|  |  |
| --- | --- |
|  | **UNIVERSIDADE ESTADUAL DE CAMPINAS**  Faculdade de Engenharia Elétrica e de Computação  EA076 - Laboratório de Sistemas Embarcados |

**Projeto 1:**

**Semáforo**

|  |  |
| --- | --- |
| **Nomes** | **RA** |
| Caio de Andrade Avelino | 139342 |
| Vinicius Jokubauskas | 139825 |
|  |  |
|  |  |

1°sem/2018

MAIN.C

#include "Cpu.h"

#include "Events.h"

#include "LED\_VERMELHO.h" // Incluindo a biblioteca do dispositivo

#include "BitIoLdd1.h"

#include "TIMER2.h" // Incluindo a biblioteca do dispositivo

#include "TimerIntLdd1.h"

#include "LED\_VERDE\_PEDESTRE.h" // Incluindo a biblioteca do dispositivo

#include "BitIoLdd5.h"

#include "LED\_VERMELHO\_PEDESTRE.h" // Incluindo a biblioteca do dispositivo

#include "BitIoLdd4.h"

#include "BOTON.h" // Incluindo a biblioteca do dispositivo

#include "ExtIntLdd1.h"

#include "LDRConverter.h"

#include "AdcLdd1.h"

#include "TIMER2.h" // Incluindo a biblioteca do dispositivo

#include "TU1.h"

#include "LED\_AMARELO.h" // Incluindo a biblioteca do dispositivo

#include "BitIoLdd2.h"

#include "LED\_VERDE.h" // Incluindo a biblioteca do dispositivo

#include "BitIoLdd3.h"

#include "PE\_Types.h"

#include "PE\_Error.h"

#include "PE\_Const.h"

#include "IO\_Map.h"

#include "LDRConverter.h" // Incluindo a biblioteca do dispositivo

/\*Funcoeses do semaforo de carros, por exemplo no vermelho acendemos o LED\_VERMELHOR com a funcao PutVal(1) e apagamos os outros

LEDs utilizando a funcao PutVal(0)\*/

#define SEMAFORO\_VERDE LED\_VERMELHO\_PutVal(0); LED\_AMARELO\_PutVal(0); LED\_VERDE\_PutVal(1);

#define SEMAFORO\_AMARELO LED\_VERMELHO\_PutVal(0); LED\_AMARELO\_PutVal(1); LED\_VERDE\_PutVal(0);

#define SEMAFORO\_VERMELHO LED\_VERMELHO\_PutVal(1); LED\_AMARELO\_PutVal(0); LED\_VERDE\_PutVal(0);

/\*Definicao das funcoes das luzes do semaforo de pedestres. Como exemplo podemos utilizar a SEMAFORO\_VERDE\_PEDESTRE, que seta a luz

do LED\_VERMELHO como desligada PutVal(0) e do LED\_VERDE como ligado PutVal(1)\*/

#define SEMAFORO\_VERDE\_PEDESTRE LED\_VERMELHO\_PEDESTRE\_PutVal(0); LED\_VERDE\_PEDESTRE\_PutVal(1);

#define SEMAFORO\_VERMELHO\_PEDESTRE LED\_VERMELHO\_PEDESTRE\_PutVal(1); LED\_VERDE\_PEDESTRE\_PutVal(0);

#define BOTON\_PRESIONADO 0

#define BOTON\_LIVRE 1

int estado = 1 // Variavel de estado

int flag\_delay2 = 0; //Flag que utilizamos para a transicao do estado amarelo para o vermelho

int flag\_delay3= 0; //Flag para o delay de transicao do estado vermelho (3) para o verde(1)

int flag\_delayPisca= 0; //Flag para checarmos se a luz vermelha do semaforo de pedestres deve piscar ou nao

