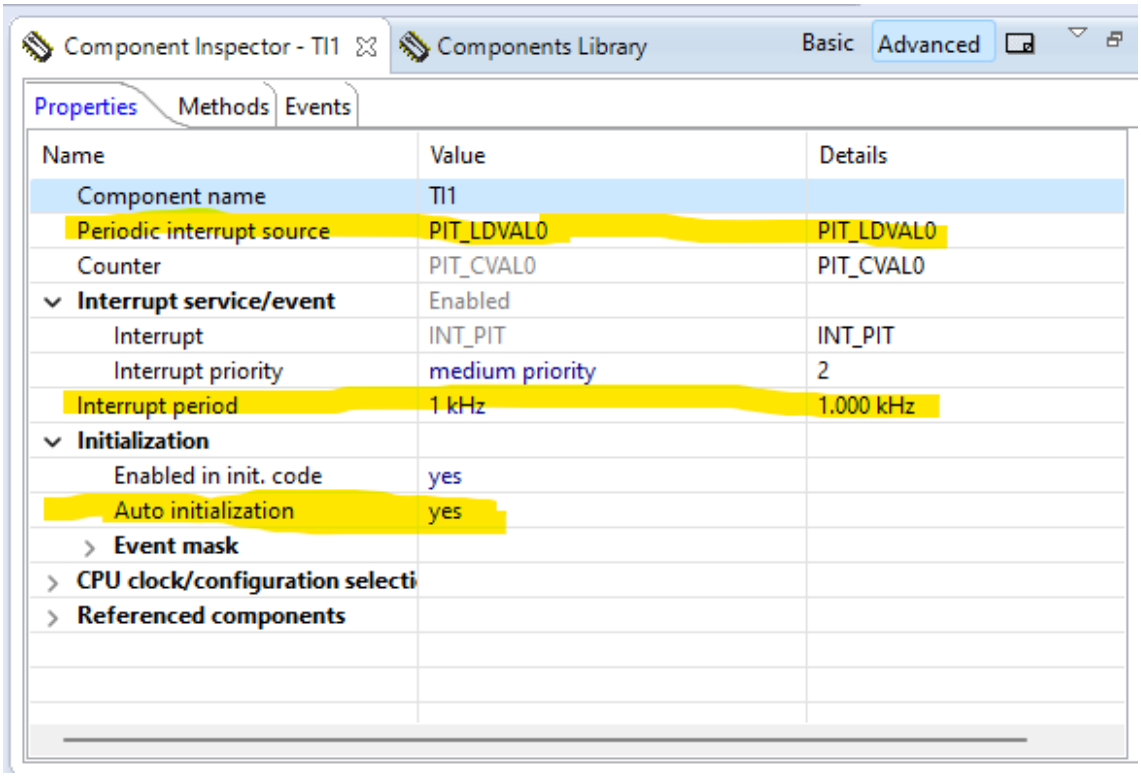
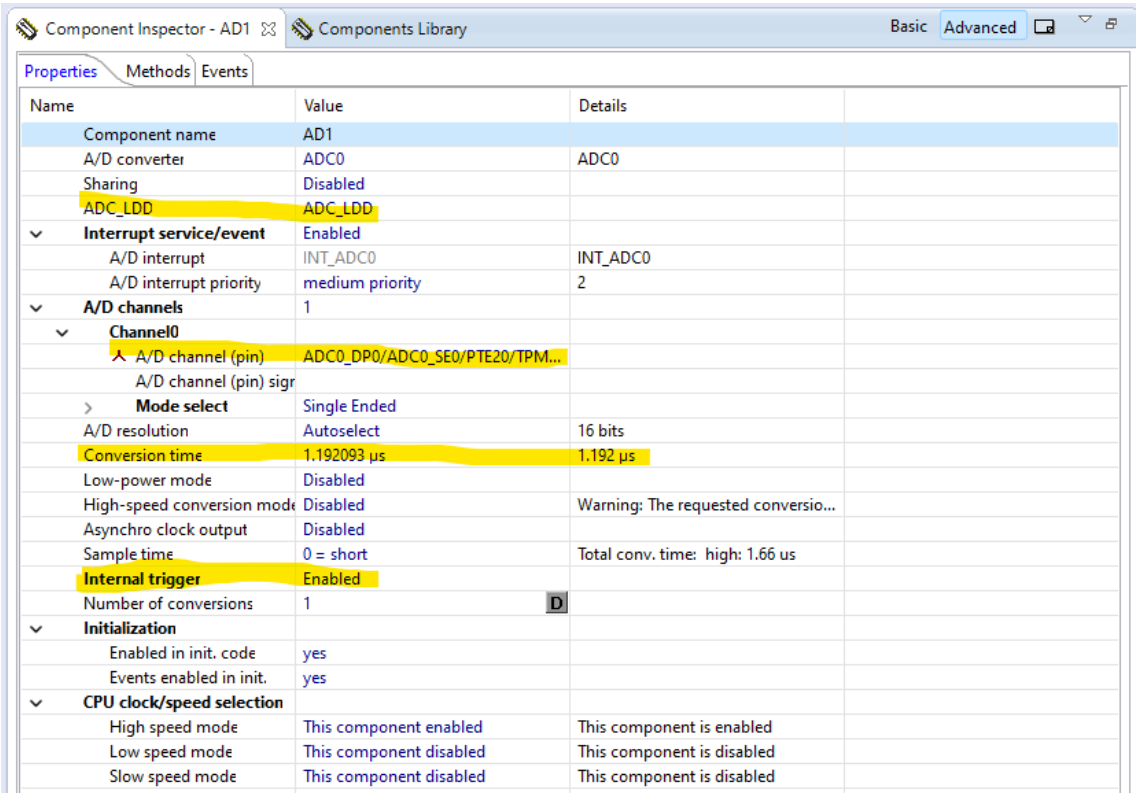
Nome: Gabriel Lujan Bonassi

