**Nome: Gabriel Lujan Bonassi**

**Nº USP: 11256816**

1. Fazer uma aquisição analógica;
2. Acender o LED azul quando o valor for próximo de 3.3 V e o LED verde quando o valor for próximo de 0 V.

Para estes dois itens, como é igual ao exercício 8, podemos reutilizar o código daquele exercício, fazendo as adaptações necessárias para utilizar as definições dos registradores dadas pelo codewarrior (através do #include “derivative.h”, que por sua vez da um #include <MKL25Z4.h>)

Código:

#include "derivative.h" /\* include peripheral declarations \*/

void ADC0\_init(void);

void LED\_set(int s);

void LED\_init(void);

int main(void)

{

     short int result;

     LED\_init();

     ADC0\_init();

     while (1)

     {

         ADC0\_SC1A = 0x10; // inicia a conversao, single-ended, AD8 selecionado como input

         while (!(ADC0\_SC1A & 0x80)){} //aguarda a conversao acabar (faco um AND entre a flag COCO e 1, quando os 2 forem 1, retorna 1 e para o while)

         result = ADC0\_RA; // le o resultado da conversao na var result

         LED\_set(result >> 7); // seta o led com base no bit 7 do result

     }

}

void ADC0\_init(void) {

    SIM\_SCGC5 |= (1<<10); // enable clock PORTB (pg. 206)

    SIM\_SCGC6 |= 0x8000000; // enable clock ADC0 (pg. 207)

    PORTE\_PCR0 = 0; // enable PTB0 pin out

    ADC0\_SC2 &= ~0x40; // software trigger

    ADC0\_CFG1 = 0x54;

}

void LED\_init(void) {

    SIM\_SCGC5 |= 0x1000; // enable clock PORTD

    SIM\_SCGC5 |= 0x400; // enable clock PORTB

    // posso dar enable nos dois clocks ao mesmo tempo? sim!

    PORTD\_PCR1 = 0x100; // enable PTD1 as GPIO (pg. 183) (Blue LED)

    PORTB\_PCR19 = 0x100; // enable PTB19 as GPIO (Green LED)

    PORTB\_PCR18 = 0x100; // enable PTB18 as GPIO (Red LED)

    GPIOB\_PDDR |= 0x80000; // make PTB19 (Green LED) as output (pg. 778) (bit relativo ao numero da porta)

    GPIOB\_PDDR |= 0x40000; // make PTB18 as output (Red LED)

    GPIOB\_PDDR |= 0x02; // make PTD1 as output (Blue LED)

