DMA_ADC_Transfer_1

DMA transfer of ADC conversion results

AURIX™ TC2xx Microcontroller Training
V1.0.0

Scope of work

The DMA is used to transfer ADC measurements results to LMURAM.

At the end of an analog-to-digital conversion of the VADC module, an interrupt is triggered, which starts the data transfer of the converted ADC results via DMA to the LMURAM. The ADC conversion is started manually via a command of a serial monitor.

Introduction

- The Direct Memory Access (DMA) transfers data from data source locations to data destination locations without intervention of the CPU or other on-chip devices.
- A DMA channel performs **transactions**. One transaction is made of **transfers**. One transfer is made of up to 16 **moves**. This structure divides the data into several parts and increases the application's efficiency.
- A transaction can be interrupted, however once a transfer is started, it cannot be interrupted.
- A move operation (8-bit,16-bit, 32-bit, 64-bit, 128-bit or 256-bit):
  1. Loads data from the data source into the DMA controller
  2. Puts data from the DMA controller to a data destination
- Any DMA move engine can service a DMA request **from any of the 128 DMA channels**. Channel 127 has the highest priority.
- Example:
  - 1024 words (32-bit per word) transaction can be composed of *256 transfers of 4 DMA word moves*, or *128 transfers of 8 DMA word moves*.

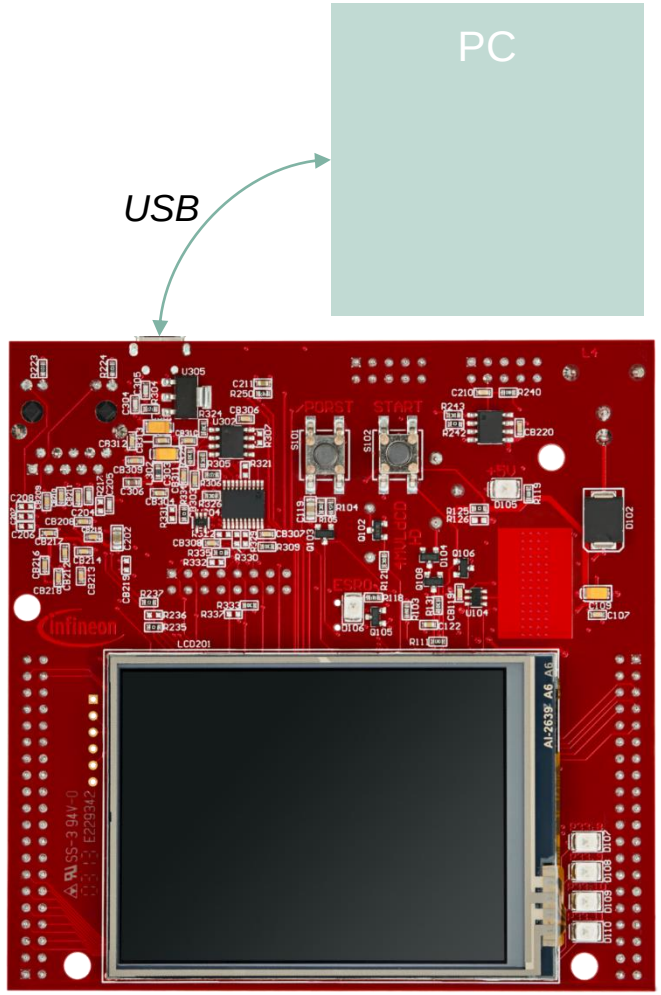
Introduction

- The Versatile Analog to Digital Converter module (VADC) of the AURIX™ TC29x comprises 11 independent analog to digital converters. Each, converting with a resolution up to 12-bit.
- Several request sources can request an Analog/Digital conversion following different configurations. A conversion can be requested to be done **once or repeatedly**.
- Interrupts can be generated once conversions are finished.

Hardware setup

This code example has been developed for the board KIT_AURIX_TC297_TFT_BC-Step.