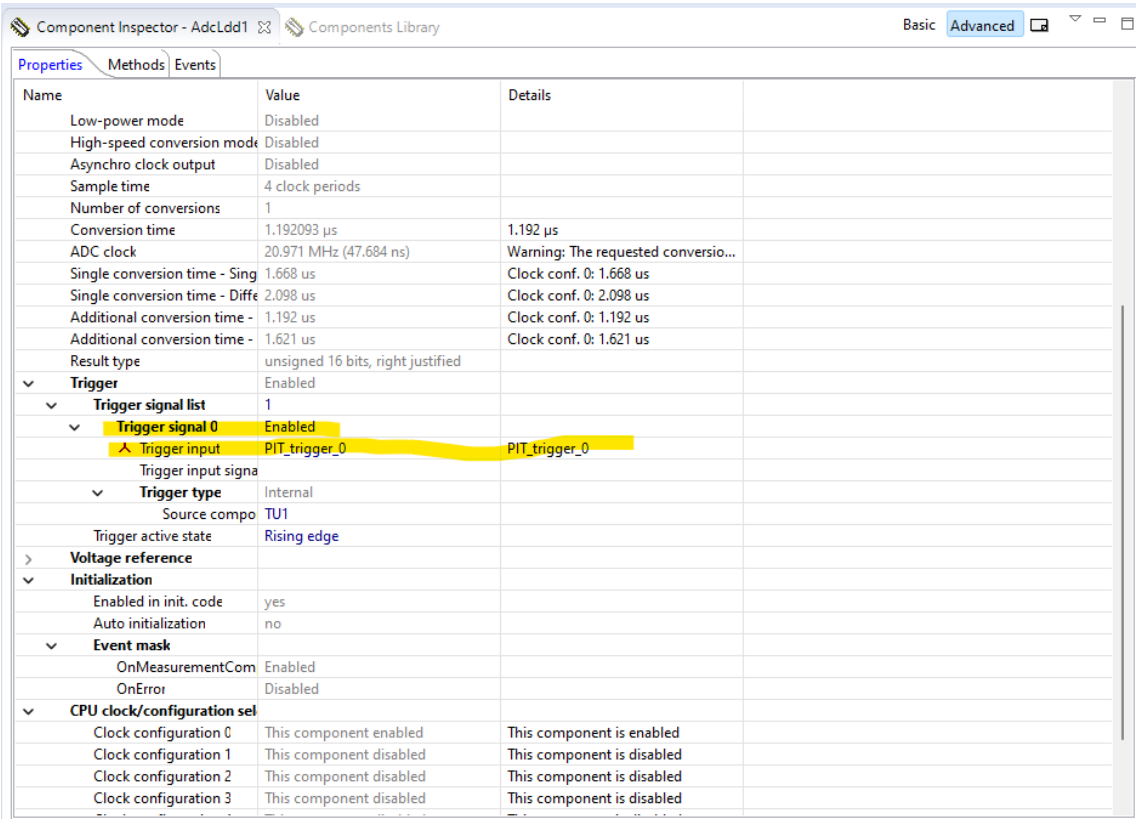
Nº USP: 11256816

1) Utilizar um timer periódico para, por interrupção, disparar a conversão AD por hardware. Usar a interrupção de fim de conversão para acender os LEDs como feito no exercício 7.

Da mesma maneira que realizado no exercício 5, vamos configurar os “Components” do Timer, ADC e os LEDs:







Código do events.h:

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

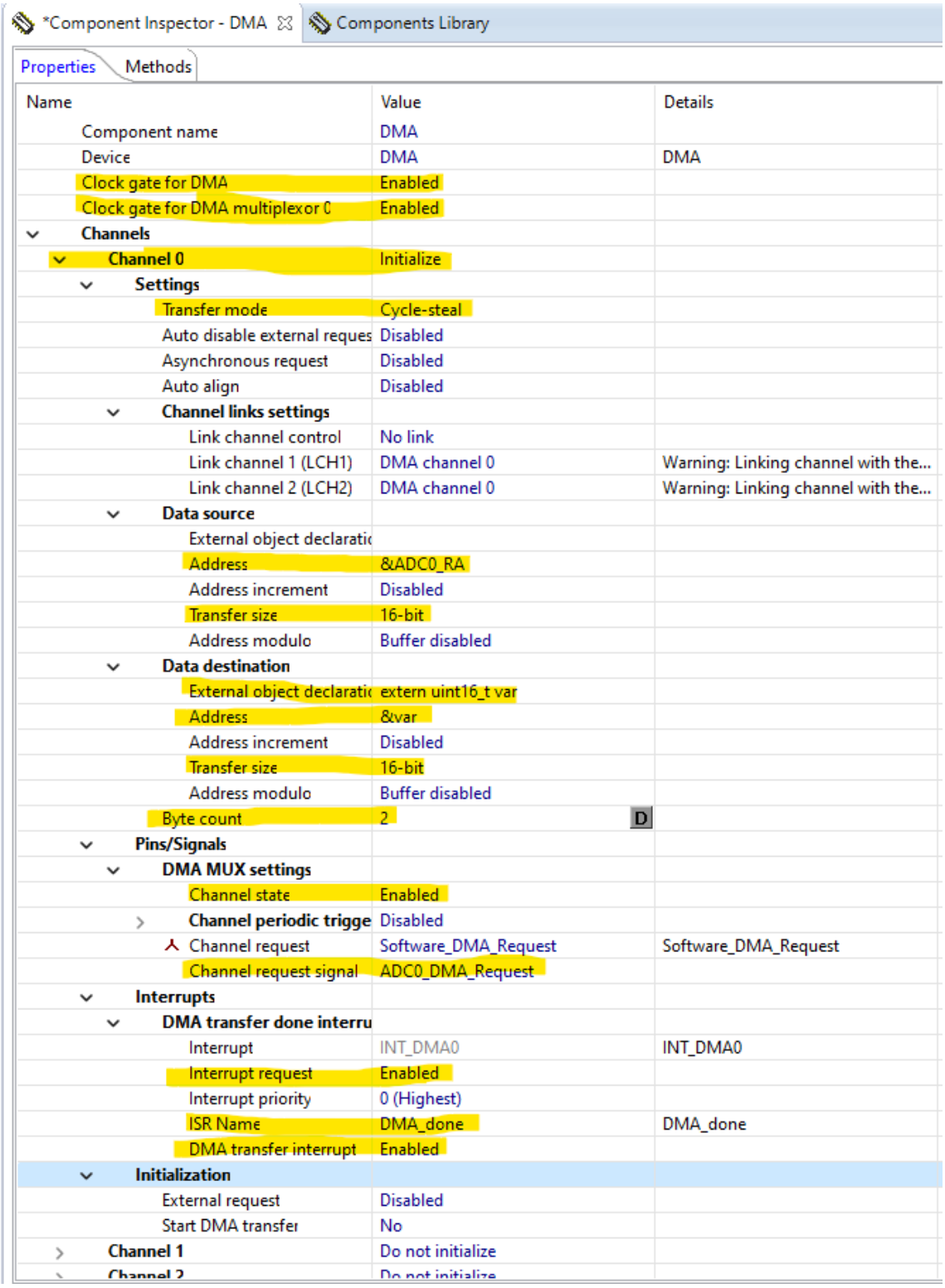
```
  /* Write your code here */
  /* For example: for(;;) { } */
  while(1) {
    if (adc_value > 200) {
      Bit1_Green_LED_SetVal(); // OFF
      Bit2_Blue_LED_ClrVal();  // ON
    } else if (adc_value > 50) {
      Bit1_Green_LED_ClrVal(); // ON
      Bit2_Blue_LED_SetVal();  // OFF
    } else {
      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.
**
** ###################################################################
*/
```

