Nome: Gabriel Lujan Bonassi

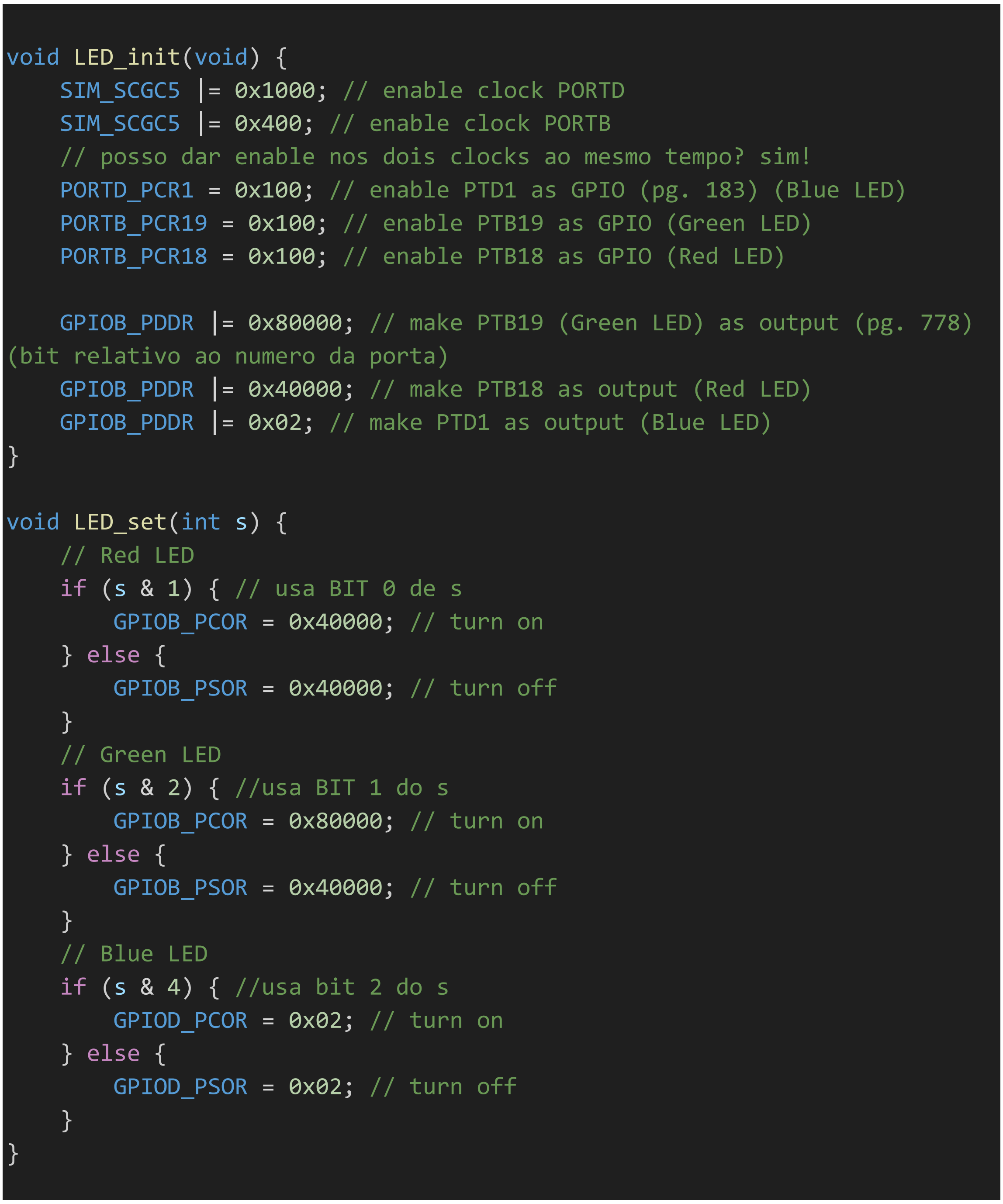
Nº USP: 11256816

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:

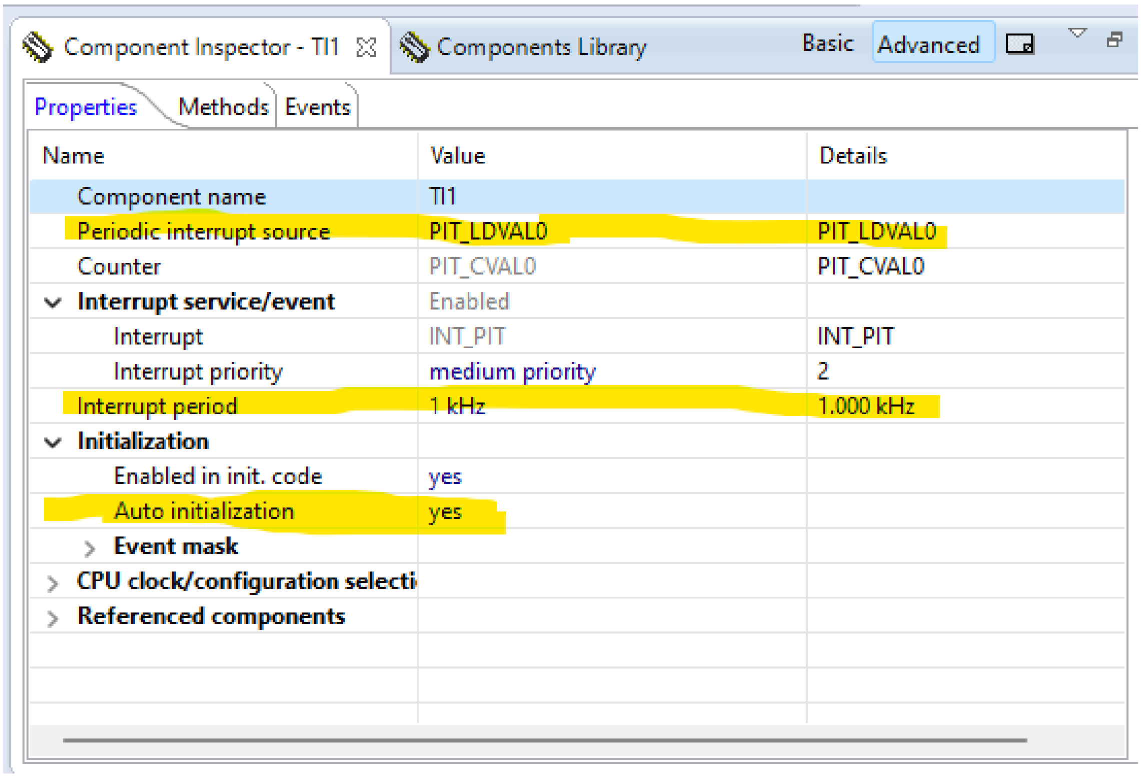
```
#include "derivative.h" /* include peripheral declarations */