The board should be connected to the PC via USB, in order to allow the UART connection.



Implementation

Application use case

The DMA_ADC_Transfer_1 example works as follows:

1. The user sends a ‘1’ character via a serial monitor (e.g. PuTTY, HTerm).
2. This triggers an Analog/Digital conversion.
3. Once the conversion is done and the result written in the VADC result register, an interrupt is triggered.
4. This interrupt calls a DMA transaction, since the DMA is the configured as the Interrupt Service Provider for this ADC interrupt.
5. Once the transaction is finished, an interrupt is triggered by the DMA and handled by the CPU, to send a feedback message to the user on the serial monitor.

This sequence can be repeated at any time.

Implementation

Configuration of the ADC

For this example, the configuration of **one group** using **one single channel** is sufficient.

In ***init_vadc()***, the VADC is initialized and configured. Then, Group0 is selected and configured.
To do the conversion only once after being triggered, the Auto Scan mode is disabled via the ***autoscanEnabled*** parameter of the structure's instance ***IfxVadc_Adc_GroupConfig***.
The following iLLD functions are needed:

- ***IfxVadc_Adc_initModuleConfig()*** initializes the module's structure with the default values.
- ***IfxVadc_Adc_initModule()*** initializes the VADC to run with the configured frequency and calibration.

- ***IfxVadc_Adc_initGroupConfig()*** initializes the group buffer with the default configuration.
- ***IfxVadc_Adc_initGroup()*** initializes the VADC group specified in the parameters.

- ***IfxVadc_Adc_initChannelConfig()*** initializes the channel buffer with the default configuration.
- ***IfxVadc_Adc_initChannel()*** initializes the channel with the specified parameters.
  Moreover, a value is set for the parameter ***resultPriority*** of this channel. This will trigger an interrupt every time a result is generated after conversion.

- ***IfxVadc_Adc_setScan()*** adds the provided group and channel parameters to the scan sequence.

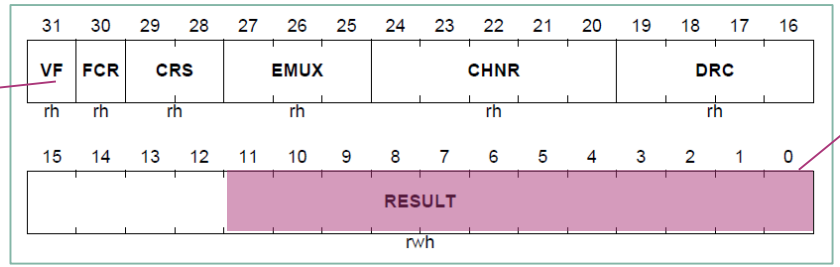
Implementation

Configuration of the ADC (cont.)

The ***run_vadc()*** function is starting the measurement and conversion of the analog values. The function is called multiple times from the while loop inside the ***CPU0_Main.c*** file.

The ***IfxVadc_Adc_startScan()*** iLLD function starts an analog to digital conversion.

Valid Flag set to 1 when a new result is written. Reset to 0 once the result has been read



12-bit result after the conversion

VADC_G0RES0 @0xF0020700

32-bit result register of Group 0 - Channel 0

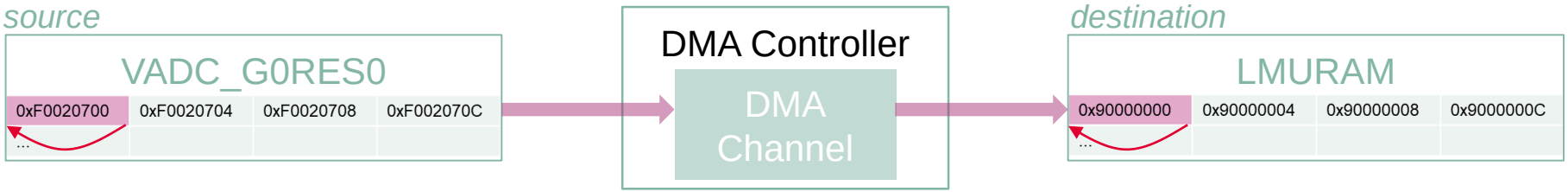
Implementation

Configuration of the DMA

The needed transaction is configured in the ***init_dma()*** function. Here, the number of transfers per transaction and the size of the word moves are defined.
In this example, we want to transfer a **32-bit result register** from the VADC. A **single transaction** with **one transfer** made of **one 32-bit word move** is fitting.