2) Utilize DMA para gravar os dados na memória.

Vamos utilizar um “Component” Init_DMA com as seguintes configurações: Inicializando o Channel 0, configurando o Data Source, o Data Destination, o Byte Count e o DMA



Por último, geramos os seguintes arquivos: events.h, main.c e DMA.c

Código do events.h:

```
/* ###################################################################
**     Filename    : Events.c
**     Project     : ex6
**     Processor   : MKL25Z128VLK4
```

```
**     Component   : Events
**     Version     : Driver 01.00
**     Compiler    : GNU C Compiler
**     Date/Time   : 2022-07-07, 01:35, # 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"

#ifdef __cplusplus
extern "C" {
#endif

extern uint16_t adc_value;

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

/*
** ===================================================================
**     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)
{
  /* Write your code here ... */
}

/*
** ===================================================================
**     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)
{
  AD1_Measure(0);
}

/*
** ===================================================================
**     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_OnEnd(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
** ===================================================================
*/
void AD1_OnCalibrationEnd(void)
{
  /* Write your code here ... */
}

PE_ISR(DMA_done)
{
    DMA_DSR0 |= DMA_DSR_BCR_DONE_MASK; // Clear Done Flag
    DMA_DSR_BCR0 |= DMA_DSR_BCR_BCR(2); // Set byte count register
}

/* END Events */

#ifdef __cplusplus
}  /* extern "C" */
#endif

/*!
** @}
*/
/*
** ###################################################################
**
**     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     : ex6
**     Processor   : MKL25Z128VLK4
**     Version     : Driver 01.01
**     Compiler    : GNU C Compiler
```

```
**     Date/Time   : 2022-07-07, 01:35, # 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 "Bit1_Green_LED.h"
#include "BitIoLdd1.h"
#include "Bit2_Blue_LED.h"
#include "BitIoLdd2.h"
#include "AD1.h"
#include "AdcLdd1.h"
#include "DMA.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"

uint16_t var;

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

/*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 */
  ADC0_SC2 |= ADC_SC2_DMAEN_MASK; // DMA Enable

  while(1) {
    if (var > 32000) {
      Bit1_Green_LED_SetVal();
      Bit2_Blue_LED_ClrVal();
    } else {
      Bit1_Green_LED_ClrVal();
      Bit2_Blue_LED_SetVal();
    }
  }

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

```
*/
```

Código do DMA.c:

