ADC_Single_Channel_1 for KIT_AURIX_TC397_TFT

ADC single channel conversion

AURIX™ TC3xx Microcontroller Training
V1.0.0

Scope of work

The Enhanced Versatile Analog-to-Digital Converter (EVADC) is configured to measure an analog signal using queued request.

An analog input channel is continuously converted using the queued mode. The input value is determined using the microcontroller's supply voltage, ground level or letting the analog pin open and floating. Three LEDs are used, each indicating a voltage interval. Thus depending on the conversion value, a certain LED will light up.

Introduction

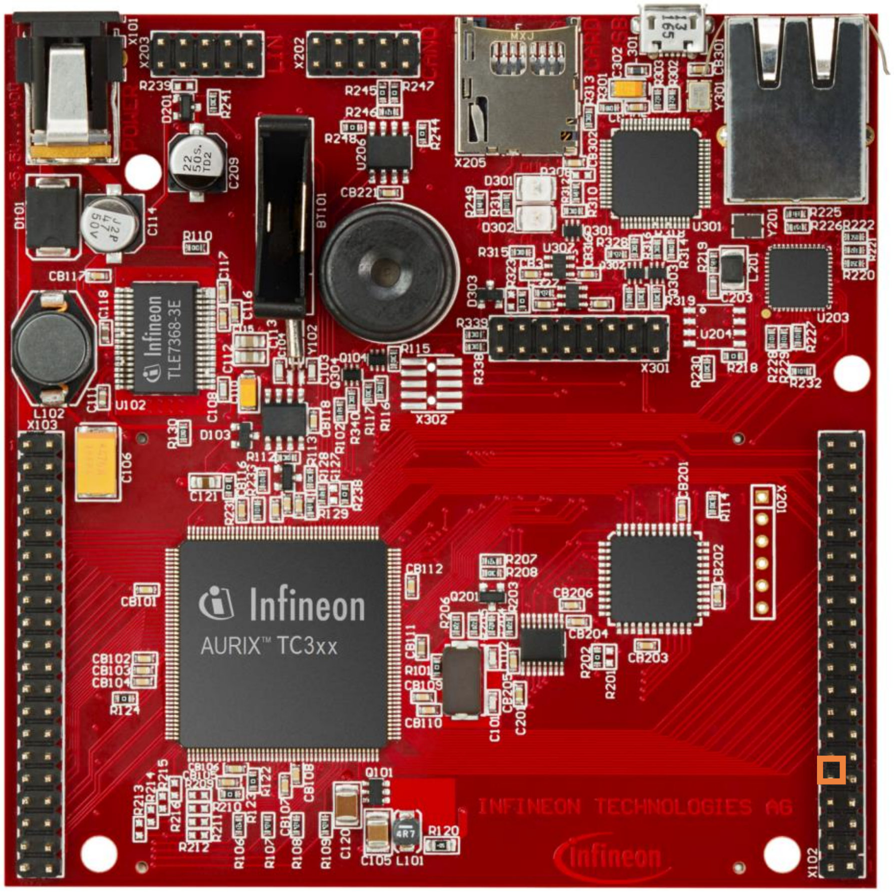
- The AURIX™ microcontrollers provide a series of analog input channels (up to 16 for each ADC) connected to a cluster of Analog/Digital Converters (up to 12) using the Successive Approximation Register (SAR) principle. Each converter of the ADC cluster is represented as a group and can operate independently of the others.

- Analog/Digital conversions can be requested by one request source:
  - **Queued request source**, specific to a single group

- A queued source can issue conversion requests for an arbitrary sequence of input channels. The channel numbers for this sequence can be freely programmed.

- The trigger for the conversion via the queued source can be sent:
  - Once (by another external module)
  - On a regular time base (by an external timer)
  - Permanently (by using the refill option)

Hardware setup



This code example has been developed for the board KIT_A2G_TC397_5V_TFT. In this example, the port pin AN2 is used.