int flag\_Pisca = 0; //Flag que utilizamos para piscar o LED\_VERMELHO\_PEDESTRE

static uint16\_t LDRMeasure; //variavel que gravamos a medida do sensor LDR

int thresholdNoite = 39985; //Threshold para definir qual o nivel de luz é considerado Noite

int LDRMeasureInt = 0; //Variavel que tambem utilizamos para salvar a medida do sensor LDR, achamos mais facil manipular uma variavel int

int isNightFlag = 0; //Flag que define se o sensor ficou tempo suficiente no escuro para ser considerado noite

int countNight = 0; // Countador de ciclos nos quais o LDR esta no estado de noite

int countDay = 0; // Contador de ciclos nos quais o LDR esta no estado de dia

void main(void)

{

PE\_low\_level\_init();

for(;;){

LDRConverter\_Measure(TRUE); //ativamos a medida do sensor LDR

LDRConverter\_GetValue16(&LDRMeasure); //gravamos essa medida na variavel static 16bits

LDRMeasureInt = LDRMeasure; //copiamos a medida em uma variavel int

if (isNightFlag==1){

/\*Teste logico para sabermos se o semaforo ira apenas piscar as luzes Vermelha dos carros e Amarela de Pedestres

ou se sera considerado dia e entraremos na nossa maquina de estados para a operaÃ§Ã£o normal do semaforo\*/

LED\_VERMELHO\_PutVal(0);

LED\_AMARELO\_PutVal(0);

LED\_VERDE\_PutVal(0);

LED\_VERMELHO\_PEDESTRE\_PutVal(0);

LED\_VERDE\_PEDESTRE\_PutVal(0);

//Utilizamos esse teste para negarmos o valor dos LEDs a cada dois ciclos de interrupcao, ou seja, a cada 1 segundo

if (countNight%2 == 0){

LED\_AMARELO\_NegVal();

LED\_VERMELHO\_PEDESTRE\_NegVal();

}

}

else if (isNightFlag ==0){ //Se nao for noite, entramos na aquina de estados

switch (estado){ //switch para selecionarmos o que ser feito de acordo com cada estado (1, 2, 3)

case 1:

/\*No estado 1 tem o SEMAFORO\_VERDE ativado e o SEMAFORO\_VERMELHO\_PEDESTRE ativado

todas a outras luzes estao desligadas\*/

SEMAFORO\_VERDE;

SEMAFORO\_VERMELHO\_PEDESTRE;

if (BOTON\_GetVal()==0){ //Se o botao é apertado, vamos para o estado 2

estado = 2;

}

else{ //Se o botao nao é ativado, continuamos no estado 1 atÃ© que o botao seja ativado ou se torne noite

estado = 1;

}

break;

case 2:

SEMAFORO\_AMARELO; //Entrada no caso 2, no qual a luz amarela do semaforo dos carros é ligada

if(flag\_delay2 == 0){ f

/\*

flag\_delay é a varÃaavel que utilizamos para sinalizar se o o tempo que a luz amarela tem que ficar ligada passou ou nao \*/

estado =2;

}

else { //flag\_delay2 é setada em um quando o tempo de delay da luz amarela acaba, quando acaba vamos para o estado 3

estado =3;

flag\_delay2 = 0;

}

break;

case 3 :

SEMAFORO\_VERMELHO; //neste estado a luz vermelha dos carros é ligada

if(flag\_delay3 == 0 && flag\_delayPisca == 0) {

/\* Enquato o delay da luz vermelha nao acaba e o flag\_delayPisca, que é setado em um quando o semaforo dos carros esta prestes

a abrir forem zero, temos a luz verde do semaforo de pedestres ligada\*/

SEMAFORO\_VERDE\_PEDESTRE;

estado =3;

}

else if(flag\_delay3 == 0 && flag\_delayPisca == 1){

/\*quando passamos um certo tempo com o semaforo vermelho de carros e a luz verde estÃ¡ prestes a acender

desligamos a luz verde do semaforo de pedestres e comecamos piscar a luz vermelha do semaforo dos pedestres \*/

estado=3;

