Measuring all ADC inputs on FRDM-KL25 using DMA

Version 1

Created by Petr_H on Sep 27, 2013 3:32 AM. Last modified by Petr_H on Nov 3, 2013 12:46 AM.

Introduction

The goal of this example is to read all ADC inputs of Kinetis KL25 in a row without the need of using CPU core for switching channels and pins and reading individual values.
The FRDM-KL25 board features Kinetis MKL25Z128VLK4 microcontroller. This MCU contains a 16-bit AD converter with 16 inputs.
In Processor Expert there is available ADC_LDD component which can be used for measuring values on these pins. However, there is a limitation that some of the input pins (e.g. ADC0_SE4a and ADC0_SE4b) are muxed to same channel and ADC_LDD doesn't allow to measure such two pins at once without additional mux-switching code. The MCU also does not provide an option of scanning through the ADC channels and the channels need to be switched by the user. To resolve the goal of measuring all inputs in a row the Peripheral Initialization components and DMA (Direct Memory Access) peripheral can be used.

Project Description

Note: The archive with the example project is attached to this article.

The DMA in this example is used for controlling all the channel switching, pin mux selection and reading the results for series of measurement into a memory buffer.
Results are written to serial console (virtual serial port) provided by the FRDM board.

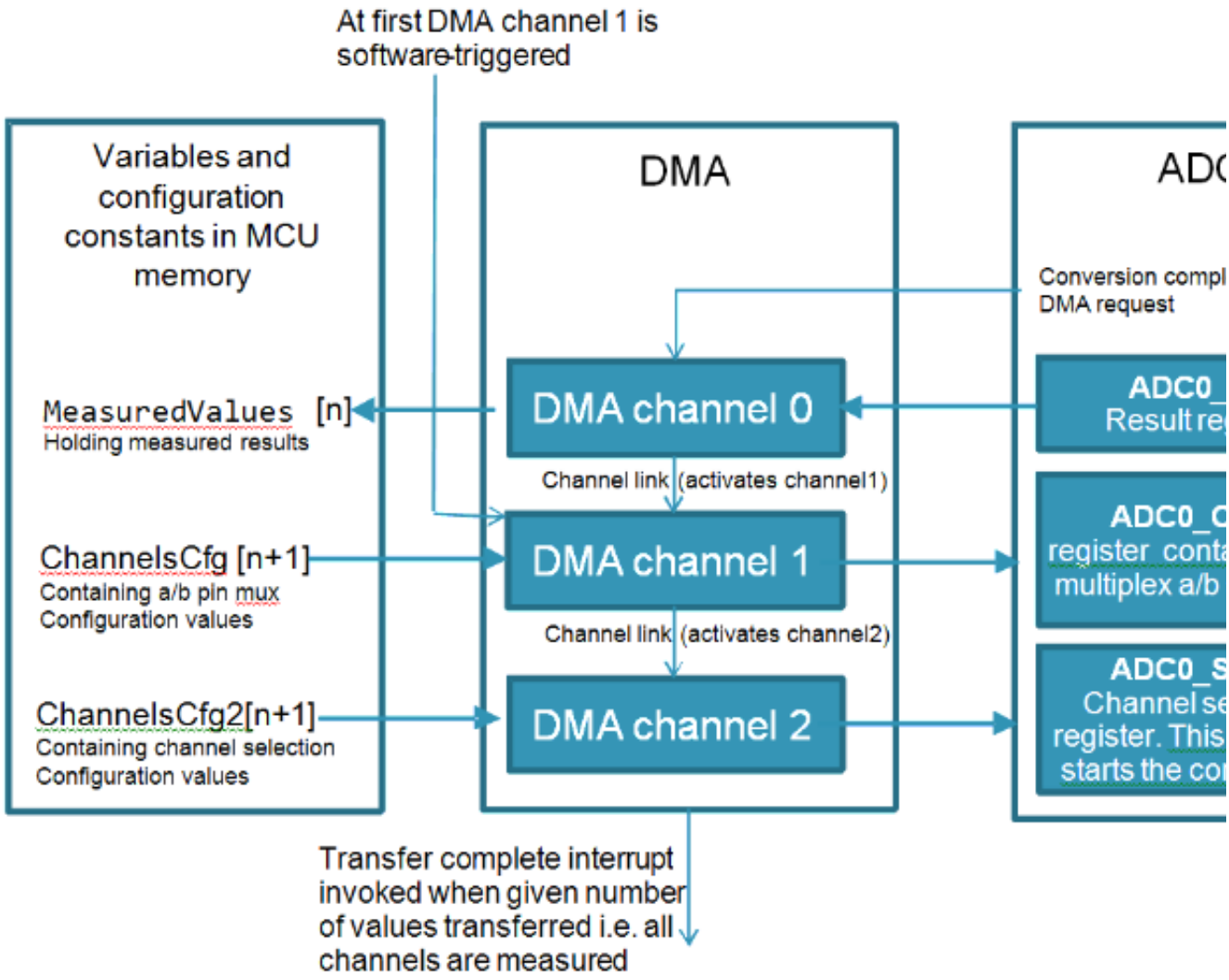
The Direct Memory Access (DMA) channels are configured for writing and reading ADC registers the following way:

- DMA channel 0 reads converted results (ADC0_RA register)
- DMA channel 1 changes the ADC pin group selection multiplexer (ADC0_CFG2 register) using values from memory array *ChannelsCfg*
- DMA channel 2 selects the ADC channel and starts conversion (ADC0_SC1A register) using values from memory array *ChannelsCfg2*

The data for the DMA channel 1 a 2 are prepared in the *ChannelsCfg* and *ChannelsCfg2* arrays prepared in memory and the DMA operates in the following cycle:
In the beginning, the DMA channel 1 transfer is started using software trigger. This selects the pin (a/b).
Then, DMA channel 2 is immediately executed because of enabled DMA channel linking. This configures the

channel and starts the conversion.
After the conversion is complete, the result is read by DMA channel 0 and stored to results array. Channel linking executes the Channel 1 transfer and the cycle continues.

After all needed channels are measured (DMA byte counter reaches 0), the DMA Interrupt is invoked so the user code is notified.
See the following figure describing the process:



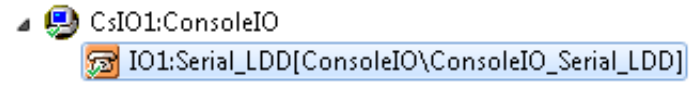
Component Configuration

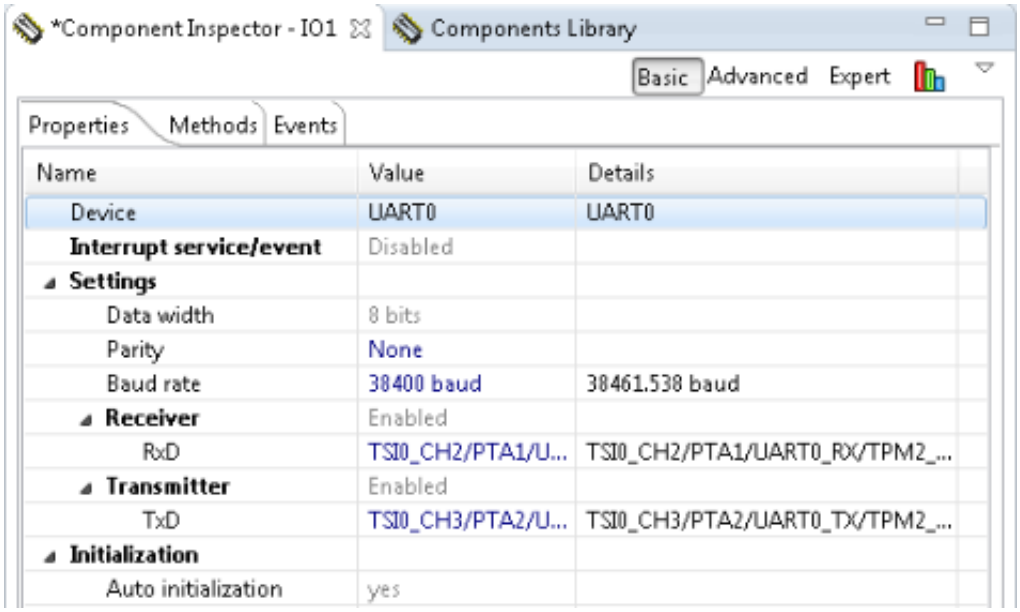
The application uses generated driver modules from the following Processor Expert components:

ConsoleIO properties setup

This component redirects printf command output to FRDM USB virtual serial port which is connected

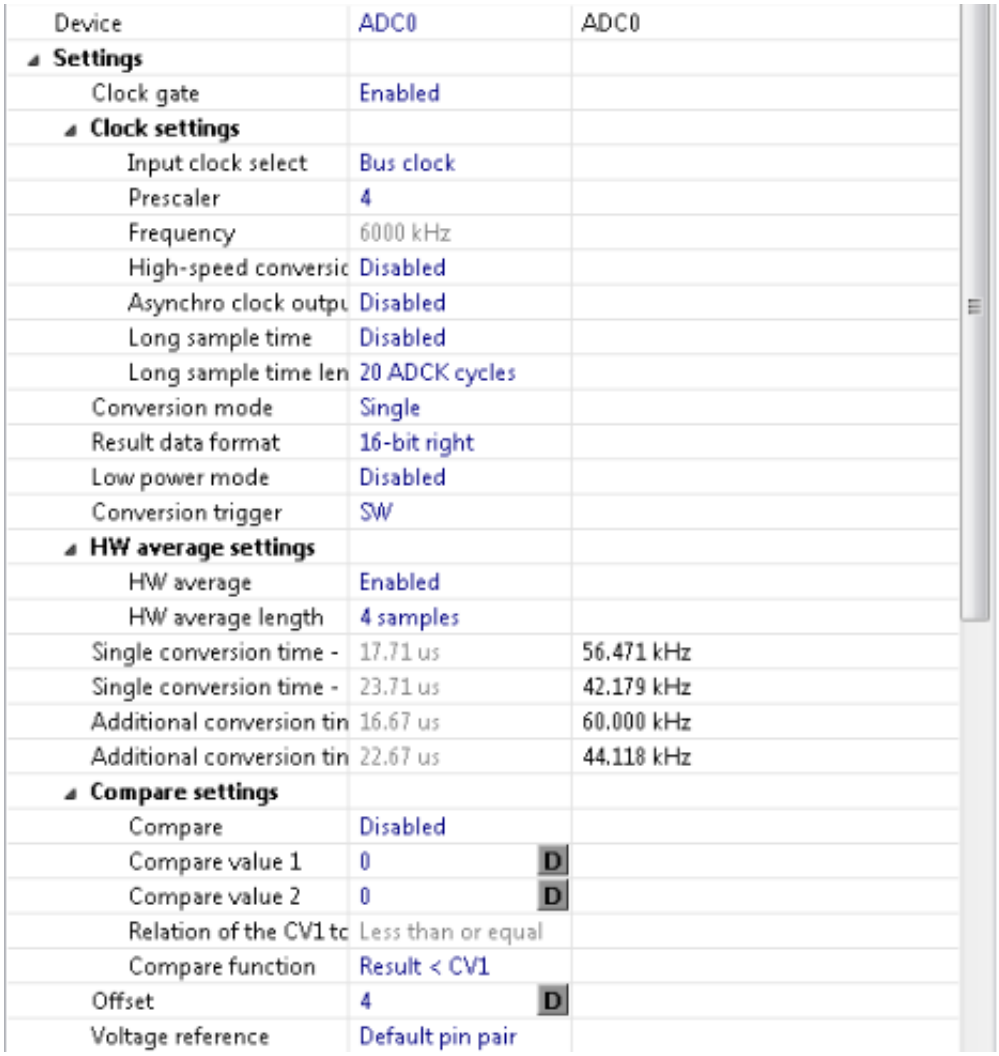
to UART0 pins PTA1/UART0_RX and PTA2/UART0_TX.
The serial device, speed and pins are configured in inherited Serial_LDD component.