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
    }
}
```

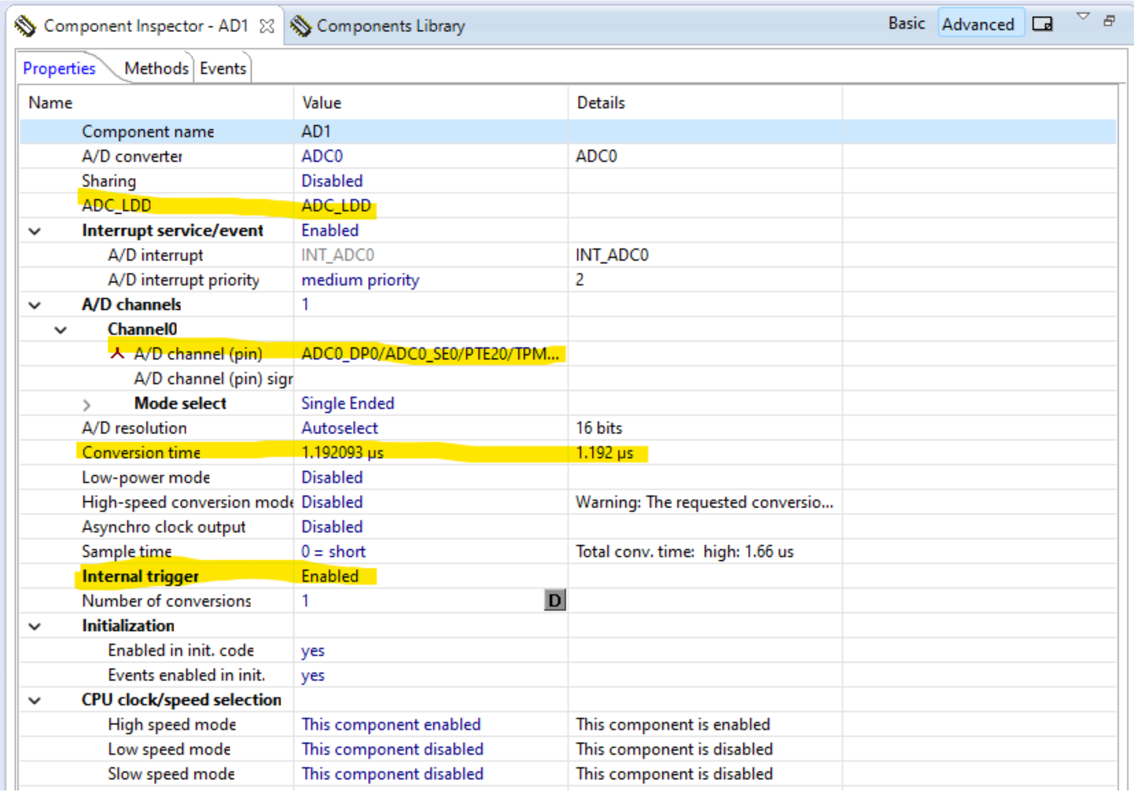
3) Utilizar um timer periódico para, por interrupção, disparar a conversão AD. Usar a interrupção de fim de conversão para acender os LEDs. Permitido o uso do Processor Expert para este item.