SEMAFORO\_VERMELHO\_PEDESTRE;

//as condicoes abaixo fazem a luz vermelha dos pedestres piscar a cada 1 segundo

if (flag\_Pisca == 1){

LED\_VERMELHO\_PEDESTRE\_PutVal(1);

}

else {

LED\_VERMELHO\_PEDESTRE\_PutVal(0);

}

}

else {

/\*condicao final, na qual o delay do semaforo vermelho de carros passou e a luz verde de carros é ligada

e a vermelha de pedestres também. Voltamos ao estado 1 \*/

estado = 1;

flag\_delay3=0;

flag\_delayPisca= 0;

flag\_Pisca=0;

}

break;

}

}

}

#ifdef PEX\_RTOS\_START

PEX\_RTOS\_START();

#endif

for(;;){}

/\*\*\* RTOS startup code. Macro PEX\_RTOS\_START is defined by the RTOS component. DON'T MODIFY THIS CODE!!! \*\*\*/

#ifdef PEX\_RTOS\_START

PEX\_RTOS\_START(); /\* Startup of the selected RTOS. Macro is defined by the RTOS component. \*/

#endif

/\*\*\* End of RTOS startup code. \*\*\*/

/\*\*\* Processor Expert end of main routine. DON'T MODIFY THIS CODE!!! \*\*\*/

for(;;){}

/\*\*\* Processor Expert end of main routine. DON'T WRITE CODE BELOW!!! \*\*\*/

} /\*\*\* End of main routine. DO NOT MODIFY THIS TEXT!!! \*\*\*/

EVENTS.C

/\* ###################################################################

\*\* Filename : Events.c

\*\* Project : Semaforo

\*\* Processor : MKL25Z128VLK4

\*\* Component : Events

\*\* Version : Driver 01.00

\*\* Compiler : GNU C Compiler

\*\* Date/Time : 2018-03-07, 20:21, # CodeGen: 0

\*\* Abstract :

\*\* This is user's event module.

\*\* Put your event handler code here.

\*\* Settings :

\*\* Contents :

\*\* Cpu\_OnNMIINT - void Cpu\_OnNMIINT(void);

\*\*

\*\* ###################################################################\*/

/\*!

\*\* @file Events.c

\*\* @version 01.00

\*\* @brief

\*\* This is user's event module.

\*\* Put your event handler code here.

\*/

/\*!

\*\* @addtogroup Events\_module Events module documentation

\*\* @{

\*/

/\* MODULE Events \*/

#include "Cpu.h"

#include "Events.h"

/\* Importando as variaveis do programa principal Main.c

Essas variaveis vao ser a principal comunicacao entre as interrupcoes e o programa principal \*/

extern int estado;

extern int flag\_delay2;

static int t = 0;

extern int flag\_delay3;

extern int flag\_delayPisca;

extern int flag\_Pisca;

extern int isNightFlag;

extern int thresholdNoite;

extern int countNight;

extern int countDay;

extern int LDRMeasureInt;

/\* User includes (#include below this line is not maintained by Processor Expert) \*/

void TIMER2\_OnInterrupt(void) //Interrupcao do timer, ativada a cada 500ms

{

if (estado==2){

/\*entramos nessa condicao quando estamos no estado 2 (sinal amarelo carros)

setamos, aqui, o delay do sinal amarelo \*/

t++;

if (t>=10) { //quando a contagem atinge, setamos a flag para ir para o prÃ³ximo estado

flag\_delay2 = 1;

t = 0;

}

}

else if (estado==3){

/\* esta condiÃ§Ã£o Ã© utilizada para contarmos o delay do estado 3 (sinal vermelho \*/

t++;

if (t>=17 && t<25){

/\* quando faltam 8 ciclos para o fim do delay, setamos a flag\_delayPisca,

para comercarmos a piscar o led vermelho do sinal de pedestres \*/

flag\_delayPisca = 1;

