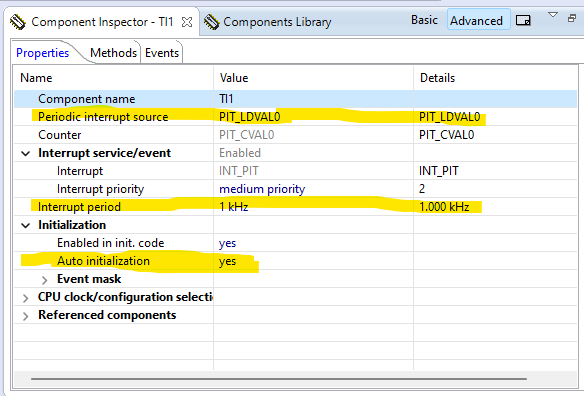
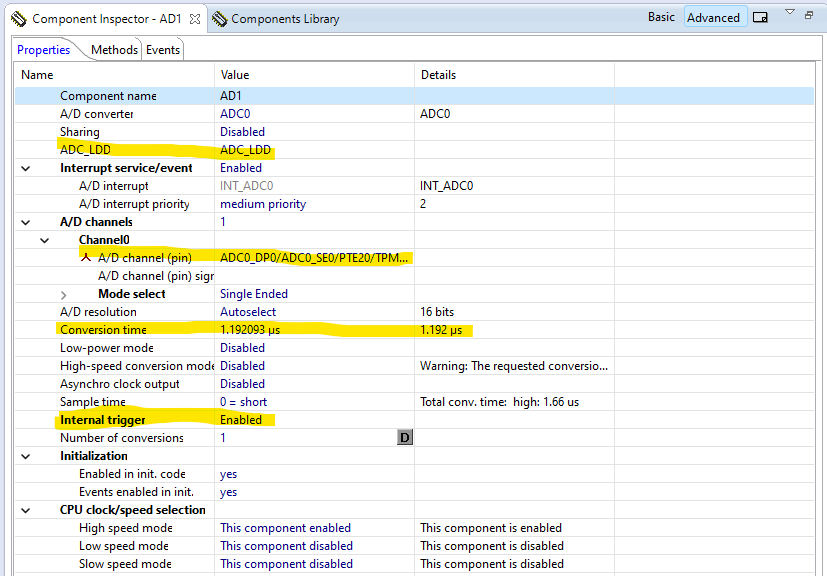
**Nome: Gabriel Lujan Bonassi**

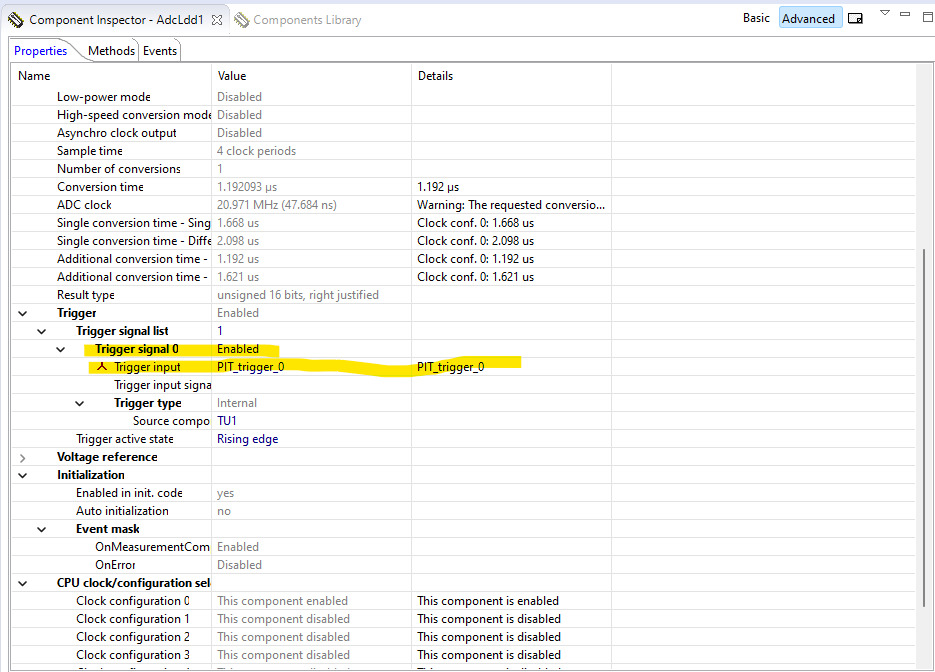
**Nº USP: 11256816**

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.

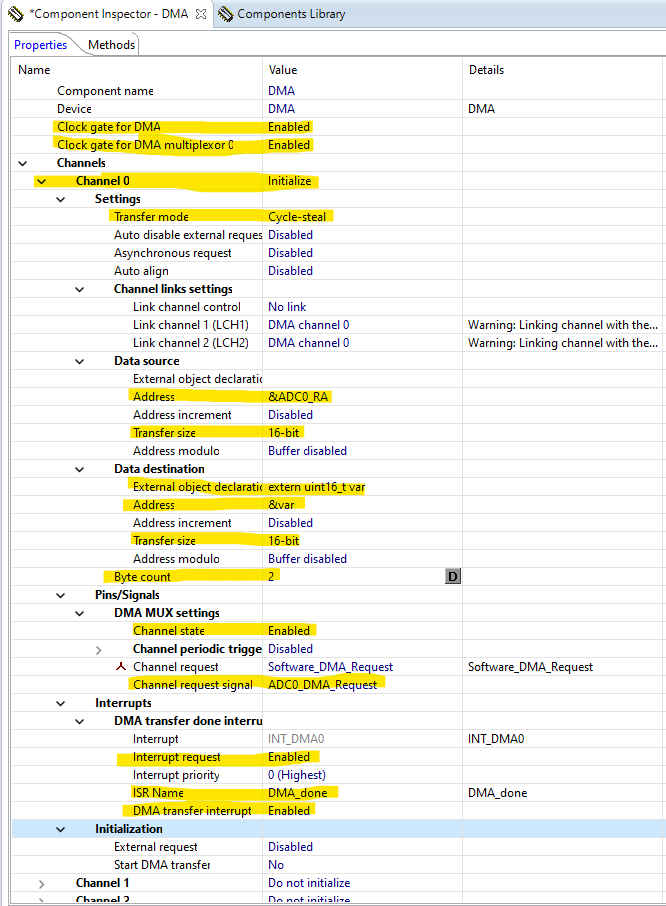
\*\*

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

\*/

1. 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"

#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.

\*\*

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

\*/