X102

P14.5	40	39	P14.4
P33.10	38	37	P20.9
P15.7	36	35	P15.6
P15.5	34	33	P15.4
P15.3	32	31	P15.2
P22.3	30	29	P22.2
P22.1	28	27	P22.0
P33.11	26	25	P23.4
P23.3	24	23	P23.2
P23.1	22	21	P23.0
P33.6	20	19	P33.8
P33.12	18	17	P33.1
P33.2	16	15	P33.3
P33.4	14	13	P33.5
AN0	12	11	AN8
AN2	10	9	AN3
AN11	8	7	AN13
AN20	6	5	AN21
GND	4	3	GND
V_UC	2	1	VCC_IN

Note: The channels can be HW filtered by the board, depending on which capacitor/resistors couples are soldered. Consult the Application Kit's Manual to check which channels are filtered by HW.

Implementation

Configuration of the EVADC

The configuration of the VADC is done in the ***initEVADC()*** function in four different steps:

- Configuration of the **EVADC module**
- Configuration of the **EVADC group**
- Configuration of the **EVADC channels**
- Filling the queue

Configuration of the EVADC module with the function *initEVADCModule()*

The default configuration of the EVADC module, given by the iLLDs, can be used for this example.
This is done by initializing an instance of the ***IfxEvadc_Adc_Config*** structure and applying default values to its fields through the function ***IfxEvadc_Adc_initModuleConfig()***.
Then, the configuration can be applied to the EVADC module with the function ***IfxEvadc_Adc_initModule()***.

Implementation

Configuration of the EVADC group with the function *initEVADCGroup()*

The configuration of the EVADC group is done by initializing an instance of the ***IfxEvadc_Adc_GroupConfig*** structure with default values through the function ***IfxEvadc_Adc_initGroupConfig()*** and modifying the following fields:

- ***groupId*** – to select which converters to configure
- ***master*** – to indicate which converter is the master. In this example, only one converter is used, therefore it is also the master
- ***arbiter*** – a structure that represents the enabled request sources. In this example, it is set to ***arbiter.requestSlotQueue0Enabled***.

Then, the user configuration is applied through the function ***IfxEvadc_Adc_initGroup()***.

Implementation

Configuration of the EVADC channels with the function *initEVADCChannels()*

The configuration of each channel is done by initializing a separate instance of the ***IfxEvadc_Adc_ChannelConfig*** structure with default values through the function ***IfxEvadc_Adc_initChannelConfig()*** and modifying the following fields:

- ***channelId*** – to select the channel to configure
- ***resultRegister*** – to indicate the register where the A/D conversion value is stored

Then, the configuration is applied to the channel with the function ***IfxEvadc_Adc_initChannel()***.

Filling the queue

Each channel is added to the queue through the function ***IfxEvadc_Adc_addToQueue()***.

When the VADC configuration is done and the queue is filled, the conversion is started with the function ***IfxEvadc_Adc_startQueue()***.

Finally, to read a conversion, the function ***IfxEvadc_Adc_getResult()*** from iLLDs is used inside the function ***readEVADC()***.

All the functions used for configuring the EVADC module, its groups and channels together with reading the conversion results can be found in the iLLD header ***IfxEvadc_Adc.h***.