/\* utilizamos essa condicao abaixo para piscarmos os LED vermelho dos pedestres,

utilizamos um outro metodo para piscar as luzes na noite, porem mantivemos os

dois modos no nosso codigo\*/

if (t%3 == 0){

flag\_Pisca = 1;

}

else {

flag\_Pisca = 0;

}

}

if (t>=25) { //apos o fim da contagem setamos as flags para podermos ir para o estado 1 novamente

flag\_delay3 = 1;

flag\_delayPisca = 0;

t = 0;

}

}

if (LDRMeasureInt>thresholdNoite){

/\* aqui contamos o numero de ciclos em que o LDR detectou baixa luminosidade\*/

countNight++;

if (countNight>=8){

/\*depois de 8 ciclos em baixa luminosidade, ou seja, 4 segundos setamos

nossa isNightFlag = 1 para podermos entrar na condiÃ§Ã£o de noite, onde

a luz amarela de carros e a vermelha de pedestres piscam \*/

isNightFlag =1;

countDay = 0; //zeramos o contador do dia

}

}

else{

countDay++; //aqui contamos o numero de ciclos na qual o LDR esteve com uma alta luminosidade

countNight++;

if (countDay>=8){ // entramos novamente na condiÃ§Ã£o de dia

isNightFlag =0; //voltamos a flag para o estado de dia

countNight = 0; //zeramos o contador da noite

}

}

}

void Cpu\_OnNMIINT(void)

{

/\* Write your code here ... \*/

}

/\*

\*\* ===================================================================

\*\* Event : BOTON\_OnInterrupt (module Events)

\*\*

\*\* Component : BOTON [ExtInt]

\*\* Description :

\*\* This event is called when an active signal edge/level has

\*\* occurred.

\*\* Parameters : None

\*\* Returns : Nothing

\*\* ===================================================================

\*/

void BOTON\_OnInterrupt(void)

{

}

}

/\*

\*\* ===================================================================

\*\* Event : LDRConverter\_OnEnd (module Events)

\*\*

\*\* Component : LDRConverter [ADC]

\*\* Description :

\*\* This event is called after the measurement (which consists

\*\* of <1 or more conversions>) is/are finished.

\*\* The event is available only when the <Interrupt

\*\* service/event> property is enabled.

\*\* Parameters : None

\*\* Returns : Nothing

\*\* ===================================================================

\*/

void LDRConverter\_OnEnd(void)

{

/\* Write your code here ... \*/

}

/\*

\*\* ===================================================================

\*\* Event : LDRConverter\_OnCalibrationEnd (module Events)

\*\*

\*\* Component : LDRConverter [ADC]

\*\* Description :

\*\* This event is called when the calibration has been finished.

\*\* User should check if the calibration pass or fail by

\*\* Calibration status method./nThis event is enabled only if

\*\* the <Interrupt service/event> property is enabled.

\*\* Parameters : None

\*\* Returns : Nothing

\*\* ===================================================================

\*/

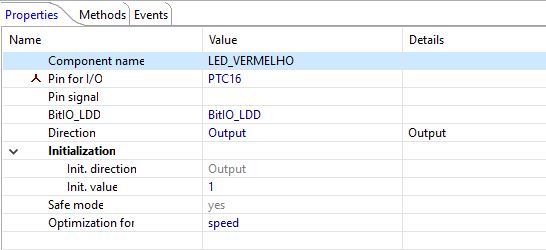
void LDRConverter\_OnCalibrationEnd(void)

