Biblioteca com Funções para as Teclas e Leds

Versão 1.0, 11/04/2020

Funções

|  |  |  |
| --- | --- | --- |
|  |  |  |
| byte | sw\_tira | (byte \*cha) |
| void | sw\_qq\_tecla | (void) |
| void | sw\_fila\_limpa | (void) |
| void | sw\_config | (void) |
| void | sw\_ler | (byte val) |
| void | sw\_busca\_seq1 | (byte val) |
| void | sw\_busca\_seq2 | (byte val) |
| byte | sw\_poe | (byte cha) |
|  |  |  |
| void | void leds\_cont | (byte ct) |
| void | led\_amo | (void) |
| void | led\_AMO | (void) |
| void | led\_Amo | (void) |
| void | led\_vd | (void) |
| void | led\_VD | (void) |
| void | led\_Vd | (void) |
| void | led\_am | (void) |
| void | led\_AM | (void) |
| void | led\_Am | (void) |
| void | led\_vm | (void) |
| void | led\_VM | (void) |
| void | led\_Vm | (void) |
| void | leds\_config | (void) |
|  |  |  |
| void | scp1 | (void) |
| void | SCP1 | (void) |
| void | Scp1 | (void) |
| void | scp2 | (void) |
| void | SCP2 | (void) |
| void | Scp2 | (void) |
| void | scope\_config | (void) |
|  |  |  |

As funções sombreadas são rotinas de apoio. Só usar se souber o que está fazendo.

**Funções para as chaves**

* void **sw\_qq\_tecla** (void)

Espera pelo acionamento de uma tecla qualquer. Retorna quando conseguir retirar algo da fila. Ignora o que for lido.

* void **sw\_fila\_limpa** (void)

Limpar fila do teclado. Joga tudo fora.

* void **sw\_ler** (byte val)

Analisa como foi o passado e, se for o caso, coloca o código na fila do teclado.

* void **sw\_busca\_seq1** (byte val)

Todo código de tecla aceita que é colocado na fila circular é passado para essa função. Ela tenta identificar a Sequência 1 (SEQ1 = ESQ, SUP, DIR). O código de SEQ1 é colocado após o código da última tecla (tecla DIR) que o caracterizou.

* void **sw\_busca\_seq2** (byte val)

Todo código de tecla aceita que é colocado na fila circular é passado para essa função. Ela tenta identificar a Sequência 2 (SEQ2 = ESQ, INF, DIR). O código de SEQ2 é colocado após o código da última tecla (tecla DIR) que o caracterizou.

* void **sw\_config** (void)

Configura o ADC e a fila para as chaves.

* byte **sw\_poe** (byte cha)

Colocar na fila circular o código de uma chave. Retorna

TRUE: se conseguiu colocar o código na fila circular das chaves.

FALSE: se a fila estava cheia e então não conseguiu colocar o código na fila circular.

* byte **sw\_tira** (byte \*cha)

Retira da fila o código da próxima chave. Retorna

TRUE: se conseguir retirar um código da fila e esse código é copiado em \*cha.

FALSE: se a fila estava vazia e então não conseguir retirar um código da fila circular.

**Funções para os leds**

* void **leds\_cont** (byte ct)

Apresenta o argumento em binário nos 4 leds: Laranja - Verde - Amarelo - Vermelho

* void **led\_amo** (void) 🡪 AMO = Apagado
* void **led\_AMO** (void) 🡪 AMO = Aceso
* void **led\_Amo** (void) 🡪 AMO = Invertido

AMO (PB7) = Amarelo Original: Apagar / Acender / Inverter

* void **led\_vd** (void) 🡪 VD = Apagado
* void **led\_VD** (void) 🡪 VD = Aceso
* void **led\_Vd** (void) 🡪 VD = Invertido

VD (PD7) = Verde: Apagar / Acender / Inverter

* void **led\_am** (void) 🡪 AM = Apagado
* void **led\_AM** (void) 🡪 AM = Aceso
* void **led\_Am** (void) 🡪 AM = Invertido

AM (PG2) = Amarelo: Apagar / Acender / Inverter

* void **led\_vm** (void) 🡪 VM = Apagado
* void **led\_VM** (void) 🡪 VM = Aceso
* void **led\_Vm** (void) 🡪 VM = Invertido