```
/* ###################################################################
**     This component module is generated by Processor Expert. Do not
modify it.
**     Filename    : DMA.c
**     Project     : ex6
**     Processor   : MKL25Z128VLK4
**     Component   : Init_DMA
**     Version     : Component 01.002, Driver 01.02, CPU db: 3.00.000
**     Compiler    : GNU C Compiler
**     Date/Time   : 2022-07-09, 19:54, # CodeGen: 9
**     Abstract    :
**          This file implements the DMA (DMA) module initialization
**          according to the Peripheral Initialization settings, and
**          defines interrupt service routines prototypes.
**     Settings    :
**          Component name                                 : DMA
**          Device                                         : DMA
**          Clock gate for DMA                             : Enabled
**          Clock gate for DMA multiplexor 0               : Enabled
**          Channels                                       :
**            Channel 0                                    : Initialize
**              Settings                                   :
**                Transfer mode                            : Cycle-steal
**                Auto disable external request            : Disabled
**                Asynchronous request                     : Disabled
**                Auto align                               : Disabled
**                Channel links settings                   :
**                  Link channel control                   : No link
**                  Link channel 1 (LCH1)                  : DMA channel
0
**                  Link channel 2 (LCH2)                  : DMA channel
0
**                Data source                              :
**                  External object declaration            :
**                  Address                                : &ADC0_RA
**                  Address increment                      : Disabled
**                  Transfer size                          : 16-bit
**                  Address modulo                         : Buffer
disabled
**                Data destination                         :
**                  External object declaration            : extern
uint16_t var
**                  Address                                : &var
**                  Address increment                      : Disabled
**                  Transfer size                          : 16-bit
```

```
**                  Address modulo                          : Buffer
disabled
**                Byte count                                : 2
**              Pins/Signals                                :
**                DMA MUX settings                          :
**                  Channel state                           : Enabled
**                  Channel periodic trigger                : Disabled
**                  Channel request                         :
Software_DMA_Request
**                  Channel request signal                  :
ADC0_DMA_Request
**              Interrupts                                  :
**                DMA transfer done interrupt               :
**                  Interrupt                               : INT_DMA0
**                  Interrupt request                       : Enabled
**                  Interrupt priority                      : 0 (Highest)
**                  ISR Name                                : DMA_done
**                  DMA transfer interrupt                  : Enabled
**              Initialization                              :
**                External request                          : Disabled
**                Start DMA transfer                        : No
**            Channel 1                                     : Do not
initialize
**            Channel 2                                     : Do not
initialize
**            Channel 3                                     : Do not
initialize
**          Initialization                                  :
**            Call Init method                              : yes
**     Contents    :
**         Init - void DMA_Init(void);
**
**     Copyright : 1997 - 2014 Freescale Semiconductor, Inc.
**     All Rights Reserved.
**
**     Redistribution and use in source and binary forms, with or without
modification,
**     are permitted provided that the following conditions are met:
**
**     o Redistributions of source code must retain the above copyright
notice, this list
**       of conditions and the following disclaimer.
**
**     o Redistributions in binary form must reproduce the above
copyright notice, this
**       list of conditions and the following disclaimer in the
documentation and/or
**       other materials provided with the distribution.
**
```

```
**     o Neither the name of Freescale Semiconductor, Inc. nor the names
of its
**       contributors may be used to endorse or promote products derived
from this
**       software without specific prior written permission.
**
**     THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND
CONTRIBUTORS "AS IS" AND
**     ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO,
THE IMPLIED
**     WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE
**     DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS
BE LIABLE FOR
**     ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR
CONSEQUENTIAL DAMAGES
**     (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES;
**     LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON
**     ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR
TORT
**     (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE
USE OF THIS
**     SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
**
**     http: www.freescale.com
**     mail: support@freescale.com
** ###################################################################*/
/*!
** @file DMA.c
** @version 01.02
** @brief
**          This file implements the DMA (DMA) module initialization
**          according to the Peripheral Initialization settings, and
**          defines interrupt service routines prototypes.
*/
/*!
**  @addtogroup DMA_module DMA module documentation
**  @{
*/

/* MODULE DMA. */

#include "DMA.h"
  /* Including shared modules, which are used in the whole project */
#include "PE_Types.h"
#include "PE_Error.h"
#include "PE_Const.h"
```