{

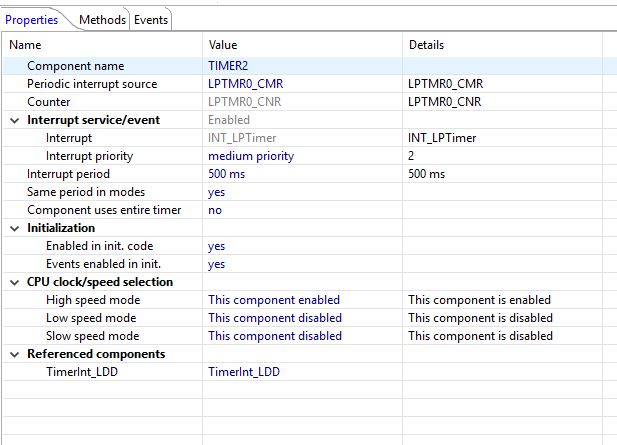
/\* Write your code here ... \*/

}

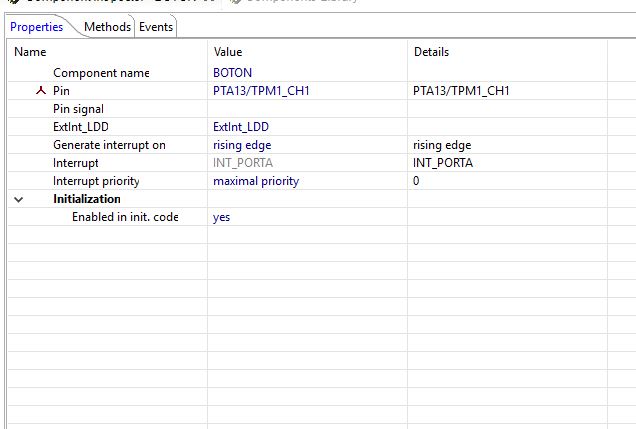
CONFIGURAÇÃO LEDs



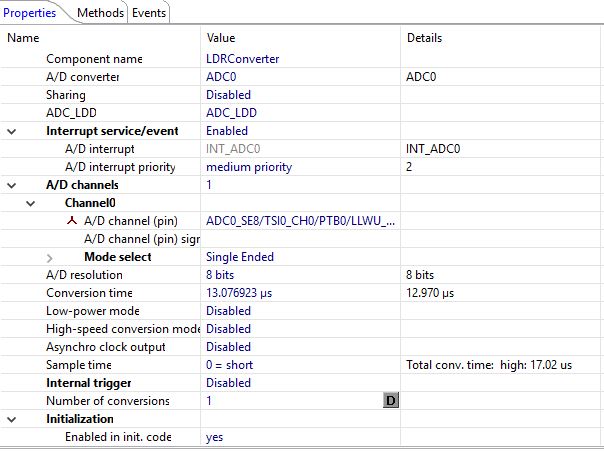
CONFIGURAÇÃO TIMER



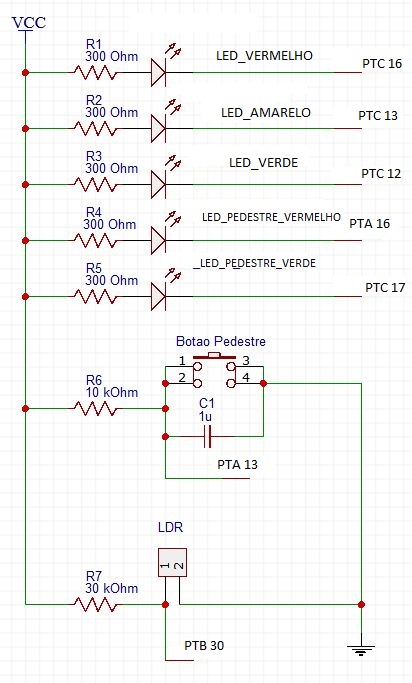
CONFIGURAÇÃO BOTÃO



CONFIGURAÇÃO CONVERSOR AD



CIRCUITO DE REFERÊNCIA



VIDEO

<https://www.youtube.com/watch?v=Sj7q8ErhJvM&feature=youtu.be>