VM (PG1) = Vermelho: Apagar / Acender / Inverter

* void **leds\_config** (void)

Configurar pinos dos leds

* void **scp1** (void) 🡪 SCP1 = Apagado
* void **SCP1** (void) 🡪 SCP1 = Aceso
* void **Scp1** (void) 🡪 SCP1 = Invertido

Pino 7 = SCP1 (PH4) = Scope 1: Apagar / Acender / Inverter

* void **scp2** (void) 🡪 SCP2 = Apagado
* void **SCP2**(void) 🡪 SCP2 = Aceso
* void **Scp2** (void) 🡪 SCP2 = Invertido

Pino 8 = SCP2 (PH5) = Scope 2: Apagar / Acender / Inverter

* void **scope\_config** (void)

Configurar pinos para Osciloscópio

----------------------- Detalhes extras ----------------------------------

**Leds:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Nome | Scope 2 | Scope 1 | Laranja | Verde | Amarelo | Vermelho |
| Sigla | SCP2 | SCP1 | AMO | VD | AM | VM |
| Pino I/O | PH5 | PH4 | PB7 | PD7 | PG2 | PG1 |
| Pino Arduino | 8 | 7 | 13 | 38 | 39 | 40 |
|  |  |  |  |  |  |  |

O led Laranja (AMO) corresponde ao led amarelo original da placa Arduino

**Chaves:**

Disposição das teclas

|  |  |  |
| --- | --- | --- |
|  | SW\_CIMA |  |
| SW\_ESQ | SW\_SEL | SW\_DIR |
|  | SW\_BAIXO |  |

Temos duas sequências especiais:

SW\_SEQ1: SW\_ESQ, SW\_CIMA, SW\_DIR

SW\_SEQ2: SW\_ESQ, SW\_BAIXO, SW\_DIR

Cada chave (tecla) tem um código que a identifica

// Codigos para as chaves

#define SW\_NADA 7

#define SW\_INF 6

#define SW\_DIR 4

#define SW\_SUP 3

#define SW\_ESQ 2

#define SW\_SEL 0

#define SW\_SEQ1 8

#define SW\_SEQ2 9

#define SW\_NAOSEI 10

Está disponível um vetor de ponteiros que permite acesso aos nomes de 3 teclas para as teclas

// Teclas - Nome das teclas com apenas 3 letras

// 0 1 2 3 4 5 6 7 8 9 10

char \*sw\_nome[]={"SEL","?1?","ESQ","SUP","DIR","?5?","INF","NAD","SQ1","SQ2","???"};

Variáveis globais usadas pelo teclado

volatile char sw\_fila[SER\_TX\_FILA\_TAM]; //Espaço para a fila teclado

volatile byte sw\_pin, sw\_pout; //Ponteiros para usar a fila

volatile byte sw\_1,sw\_2,sw\_n,sw\_v; //Variáveis para detectar teclas acionadas

volatile byte sw\_st\_seq1,sw\_st\_seq2; //Maq Estados para buscar sequências SEQ1 e SEQ2

**Lógica para funcionamento do teclado:**

Os resistores do divisor resistivo foram pensados de forma a possibilitar a identificação de cada tecla com apenas os 3 bits mais à esquerda do ADC. Como são 3 bits, temos 8 possibilidades. Dois códigos são especiais:

* 111x xxxx 🡪 nenhuma tecla acionada e
* 000x xxxx 🡪 tecla SEL (faz um curto para a terra).

Sobram 6 códigos, dos quais usamos apenas 4. Dois códigos estão sobrando (ausentes).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ADC-Binário** | **Tecla** |  | **ADC-Binário** | **Tecla** |
| 111x xxxx | NADA |  | 011x xxxx | SUP |
| 110x xxxx | INF |  | 010x xxxx | ESQ |
| 101x xxxx | Ausente |  | 001x xxxx | Ausente |
| 100x xxxx | DIR |  | 000x xxxx | SEL |
|  |  |  |  |  |

O Timer 1 é o responsável por ler o ADC. A cada 2 leituras, calcula a média e chama a função **sw\_ler(sw\_val)**, ver fluxograma logo adiante. Esta função conhece o passado do teclado e, se for o caso, valida a nova tecla colocando o seu código na fila circular. Uma nova tecla só é aceita se o teclado passou anteriormente pelo estado de nenhuma tecla acionada.