Configurando o timer periódico usando o “Component” TimerInt_LDD:



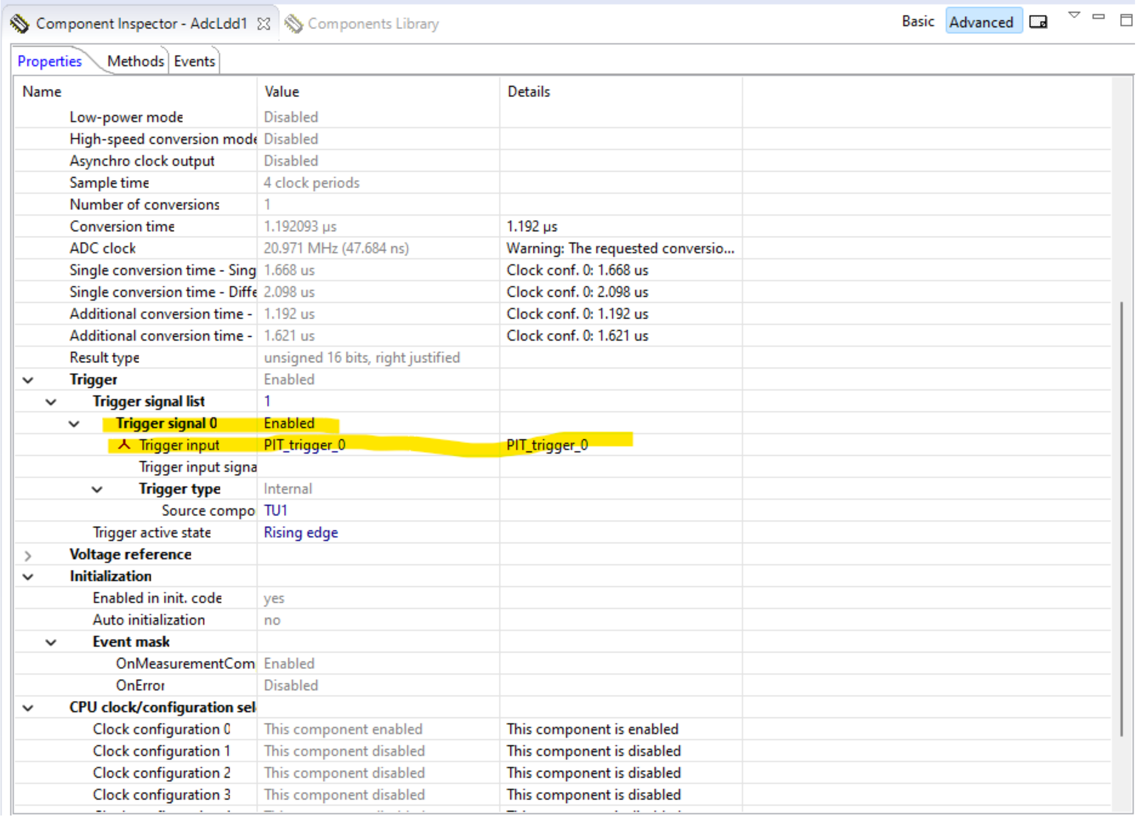
Name	Value	Details
Component name	TI1	
Periodic interrupt source	PIT_LDVAL0	PIT_LDVAL0
Counter	PIT_CVAL0	PIT_CVAL0
Interrupt service/event	Enabled	
Interrupt	INT_PIT	INT_PIT
Interrupt priority	medium priority	2
Interrupt period	1 kHz	1.000 kHz
Initialization		
Enabled in init. code	yes	
Auto initialization	yes	
Event mask		
CPU clock/configuration selecti		
Referenced components		

Configurando o “Component” ADC:



Name	Value	Details
Component name	AD1	
A/D converter	ADC0	ADC0
Sharing	Disabled	
ADC_LDD	ADC_LDD	
Interrupt service/event	Enabled	
A/D interrupt	INT_ADC0	INT_ADC0
A/D interrupt priority	medium priority	2
A/D channels	1	
Channel0		
A/D channel (pin)	ADC0_DP0/ADC0_SE0/PTE20/TPM...	