}

void LED\_set(int s) {

    // Red LED

    if (s & 1) { // usa BIT 0 de s

        GPIOB\_PCOR = 0x40000; // turn on

    } else {

        GPIOB\_PSOR = 0x40000; // turn off

    }

    // Green LED

    if (s & 2) { //usa BIT 1 do s

        GPIOB\_PCOR = 0x80000; // turn on

    } else {

        GPIOB\_PSOR = 0x40000; // turn off

    }

    // Blue LED

    if (s & 4) { //usa bit 2 do s

        GPIOD\_PCOR = 0x02; // turn on

    } else {

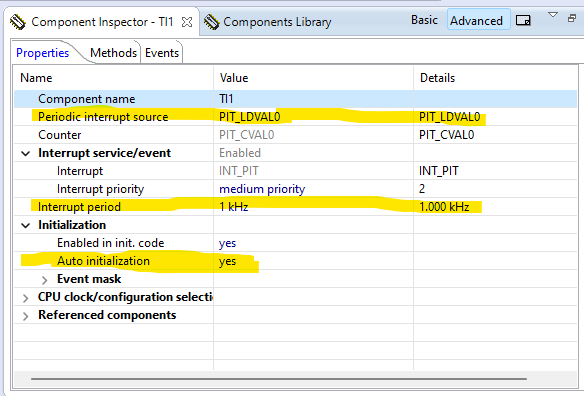
        GPIOD\_PSOR = 0x02; // turn off

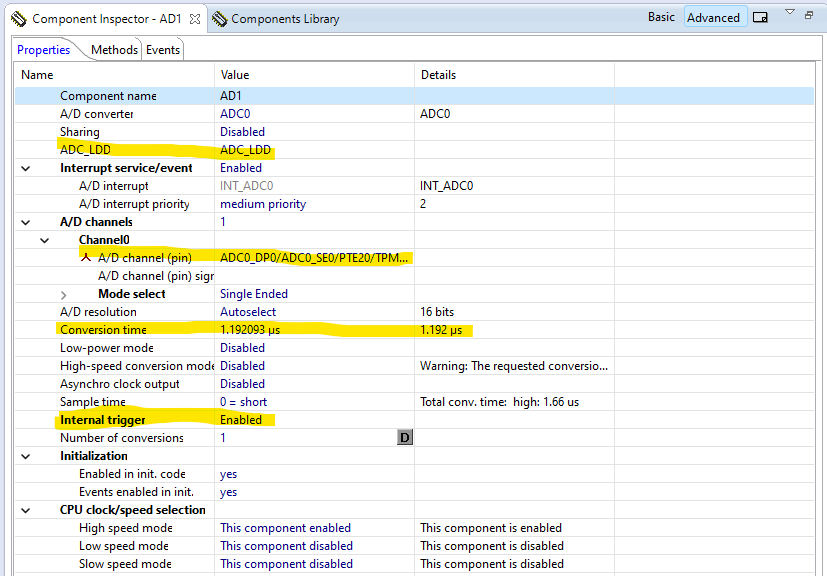
    }

}

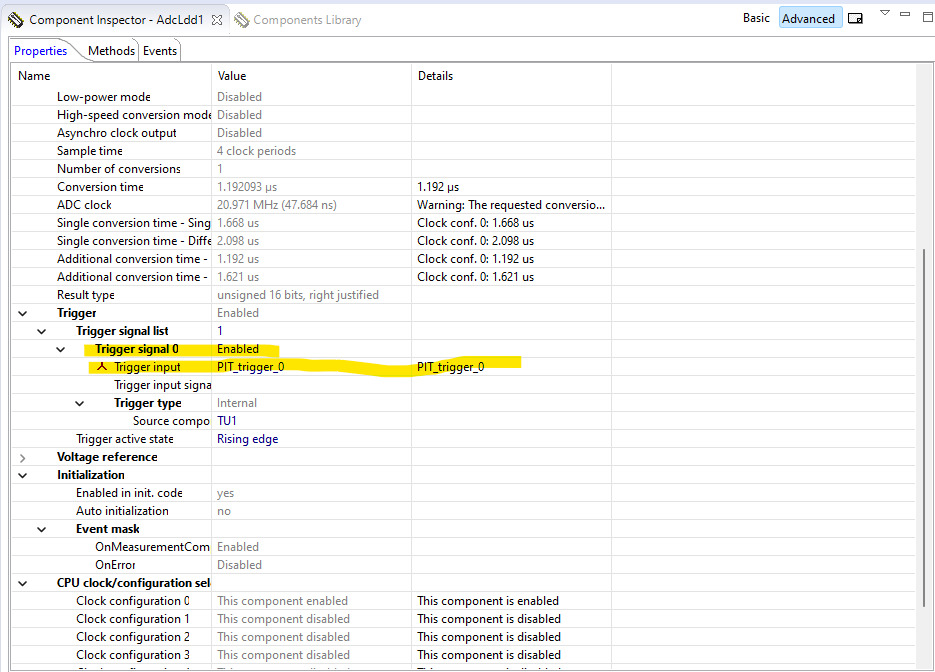
1. Utilizar um timer periódico para, por interrupção, disparar a conversão AD. Usar a interrupção de fim de conversão para acender os LEDs. Permitido o uso do Processor Expert para este item.

Configurando o timer periódico usando o “Component” TimerInt\_LDD:

Configurando o “Component” ADC:



Repare que o “Internal trigger” está em “Enabled”. Vamos configurar o input do trigger do ADC utilizando o Timer que criamos:



Agora, vamos configurar dois arquivos: events.c e main.c

Código do events.c:

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

\*\*     Filename    : Events.h

\*\*     Project     : Ex5

\*\*     Processor   : MKL25Z128VLK4

\*\*     Component   : Events

\*\*     Version     : Driver 01.00

\*\*     Compiler    : GNU C Compiler

\*\*     Date/Time   : 2023-07-07, 14:11, # 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.h

\*\* @version 01.00

\*\* @brief

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

\*\*         Put your event handler code here.

\*/

/\*!

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

\*\*  @{

\*/

#ifndef \_\_Events\_H

#define \_\_Events\_H

/\* MODULE Events \*/

#include "PE\_Types.h"

#include "PE\_Error.h"

#include "PE\_Const.h"

#include "IO\_Map.h"

#include "TI1.h"

#include "TU1.h"

#include "AD1.h"

#include "AdcLdd1.h"

#include "Bit1\_Green\_LED.h"

#include "BitIoLdd1.h"

#include "Bit2\_Blue\_LED.h"

#include "BitIoLdd2.h"

#ifdef \_\_cplusplus

extern "C" {

#endif

/\*

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

\*\*     Event       :  Cpu\_OnNMIINT (module Events)

\*\*

\*\*     Component   :  Cpu [MKL25Z128LK4]

\*/

/\*!