```c
#include "IO_Map.h"
#include "Cpu.h"


/*
** ===================================================================
**     Method      :  DMA_Init (component Init_DMA)
**     Description :
**         This method initializes registers of the DMA module
**         according to the Peripheral Initialization settings.
**         Call this method in user code to initialize the module. By
**         default, the method is called by PE automatically; see "Call
**         Init method" property of the component for more details.
**     Parameters  : None
**     Returns     : Nothing
** ===================================================================
*/
/* Channel 0 data destination external object declaration */
extern uint16_t var;

void DMA_Init(void)
{
  /* SIM_SCGC7: DMA=1 */
  SIM_SCGC7 |= SIM_SCGC7_DMA_MASK;
  /* SIM_SCGC6: DMAMUX=1 */
  SIM_SCGC6 |= SIM_SCGC6_DMAMUX_MASK;
  /* DMAMUX0_CHCFG0: ENBL=0,TRIG=0,SOURCE=0 */
  DMAMUX0_CHCFG0 = DMAMUX_CHCFG_SOURCE(0x00);
  /* DMAMUX0_CHCFG1: ENBL=0,TRIG=0,SOURCE=0 */
  DMAMUX0_CHCFG1 = DMAMUX_CHCFG_SOURCE(0x00);
  /* DMAMUX0_CHCFG2: ENBL=0,TRIG=0,SOURCE=0 */
  DMAMUX0_CHCFG2 = DMAMUX_CHCFG_SOURCE(0x00);
  /* DMAMUX0_CHCFG3: ENBL=0,TRIG=0,SOURCE=0 */
  DMAMUX0_CHCFG3 = DMAMUX_CHCFG_SOURCE(0x00);
  /* DMA_DSR_BCR0: DONE=1 */
  DMA_DSR_BCR0 |= DMA_DSR_BCR_DONE_MASK;
  /* DMA_SAR0 = &ADC0_RA */
  DMA_SAR0 = (uint32_t)(&ADC0_RA);
  /* DMA_DAR0 = &var */
  DMA_DAR0 = (uint32_t)(&var);
  /* DMA_DSR_BCR0: ??=0,CE=0,BES=0,BED=0,??=0,REQ=0,BSY=0,DONE=0,BCR=2 */
  DMA_DSR_BCR0 = DMA_DSR_BCR_BCR(0x02);
  /* DMA_DCR0:
EINT=1,ERQ=0,CS=1,AA=0,??=0,??=0,??=0,??=0,EADREQ=0,SINC=0,SSIZE=2,DINC=0
,DSIZE=2,START=0,SMOD=0,DMOD=0,D_REQ=0,??=0,LINKCC=0,LCH1=0,LCH2=0 */
  DMA_DCR0 = DMA_DCR_EINT_MASK |
             DMA_DCR_CS_MASK |
             DMA_DCR_SSIZE(0x02) |
             DMA_DCR_DSIZE(0x02) |
```

```
            DMA_DCR_SMOD(0x00) |
            DMA_DCR_DMOD(0x00) |
            DMA_DCR_LINKCC(0x00) |
            DMA_DCR_LCH1(0x00) |
            DMA_DCR_LCH2(0x00);
  /* DMAMUX0_CHCFG0: ENBL=1,TRIG=0,SOURCE=0 */
  DMAMUX0_CHCFG0 = (DMAMUX_CHCFG_ENBL_MASK | DMAMUX_CHCFG_SOURCE(0x00));
}

/*
** ###################################################################
**
**  The interrupt service routine(s) must be implemented
**  by user in one of the following user modules.
**
**  If the "Generate ISR" option is enabled, Processor Expert generates
**  ISR templates in the CPU event module.
**
**  User modules:
**      main.c
**      Events.c
**
** ###################################################################
PE_ISR(DMA_done)
{
// NOTE: The routine should include actions to clear the appropriate
//       interrupt flags.
//
}
*/


/* END DMA. */
/*!
** @}
*/
/*
** ###################################################################
**
**     This file was created by Processor Expert 10.3 [05.09]
**     for the Freescale Kinetis series of microcontrollers.
**
** ###################################################################
*/
```