A/D channel (pin) sigr		
Mode select	Single Ended	
A/D resolution	Autoselect	16 bits
Conversion time	1.192093 µs	1.192 µs
Low-power mode	Disabled	
High-speed conversion mode	Disabled	Warning: The requested conversio...
Asynchro clock output	Disabled	
Sample time	0 = short	Total conv. time: high: 1.66 us
Internal trigger	Enabled	
Number of conversions	1 D	
Initialization		
Enabled in init. code	yes	
Events enabled in init.	yes	
CPU clock/speed selection		
High speed mode	This component enabled	This component is enabled
Low speed mode	This component disabled	This component is disabled
Slow speed mode	This component disabled	This component is disabled

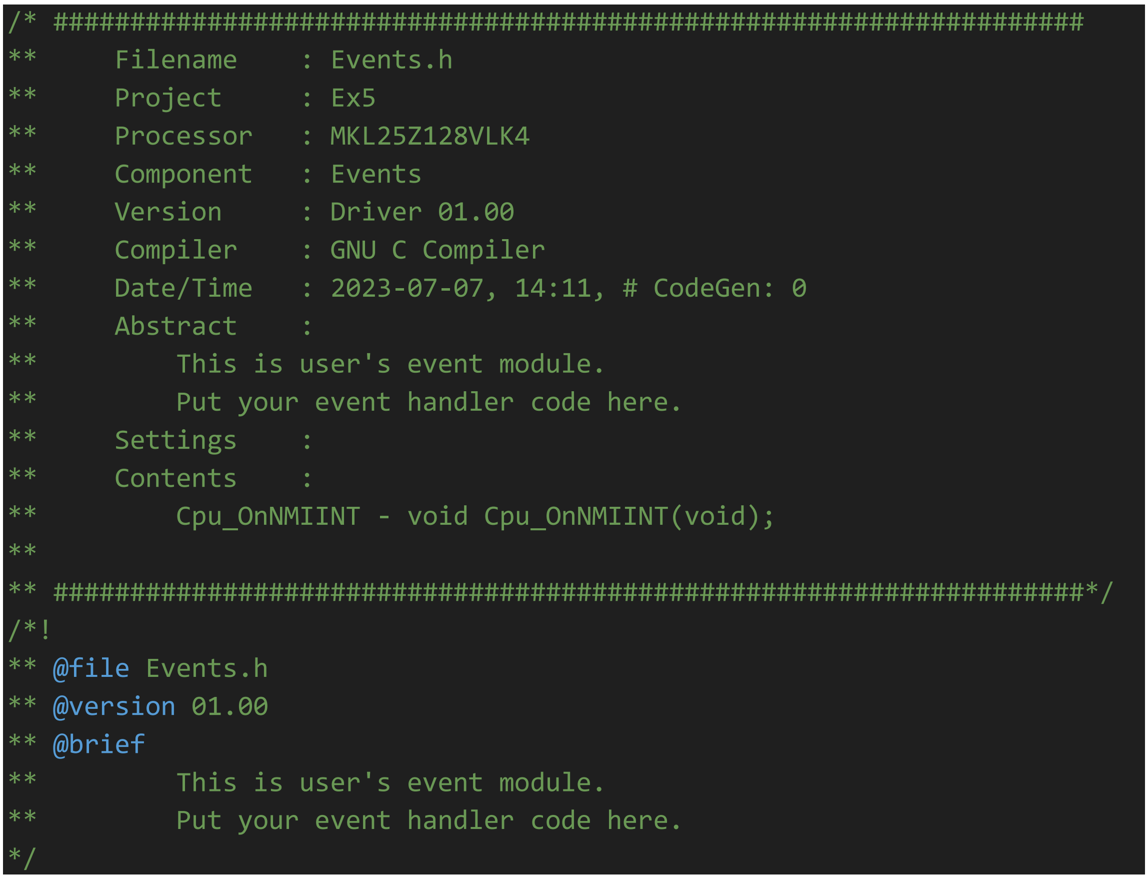
Repare que o “Internal trigger” está em “Enabled”. Vamos configurar o input do trigger do ADC utilizando o Timer que criamos:



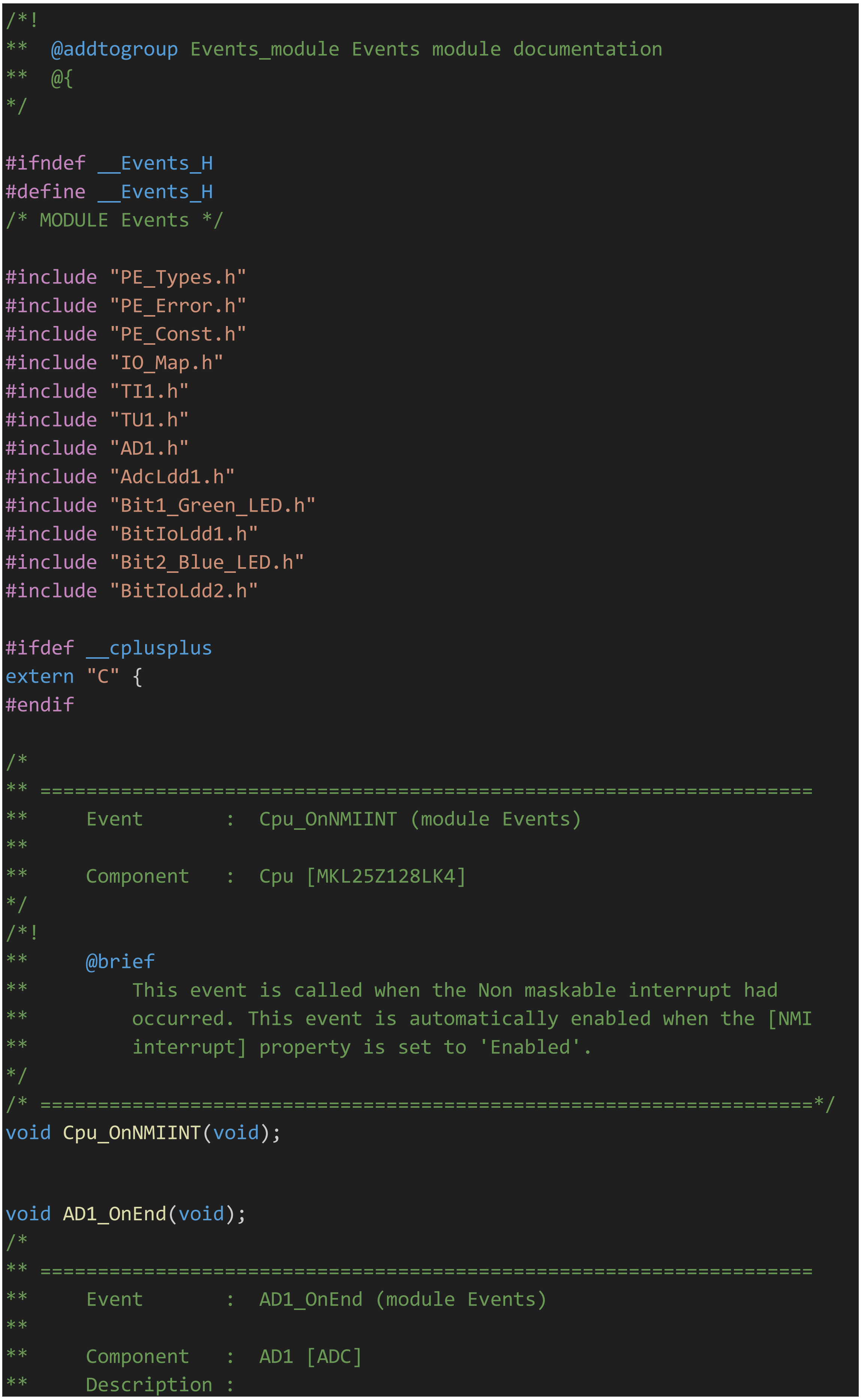
Name	Value	Details
Low-power mode	Disabled	
High-speed conversion mode	Disabled	
Asynchro clock output	Disabled	
Sample time	4 clock periods	
Number of conversions	1	
Conversion time	1.192093 µs	1.192 µs
ADC clock	20.971 MHz (47.684 ns)	Warning: The requested conversio...