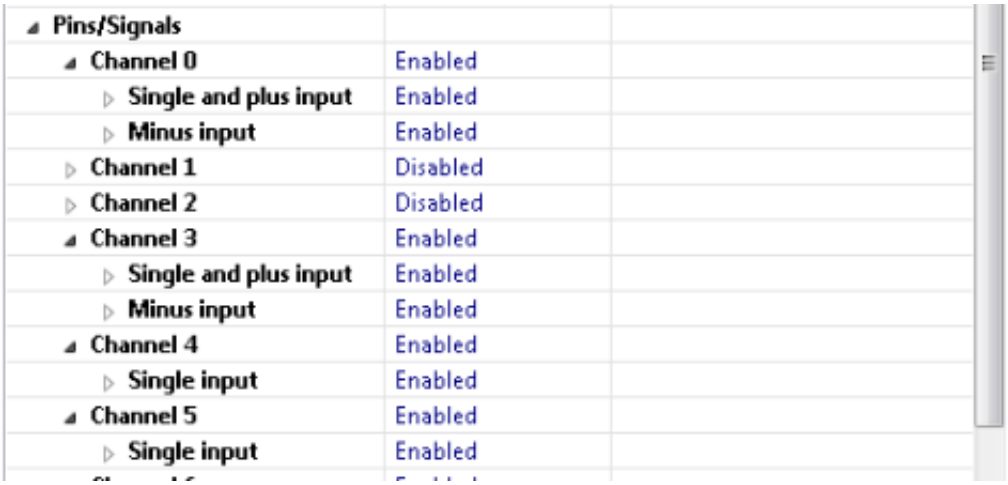
Init_ADC properties setup

The Init_ADC provides ADC initialization code with all channels enabled and set to single-ended.
The clock can be selected according to any valid value, according to the user needs.The same with HW average settings.
Compare functionality will not be used in this demo.



Device	ADC0	ADC0
Settings		
Clock gate	Enabled	
Clock settings		
Input clock select	Bus clock	
Prescaler	4	
Frequency	6000 kHz	
High-speed conversic	Disabled	
Asynchro clock outpu	Disabled	
Long sample time	Disabled	
Long sample time len	20 ADCK cycles	
Conversion mode	Single	
Result data format	16-bit right	
Low power mode	Disabled	
Conversion trigger	SW	
HW average settings		
HW average	Enabled	
HW average length	4 samples	
Single conversion time -	17.71 us	56.471 kHz
Single conversion time -	23.71 us	42.179 kHz
Additional conversion tin	16.67 us	60.000 kHz
Additional conversion tin	22.67 us	44.118 kHz
Compare settings		
Compare	Disabled	
Compare value 1	0 D	
Compare value 2	0 D	
Relation of the CV1 tc	Less than or equal	
Compare function	Result < CV1	
Offset	4 D	
Voltage reference	Default pin pair	

Pins configuration - all pins available on the board are enabled:



Pins/Signals	
Channel 0	Enabled
Single and plus input	Enabled
Minus input	Enabled
Channel 1	Disabled
Channel 2	Disabled
Channel 3	Enabled
Single and plus input	Enabled
Minus input	Enabled
Channel 4	Enabled
Single input	Enabled
Channel 5	Enabled
Single input	Enabled

Interrupts, DMA and Triggering - Interrupts are disabled, DMA request is enabled. Triggering is disabled, as it's not used in this demo project, However, the application could be extended to use it.

▲ **Interrupts/DMA**		
Interrupt	INT_ADC0	INT_ADC0
Interrupt request	Disabled	
Interrupt priority	0 (Highest)	
ISR name		
Conversion complete A inte	Disabled	
Conversion complete B inte	Disabled	
DMA request	Enabled	
▲ **Initialization**		
▲ **ADC part triggered by trig**		
Initial channel select A	ADC disabled	
Differential mode A	Disabled	
▲ **ADC part triggered by trig**		
Initial channel select B	ADC disabled	
Differential mode B	Disabled	
Call Init method	no	

Init_DMA properties setup

The Init_DMA provides provides initialization code for the DMA.
Clock gate and DMA multiplexor are enabled:

Name	Value	Details
Device	DMA	DMA
Clock gate for DMA	Enabled	
Clock gate for DMA multiplexor 0	Enabled	

DMA Channel 0

- 16-bit results are transferred from ADC0_RA register (see proprerty **Data source / Address**).
- Transfer mode is Cycle-steal, which means that only one transaction is done per each external request.
- The destination address initial value is not filled in the inspector because it's filled repeatedly in the application code.
- Channel linking is set to trigger channel 1 after each transfer
- DMA mux settings for the Channel 0 are enabled and ADC0_DMA_Request is selected, which is the signal from ADC when the conversion ends.
- "DMA transfer done" interrupt for this channel is enabled. The ADCint ISR function will be called.
- External request (request from ADC) is Enabled to start the transfer. Byte count will also be changed before every sequence