\*\*     @brief

\*\*         This event is called when the Non maskable interrupt had

\*\*         occurred. This event is automatically enabled when the [NMI

\*\*         interrupt] property is set to 'Enabled'.

\*/

/\* ===================================================================\*/

void Cpu\_OnNMIINT(void);

void AD1\_OnEnd(void);

/\*

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

\*\*     Event       :  AD1\_OnEnd (module Events)

\*\*

\*\*     Component   :  AD1 [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 AD1\_OnCalibrationEnd(void);

/\*

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

\*\*     Event       :  AD1\_OnCalibrationEnd (module Events)

\*\*

\*\*     Component   :  AD1 [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

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

\*/

/\*

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

\*\*     Event       :  TI1\_OnInterrupt (module Events)

\*\*

\*\*     Component   :  TI1 [TimerInt\_LDD]

\*/

/\*!

\*\*     @brief

\*\*         Called if periodic event occur. Component and OnInterrupt

\*\*         event must be enabled. See [SetEventMask] and [GetEventMask]

\*\*         methods. This event is available only if a [Interrupt

\*\*         service/event] is enabled.

\*\*     @param

\*\*         UserDataPtr     - Pointer to the user or

\*\*                           RTOS specific data. The pointer passed as

\*\*                           the parameter of Init method.

\*/

/\* ===================================================================\*/

void TI1\_OnInterrupt(LDD\_TUserData \*UserDataPtr);

/\* END Events \*/

#ifdef \_\_cplusplus

}  /\* extern "C" \*/

#endif

#endif

/\* ifndef \_\_Events\_H\*/

/\*!

\*\* @}

\*/

/\*

\*\* ###################################################################

\*\*

\*\*     This file was created by Processor Expert 10.3 [05.09]

\*\*     for the Freescale Kinetis series of microcontrollers.

\*\*

\*\* ###################################################################

\*/

Código do main.c:

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

\*\*     Filename    : main.c

\*\*     Project     : Ex5

\*\*     Processor   : MKL25Z128VLK4

\*\*     Version     : Driver 01.01

\*\*     Compiler    : GNU C Compiler

\*\*     Date/Time   : 2023-07-07, 14:11, # CodeGen: 0

\*\*     Abstract    :

\*\*         Main module.

\*\*         This module contains user's application code.

\*\*     Settings    :

\*\*     Contents    :

\*\*         No public methods

\*\*

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

/\*!

\*\* @file main.c

\*\* @version 01.01

\*\* @brief

\*\*         Main module.

\*\*         This module contains user's application code.

\*/

/\*!

\*\*  @addtogroup main\_module main module documentation

\*\*  @{

\*/

/\* MODULE main \*/

/\* Including needed modules to compile this module/procedure \*/

#include "Cpu.h"

#include "Events.h"

#include "TI1.h"

#include "TU1.h"

#include "AD1.h"

#include "AdcLdd1.h"

#include "Bit1\_Green\_LED.h"

#include "BitIoLdd1.h"

#include "Bit2\_Blue\_LED.h"

#include "BitIoLdd2.h"

/\* Including shared modules, which are used for whole project \*/

#include "PE\_Types.h"

#include "PE\_Error.h"

#include "PE\_Const.h"

#include "IO\_Map.h"

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

uint16\_t adc\_value;

/\*lint -save  -e970 Disable MISRA rule (6.3) checking. \*/

int main(void)

/\*lint -restore Enable MISRA rule (6.3) checking. \*/

{

  /\* Write your local variable definition here \*/

  /\*\*\* Processor Expert internal initialization. DON'T REMOVE THIS CODE!!! \*\*\*/

  PE\_low\_level\_init();

  /\*\*\* End of Processor Expert internal initialization.                    \*\*\*/

  /\* Write your code here \*/

  /\* For example: for(;;) { } \*/

  while(1) {

    if (adc\_value > 200) {

      Bit1\_Green\_LED\_SetVal(); // OFF

      Bit2\_Blue\_LED\_ClrVal();  // ON

    } else if (adc\_value > 50) {

      Bit1\_Green\_LED\_ClrVal(); // ON

      Bit2\_Blue\_LED\_SetVal();  // OFF

    } else {

      Bit1\_Green\_LED\_SetVal(); // OFF

      Bit1\_Green\_LED\_SetVal(); // OFF

    }

   }

  /\*\*\* Don't write any code pass this line, or it will be deleted during code generation. \*\*\*/

  /\*\*\* 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!!! \*\*\*/

/\* END main \*/

/\*!

\*\* @}

\*/

/\*

\*\* ###################################################################

\*\*

\*\*     This file was created by Processor Expert 10.3 [05.09]

\*\*     for the Freescale Kinetis series of microcontrollers.

\*\*

\*\* ###################################################################

\*/