Single conversion time - Sing	1.668 us	Clock conf. 0: 1.668 us
Single conversion time - Diffe	2.098 us	Clock conf. 0: 2.098 us
Additional conversion time -	1.192 us	Clock conf. 0: 1.192 us
Additional conversion time -	1.621 us	Clock conf. 0: 1.621 us
Result type	unsigned 16 bits, right justified	
Trigger	Enabled	
Trigger signal list	1	
Trigger signal 0	Enabled	
Trigger input	PIT_trigger_0	PIT_trigger_0
Trigger input signa		
Trigger type	Internal	
Source compo	TU1	
Trigger active state	Rising edge	
Voltage reference		
Initialization		
Enabled in init. code	yes	
Auto initialization	no	
Event mask		
OnMeasurementCom	Enabled	
OnError	Disabled	
CPU clock/configuration sel		
Clock configuration 0	This component enabled	This component is enabled
Clock configuration 1	This component disabled	This component is disabled
Clock configuration 2	This component disabled	This component is disabled
Clock configuration 3	This component disabled	This component is disabled

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

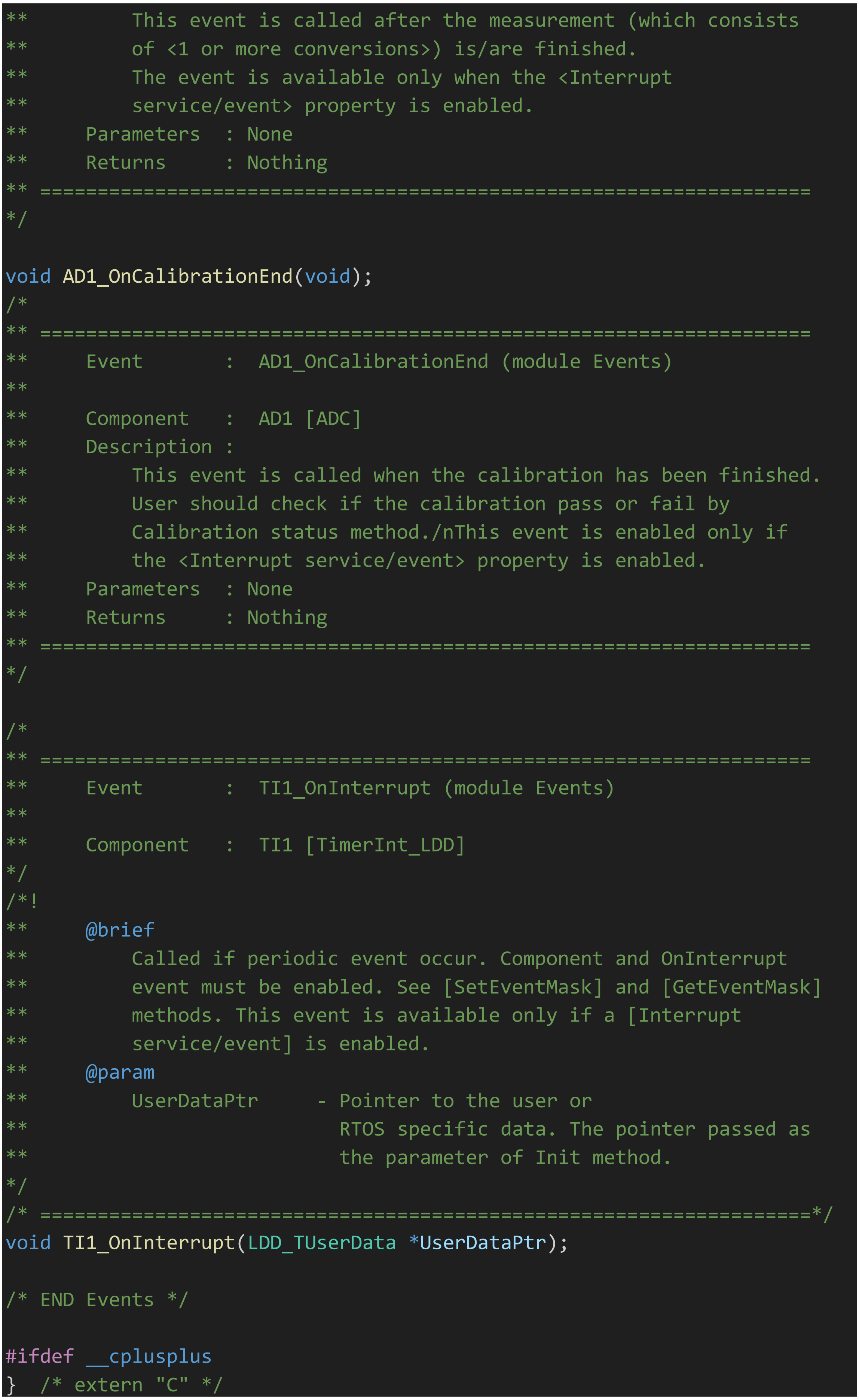
Código do events.c:



```
/* ###################################################################
**     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 :
```

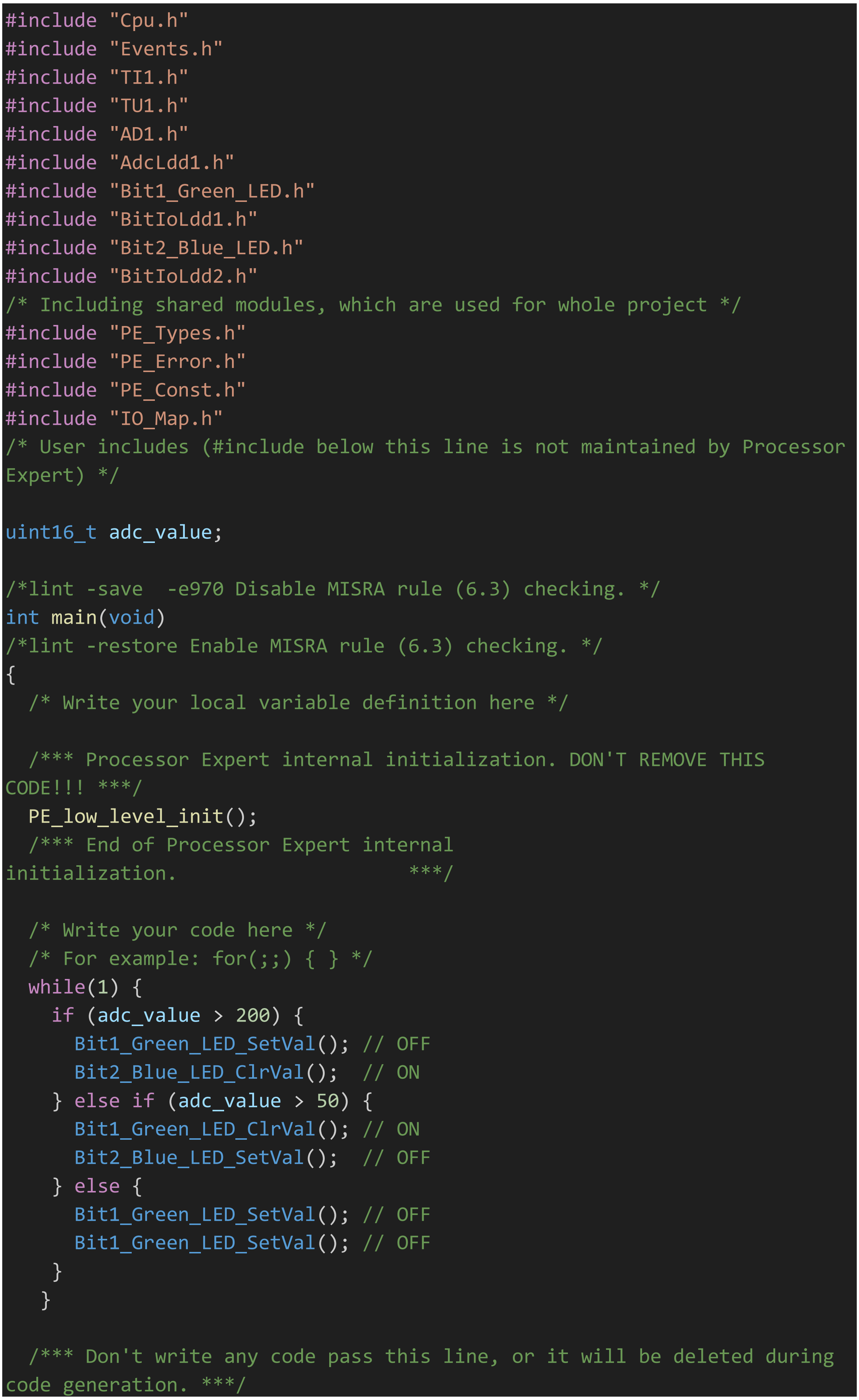