O Timer 1 está programado para interromper a cada 10 ms. Porém, ele não faz a leitura do ADC toda vez. Ele tem outras funções, de acordo com a tabela abaixo. Ver mais detalhes no texto sobre timers. A consulta do ADC para o teclado é feita, aproximadamente, em 37,5 Hz (100\*(10/32)), ou seja, uma leitura a cada 26,6 ms.

Funcionou bem: timer1\_cont = 0, 1, ..., 31, 0, 1, ..., interrupção em 100 Hz

|  |  |  |  |
| --- | --- | --- | --- |
| 0-ADC Start | 8-+Ler, ADC start\* | 16-ADC Start | 24-+Ler, ADC start\* |
| 1-+Ler, ADC start | 9-+Ler, ADC start | 17-+Ler, ADC start | 25-+Ler, ADC start |
| 2-+Ler, ADC start\* | 10-+Ler, ADC start\* | 18-+Ler, ADC start\* | 26-+Ler, ADC start\* |
| 3-+Ler, ADC start | 11-+Ler, ADC start | 19-+Ler, ADC start | 27-+Ler, ADC start |
| 4-+Ler, ADC start\* | 12-Ler, Canal1(VCAR)\* | 20-+Ler, ADC start\* | 28-Ler, Canal 2(VCAP)\* |
| 5-+Ler, ADC start | 13-ADC Start | 21-+Ler, ADC start | 29-ADC Start |
| 6-+Ler, ADC start\* | 14- Ler, ADC start | 22-+Ler, ADC start\* | 30-Ler, ADC start |
| 7-+Ler, ADC start | 15-Ler, Canal 0 | 23-+Ler, ADC start | 31-Ler, Canal 0 |

\* indica a fase para tirar a média na leitura do teclado.

Lógica para detectar teclas acionadas

sw\_1 e sw\_2 🡪 garantir duas leituras iguais, evita transitórios;

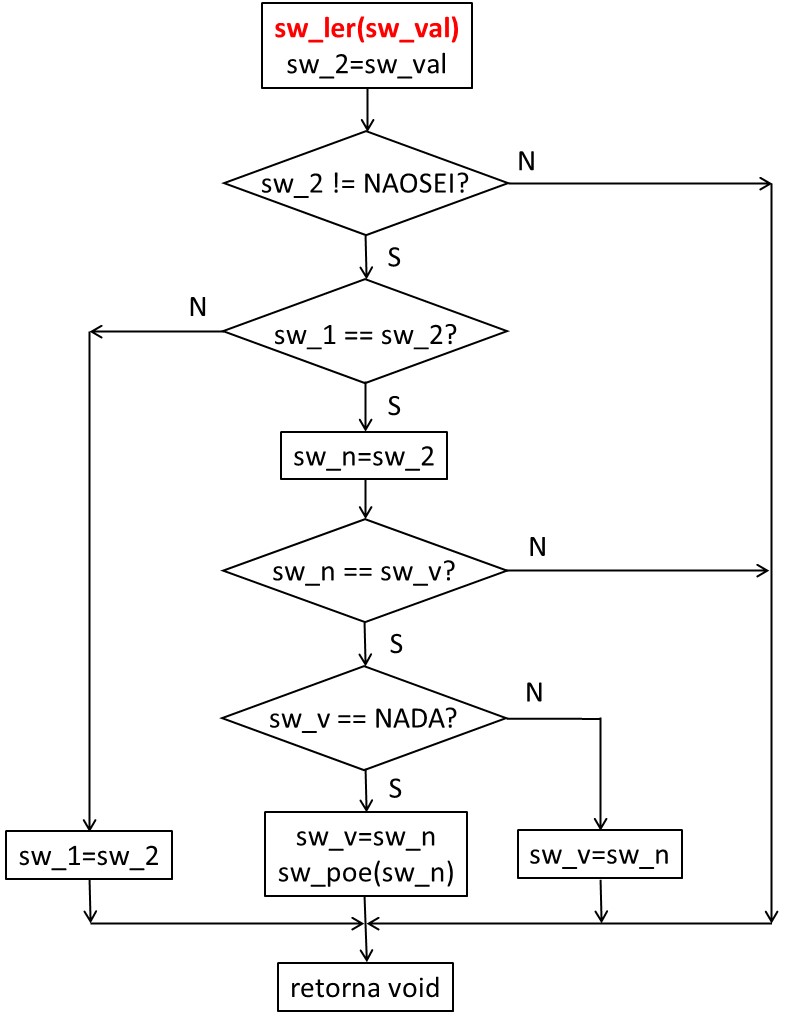
sw\_n 🡪 nova tecla

sw\_v 🡪 tecla velha

Tecla é considerada válida quando duas leituras seguidas são iguais: sw\_1=sw\_2

Neste caso sw\_n recebe o código da tecla.