Property values:

Channels		
Channel 0	Initialize	
Settings		
Transfer mode	Cycle-steal	
Auto disable external request	Disabled	
Asynchronous request	Enabled	
Auto align	Disabled	
Channel links settings		
Link channel control	LCH1 after each cycle steal transfer	
Link channel 1 (LCH1)	DMA channel 1	
Link channel 2 (LCH2)	DMA channel 0	Warning: Linking channel with the same numb
Data source		
External object declaration		
Address	(uint32_t)&ADC0_RA	
Address increment	Disabled	
Transfer size	16-bit	
Address modulo	Buffer disabled	
Data destination		
External object declaration		
Address		
Address increment	Enabled	
Transfer size	16-bit	
Address modulo	Buffer disabled	
Byte count	16 D	
Pins/Signals		
DMA MUX settings		
Channel state	Enabled	
Channel periodic trigger	Disabled	
Channel request	ADC0_DMA_Request	ADC0_DMA_Request
Interrupts		
DMA transfer done interrupt		
Interrupt	INT_DMA0	INT_DMA0
Interrupt request	Enabled	
Interrupt priority	0 (Highest)	
ISR Name	ADCint	ADCint
DMA transfer interrupt	Enabled	
Initialization		
External request	Enabled	
Start DMA transfer	No	

DMA Channel 1

- DMA channel 1 changes the ADC pin group selection multiplexer (ADC0_CFG2 register) using values from memory array *ChannelsCfg.*
- Please note that the source address initial value is not filled, will be set in the application code along with the Byte count value.
- There is no HW trigger for this channel, it's set to be triggered by SW only (and by linking mechanism, which will be used).
- The linking from this channel is set to trigger CH2 after the transfer.
- No interrupt is enabled for this channel

Channel 1	Initialize	
Settings		
Transfer mode	Cycle-steal	
Auto disable external request	Enabled	
Asynchronous request	Enabled	
Auto align	Disabled	
Channel links settings		
Link channel control	LCH1 after each cycle steal transfer	
Link channel 1 (LCH1)	DMA channel 2	
Link channel 2 (LCH2)	DMA channel 0	
Data source		
External object declaration		
Address	0	
Address increment	Enabled	
Transfer size	8-bit	
Address modulo	Buffer disabled	
Data destination		
External object declaration		
Address	&ADC0_CFG2	
Address increment	Disabled	
Transfer size	8-bit	
Address modulo	Buffer disabled	
Byte count	8 D	
Pins/Signals		
DMA MUX settings		
Channel state	Enabled	
Channel periodic trigger	Disabled	
Channel request	Software_DMA_Request	Software_DMA_Request
Interrupts		
DMA transfer done interrupt		
Interrupt	INT_DMA1	INT_DMA1
Interrupt request	Disabled	
Interrupt priority	0 (Highest)	
ISR Name		
DMA transfer interrupt	Disabled	
Initialization		
External request	Enabled	
Start DMA transfer	No	

DMA Channel 2

- DMA channel 2 selects the ADC channel and starts conversion (ADC0_SC1A register) using values from memory array *ChannelsCfg2.*
- No channel is linked after the transfer ends - No link.
- No external channel request is selected, this channel transfer is triggered by linking from CH2.

Channel 2	Initialize	
Settings		
Transfer mode	Cycle-steal	
Auto disable external request	Enabled	
Asynchronous request	Enabled	
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	0	
Address increment	Enabled	
Transfer size	8-bit	
Address modulo	Buffer disabled	
Data destination		
External object declaration		
Address	&ADC0_SC1A	
Address increment	Disabled	
Transfer size	8-bit	
Address modulo	Buffer disabled	
Byte count	8 D	
Pins/Signals		
DMA MUX settings		
Channel state	Enabled	
Channel request	Software_DMA_Request	Software_DMA_Request
Interrupts		
DMA transfer done interrupt		
Interrupt	INT_DMA2	INT_DMA2
Interrupt request	Disabled	
Interrupt priority	0 (Highest)	
ISR Name		
DMA transfer interrupt	Disabled	
Initialization		
External request	Enabled	
Start DMA transfer	No	

TimerUnit_LDD

- It's used in the application code to provide delay to slow down console output.
- The TPM0 counter is used with the period of approx. 350ms.
- No interurpt is used.
- Auto initialization is enabled.