```
**          This event is called after the measurement (which consists
**          of <1 or more conversions>) is/are finished.
**          The event is available only when the <Interrupt
**          service/event> property is enabled.
**      Parameters  : None
**      Returns     : Nothing
** ===================================================================
*/

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" */
```

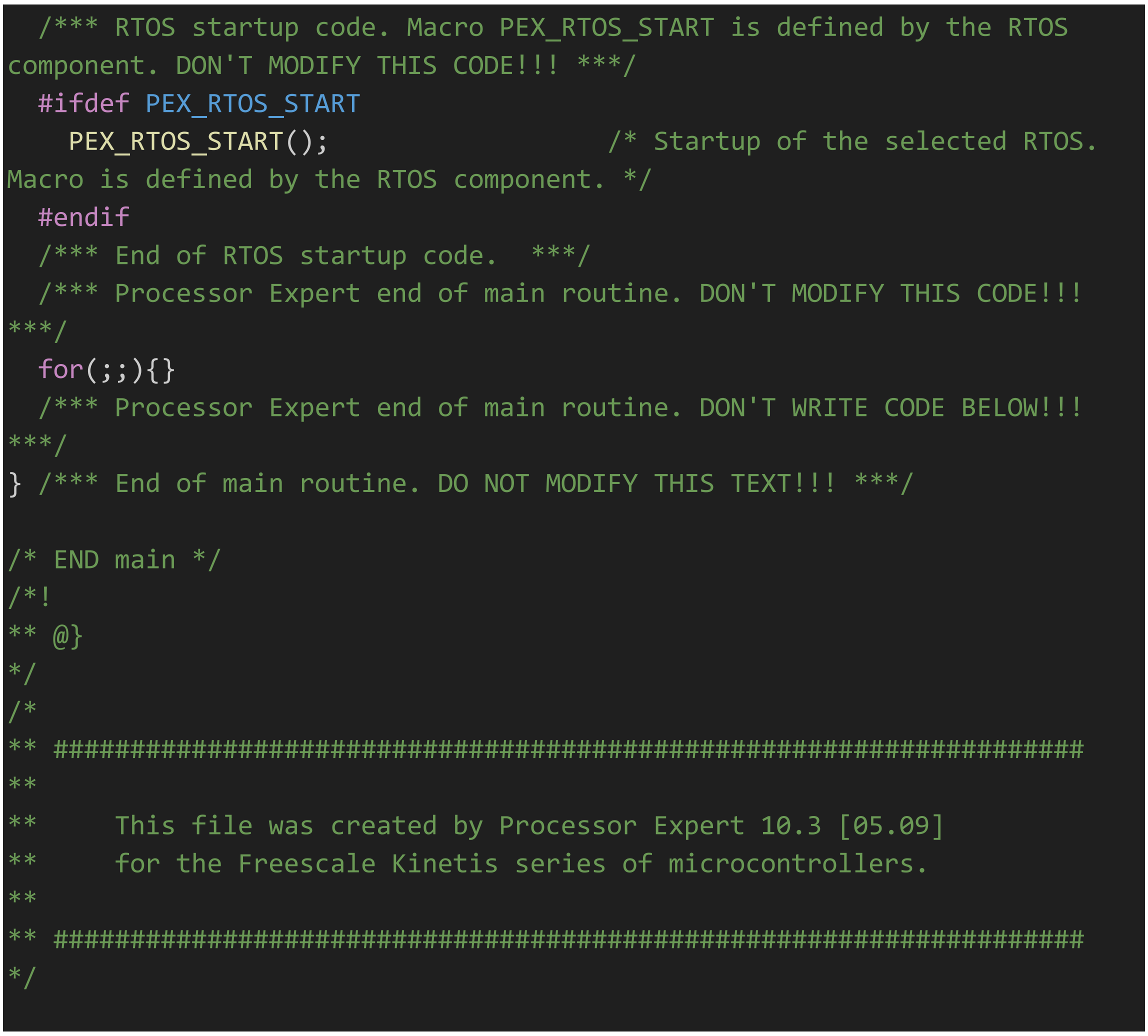
```
#endif

#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.
**
** ###################################################################
*/
```