All of the above can be achieved with a **single DMA channel** (in this case: channel 1).

The data source and destination locations are also set in the same function.
Additionally, the DMA channel is configured in such a way, that the source and destination addresses are not incremented after the transaction, since we always want to transfer from the same result register to the same location in the LMURAM. This is configured via the ***destinationAddressCircularRange*** and ***sourceAddressCircularRange*** parameters.



An interrupt on the DMA channel is configured in order to send a feedback to the user. This service is provided by the CPU after each finished transaction.

Implementation

Configuration of the UART

In this tutorial, the UART connection is used to make the debugging more convenient and easier to understand.

The function ***init_uart()*** is initializing the UART communication.

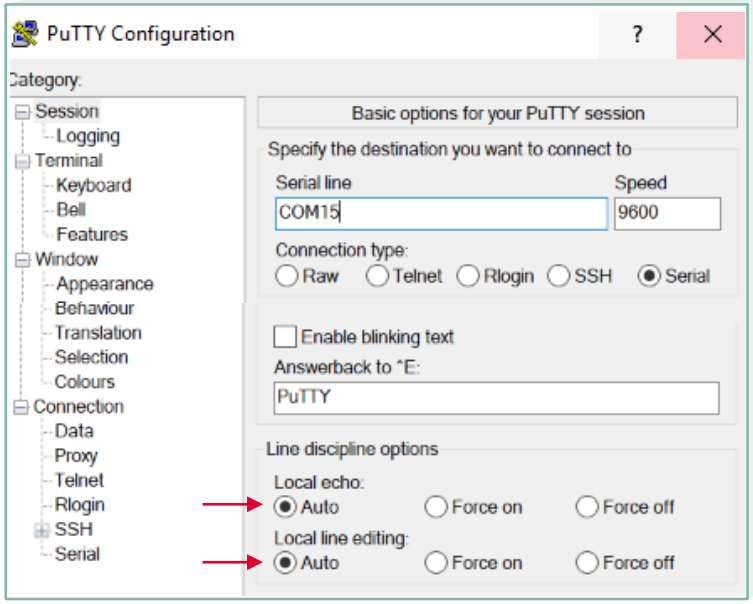
The iLLD function ***IfxAsclin_Asc_initModuleConfig()*** fills the configuration structure ***ascConf*** with the default values. Then, the parameters are set to their correct value, depending on the needed connection: baudrate, Tx and Rx buffer size, Tx and Rx pin configuration etc.

Finally, ***IfxAsclin_Asc_initModule()*** initializes the module with the user configuration done previously.