Implementation

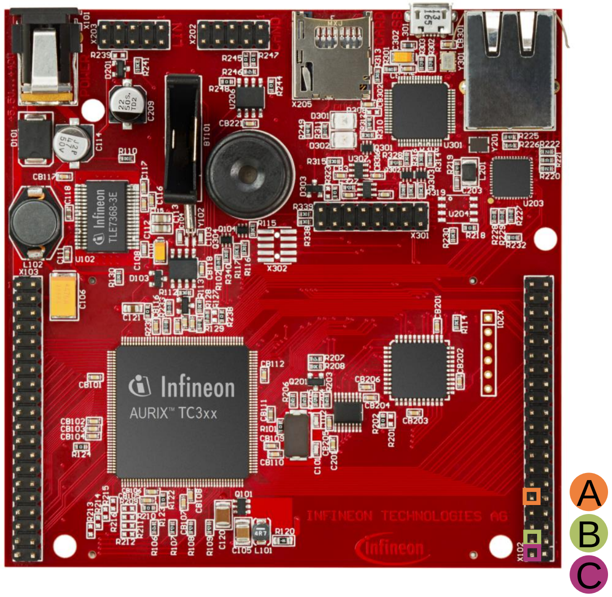
- The visualization with LEDs is done using the functions ***initializeLEDs()*** and ***indicateConversionValue()***.
  - The function ***initializeLEDs()***
    - initializes the port pins 13.0, 13.1 and 13.2 as push-pull outputs using the function ***IfxPort_setPinMode()***
    - set the port pins 13.0, 13.1 and 13.2 to high state in order to switch the LEDs off by calling the function ***IfxPort_setPinHigh()***
  - The function ***indicateConversionValue()*** is continuously executed and
    - defines an object ***conversionResult*** of the type ***Ifx_VADC_RES***
    - uses the function ***IfxVadc_Adc_getResult()*** to continuously retrieve the result value until the **valid flag** of the object ***conversionResult*** turns to high signaling that a new measurement is available
    - lights up the LED D107 (P13.0) if the discrete converted value is greater than 0xC00
    - lights up the LED D108 (P13.1) if the discrete converted value is smaller and equal than 0xC00 and greater and equal than 0x300
    - lights up the LED D109 (P13.2) if the discrete converted value is smaller than 0x300

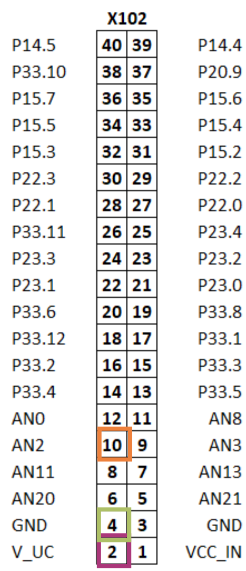
Run and Test

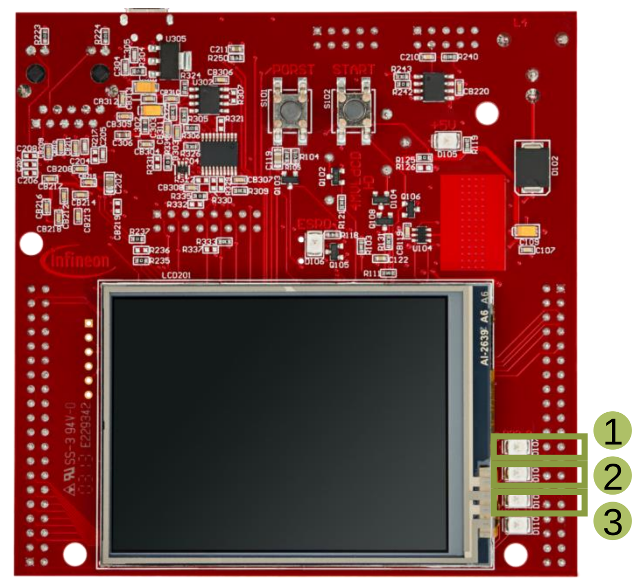
After code compilation and flashing the device, verify the behavior of the LEDs:

› Connect the pins according to the table and observe the LEDs (1), (2) and (3).

Connection	LED
leave (A) floating	(2) on, (1) (3) off *
(A) connected with (B)	(3) on, (1) (2) off
(A) connected with (C)	(1) on, (2) (3) off







*__Note:__ Depending on the surrounding noise, the floating port pin case may result in lighting up another LED (most general case is still (2)).

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

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2020-09
Published by
Infineon Technologies AG
81726 Munich, Germany

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

Document reference
ADC_Single_Channel_1_ KIT_TC397_TFT