Module name	TPM0	TPM0
Counter	TPM0_CNT	TPM0_CNT
Counter direction	Up	
Input clock source	Internal	
Counter frequency	Auto select	187.500 kHz
Counter restart	On-match	
Period device	TPM0_MOD	TPM0_MOD
Period	349.525333 ms	349.525 ms
Interrupt	Disabled	
Channel list	0	
Initialization		
Auto initialization	yes	

Code

The channels/pins to be measured are specified in the ChannelsCfg and ChannelsCfg2 arrays.
These arrays contains a list of pins to be measured, the order can be changed according to the user needs, the channels can even be measured multiple times.
The special value 0x1F stops the conversion.

```
// configuration array for channels - channel numbers. Should ends with 0x1F which stops conversion
// seconcd onfiguration array coreesponding to channels selecting A/B pins
// For example: 0 + PIN_A corresponds to the pin ADC0_SE0,   5 + PIN_5 selects the pin ADC0_SE5b
// You can use these arrays to reorder the measurement as you need
const uint8_t ChannelsCfg [ADC_CHANNELS_COUNT + 1] =  { 0,     4,     3,     7,
4,    23,    8,     9,    11,     12,    13,    14,    15,    5,     6,     7,     0x1F
};
const uint8_t ChannelsCfg2[ADC_CHANNELS_COUNT + 1] =  {PIN_A, PIN_A, PIN_A, PIN_A,
PIN_B, PIN_A, PIN_A, PIN_A, PIN_A, PIN_A, PIN_A, PIN_A, PIN_A, PIN_B, PIN_B, PIN_B,    0
};
```

In the main loop, the application first re-initializes the DMA values and strarts the sequence by software triggerring the DMA channel 1.

```
// loop
while (TRUE) {
   // clear flag
   Measured = FALSE;
   // reset DMA0 destination pointer to beginning of the buffer
   DMA_DAR0 = (uint32_t) &MeasuredValues;
   // reset DMA1 source pointer (MUX switching writes)
   DMA_SAR1 = (uint32_t) &ChannelsCfg2;
```

```
    // reset DMA2 source pointer (channel switching and conversion start writes)
    DMA_SAR2 = (uint32_t) &ChannelsCfg;
    // number of total bytes to be transfered from the ADC result register A
    DMA_DSR_BCR0 = ADC_CHANNELS_COUNT * 2;
    // set number of total bytes to be transfered to the ADC0_CFG2
    DMA_DSR_BCR1 = ADC_CHANNELS_COUNT + 1;
    // set number of total bytes to be transfered to the ADC0_SC1A.
    DMA_DSR_BCR2 = ADC_CHANNELS_COUNT + 1;
  // start first DMA1 transfer (selects mux, then fires channel 2 to select channel which starts
the conversion)
    DMA_DCR1 |= DMA_DCR_START_MASK;
  // wait till it's all measured
  while (!Measured) {}
  // print all measured values to console
  for (i=0; i<ADC_CHANNELS_COUNT; i++) {
    printf ("%7u", (uint16_t) MeasuredValues[i]);
  }
  printf ("\n");
  // reset the counter
  TU1_ResetCounter(TU1_DeviceData);
  // wait for some time to slow down output
  while (TU1_GetCounterValue(TU1_DeviceData) < 50000) {}
}
```

Running the project

The project can be run usual way.

- import the project into CodeWarrior for MCUs V10.5.
- Build the project
- Connect the FRDM-KL25 board
- Start debuging and run the code
- Run terminal application or use the Terminal view in eclpise. Set it to use the virtual serial port created for the board. The parameters should be set to 38400,no parity, 8 bits ,1 stopbit.

KL25_MultiADC_DMA_CW10_5.zip
1.2 MB

4046 Views Categories:
Tags: tips_and_tricks, kinetis, freedom-board, adc, processor_expert, dma, frdm-kl25

Average User Rating

(2 ratings)

4 Comments