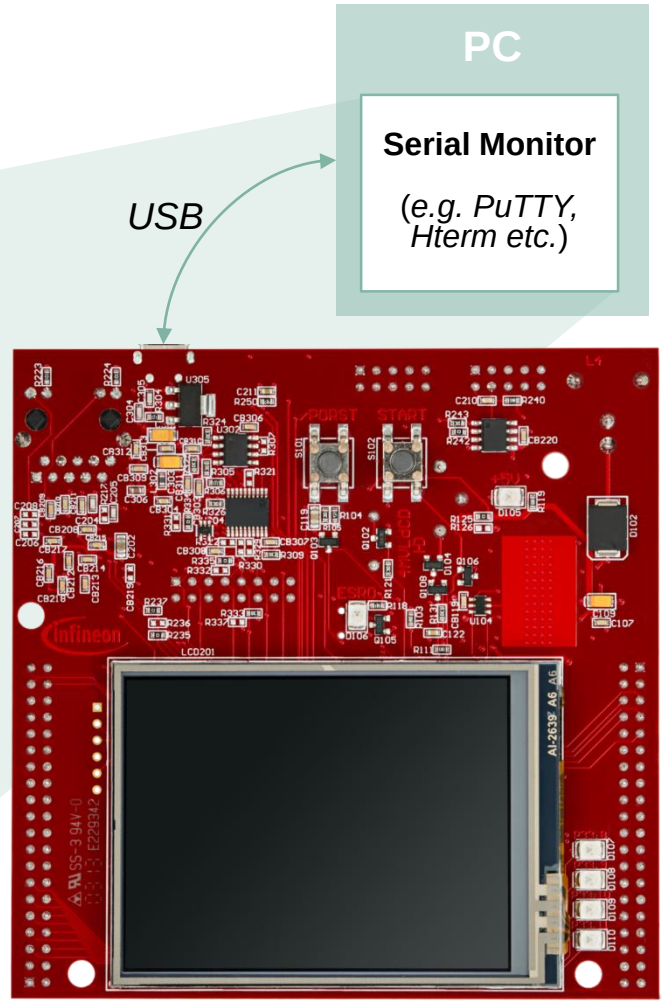
Software setup

Serial monitor settings:

- Speed (baud): 9600
- Data bits: 8
- Stop bit: 1



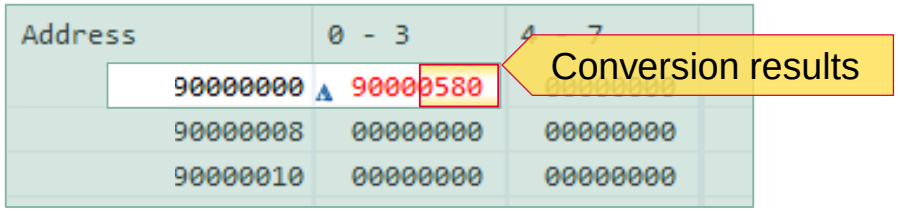
Example on PuTTY



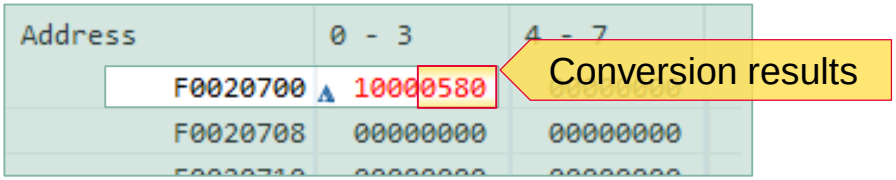
Run and Test

After code compilation and flashing the device, perform the following steps:

- Connect the board to the PC
- Open the serial monitor with the appropriate COM port and settings
- Send ‘1’
- The successful DMA transfer can be observed with the debugger by adding the below address in the memory watcher at the same time:
  - Check the memory at the LMURAM address



  - It matches the result of the VADC peripheral result register *



- Send ‘1’ again to start another conversion

*_**9**0000578h to **1**0000578h because Valid Flag (VF, bit 31) is reset to 0 after reading the VADC result register_

References

- AURIX™ Development Studio is available online:
- https://www.infineon.com/aurixdevelopmentstudio
- Use the *„Import...“* function to get access to more code examples.

- More code examples can be found on the GIT repository:
- https://github.com/Infineon/AURIX_code_examples

- For additional trainings, visit our webpage:
- https://www.infineon.com/aurix-expert-training

- For questions and support, use the AURIX™ Forum:
- https://www.infineonforums.com/forums/13-Aurix-Forum

Edition 2019-10
Published by
Infineon Technologies AG
81726 Munich, Germany

Do you have a question about this document?
Email: erratum@infineon.com

Document reference
DMA_ADC_Transfer_1