até carregarmos no botão de *stop*. Como os botões são ativos a 0, esses instantes correspondem à descida do sinal de cada botão, como se pode verificar na figura. Esta medição era muito próxima do resultado obtido com o cronómetro (um erro de aproximadamente 1ms que pode ser causado pelo momento em que ocorre a a interrupção – depois da ordem de escrita do tempo), pelo que concluímos que o cronómetro estava a medir bem o tempo.

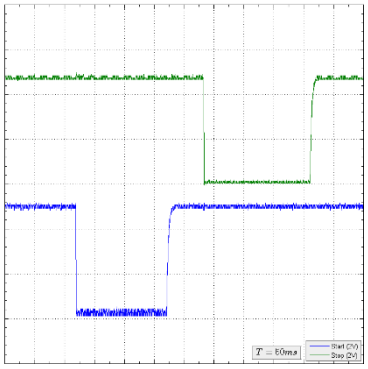

Figura 3 – Verificação da medição do tempo. (azul – botão *start*; verde – botão *stop*)

No exemplo da figura 3 foi medido 214ms no osciloscópio, tendo sido mostrado, quer no lcd de 16×2 quer na porta série 0,21s. Portanto, após um conjunto de testes sucessivos, podemos assumir que apenas podemos ter um erro quando temos múltiplos de 10ms.

Referências

[1] "ATMEL 8-BIT MICROCONTROLLER WITH 4/8/16/32KB IN-SYSTEM PROGRAMMABLE FLASH," Atmel Corporation., Datashet ATmega48A/PA/88A/PA/168A/PA/328/P, Ago. 2014.

[2] "HD44780U (LCD-II)," Hitachi, Ltd., datasheet HD44680, 1998.

[3] "CPS03631AB," Wuxi Compul Electronics Co,.Ltd, datasheet CPS03631BR-11 High Efficiency Red.

[4] "74HC595; 74HCT595," NXP Semiconductor N.V., datasheet 74HC_HCT595, Jan. 26 2015.

[5] (2014, Nov.). *Interrupts*. Available: http://www.nongnu.org/avr-libc/user-manual/group__avr__interrupts.html

[6] D. Weiman. (2015, Nov.). *LCD Programming Example using 'C'*. Available: http://web.alfredstate.edu/weimandn/programming/lcd/ATmega328/LCD_code_gcc_4d.html

[7] "MSO1000Z/DS1000Z Series Digital Oscilloscope," Rigol Technologies, Inc., user's guide DS1074, Dec. 2015.

Anexos

Em anexo a este documento está:

1. Esquema de montagem do projecto
2. Ficheiros do código do projecto implementados em C
3. Imagens amplificadas do osciloscópio