Se sw\_n<>sw\_v e sw\_v = NADA, nova tecla é colocada no buffer e depois sw\_v=sw\_n.



*Gabaritos para configurar ADC (8 bits alinhado pela esquerda, ler apenas ADCH)*

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| **ADMUX** | REFS1 | REFS0 | ADLAR | MUX4 | MUX3 | MUX2 | MUX1 | MUX0 |
| 0,1,1 | 1,1,1 | 1 | 0 | 0 | 0 | 0,0,1 | 0,1,0 |
| **ADCSRA** | ADEN | ADSC | ADATE | ADIF | ADIE | ADPS2 | ADPS1 | ADPS0 |
| 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| **ADCSRB** | - | ACME | - | - | MUX5 | ADST2 | ADST1 | ADST0 |
| - | 0 | - | - | 0 | 1 | 0 | 1 |

Configuração para os diversos canais

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Analógico | Canal ADC | MUX5:0 | REFS1:0 | REF volts |
| Teclado | 0 | 0 0 0 0 0 | 0 1 | AVCC |
| Tensão VCAR | 1 | 0 0 0 0 1 | 1 1 | 2,56 V |
| Tensão VCAP | 2 | 0 0 0 1 0 | 1 1 | 2,56 V |

Teclado: ADMUX = (1<<REFS0) | (1<<ADLAR); //Ref = AVCC

VCAR: ADMUX = (1<<REFS1) | (1<<REFS0) | (1<<ADLAR) | (1<<MUX0); //Canal 1, Ref = 2,56

VCAP: ADMUX = (1<<REFS1) | (1<<REFS0) | (1<<ADLAR) | (1<<MUX1); //Canal 2, Ref = 2,56

Cálculo dos resistores do divisor resistivo do teclado

|  |  |
| --- | --- |
| (a) Circuito | (b) Numérica |

Vi = tensão no adc ao acionar a chave SWi.

🡪

🡪

🡪

|  |  |  |
| --- | --- | --- |
|  | Calculados | Comerciais |
| R1 | 10.000,00 | 10.000 |
| R2 | 2.307,69 | 2.400 |
| R3 | 2.237,76 | 2.200 |
| R4 | 3.232,32 | 3.300 |
| R5 | 5.079,37 | 5.100 |
| R6 | 9.142,86 | 9.100 |
| R7 | 21.333,33 | 22.000 |
|  |  |  |

Uma imagem contendo captura de tela

Descrição gerada automaticamente

% Valores comerciais

R7 = 22000; %21333.33

R6 = 9100; % 9142.86

R5 = 5100; % 5079.37

R4 = 3300; % 3232.32

R3 = 2200; % 2237.76

R2 = 2400; % 2307.69

R1 = 10000; %10000

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  | R | V | Hexa |  | R | V | Hexa |  | 3 Bits |
| 1 | 10.000,00 | 0,0 | 0 |  | 10.000 | 0,0 | 00 |  | 000 |
| 2 | 2.307,69 | 0,9375 | 30 |  | 2.400 | 0,9677 | 32 |  | 001 |
| 3 | 2.237,76 | 1,5625 | 50 |  | 2.200 | 1,5753 | 51 |  | 010 |
| 4 | 3.232,32 | 2,1875 | 70 |  | 3.300 | 2,2067 | 71 |  | 011 |
| 5 | 5.079,37 | 2,8125 | 90 |  | 5.100 | 2,8261 | 91 |  | 100 |
| 6 | 9.142,86 | 3,4375 | B0 |  | 9.100 | 3,4424 | B0 |  | 101 |
| 7 | 21.333,33 | 4,0625 | D0 |  | 22.000 | 4,0758 | D1 |  | 110 |
|  |  |  |  |  |  |  |  |  |  |