ADC_Queued_Scan_1 for KIT_AURIX_TC397_TFT

ADC queued source

AURIX™ TC3xx Microcontroller Training
V1.0.0

Scope of work

The Enhanced Versatile Analog-to-Digital Converter (EVADC) is configured to measure multiple analog signals in a sequence using queued request.

The Queued Request of the Enhanced Versatile Analog-to-Digital Converter (EVADC) module is used to continuously scan the analog inputs channels 1, 4 and 5 of group 2.

Introduction

- The Enhanced Versatile Analog-to-Digital Converter module (EVADC) of the AURIX™ TC39x comprises 12 independent analog to digital converters (EVADC groups) with up to 16 analog input channels each.

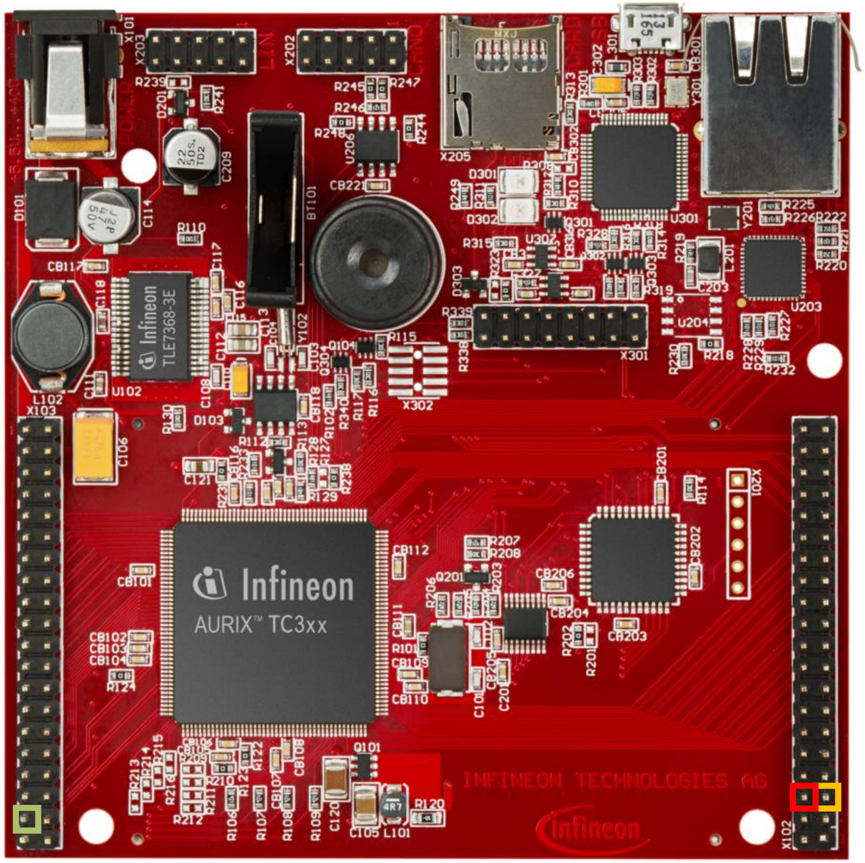
- Each channel can convert analog inputs with a resolution of up to 12-bit.

- 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.
- The signals to be measured have to be connected to channels 1, 4 and 5 of the group 2 of the VADC (pins AN17, AN20, AN21).

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

X103

VCC_IN	1	2	V_UC
GND	3	4	GND
P21.2	5	6	P21.3
P14.8	7	8	P14.7
P14.6	9	10	P20.0
P21.4	11	12	P21.5
P02.0	13	14	P02.1
P02.2	15	16	P02.3
P02.4	17	18	P02.5
P02.6	19	20	P02.7
P02.8	21	22	P00.0
P00.1	23	24	P00.2
P00.3	25	26	P00.4
P00.5	27	28	P00.6
P00.7	29	30	P00.8
P00.9	31	32	P00.10
P00.11	33	34	P00.12
AN19	35	36	AN18
AN17	37	38	AN16
AN25	39	40	AN24

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 EVADC 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***
- ***triggerConfig*** – a parameter that specify the trigger configuration

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

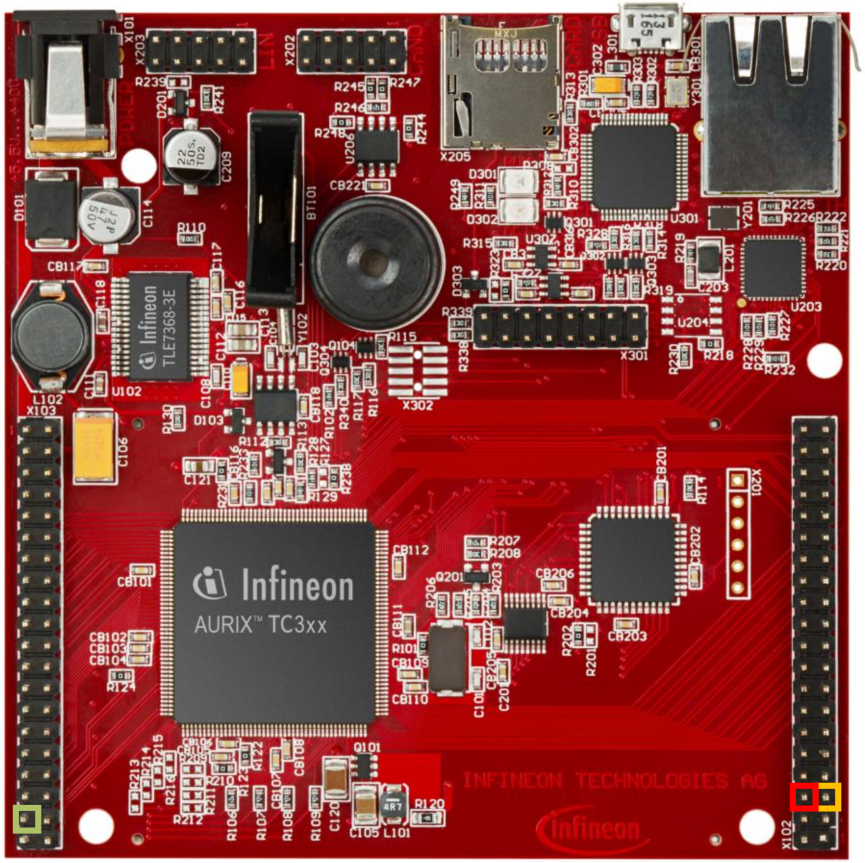
Each channel is added to the queue through the function ***IfxEvadc_Adc_addToQueue()***.

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

To read a conversion, the iLLD API ***IfxEvadc_Adc_getResult()*** 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***.

Hardware setup



The signals to be measured have to be connected to channels 1, 4 and 5 of the group 2 of the EVADC (pins AN17, AN20, AN21).

Note: For the testing purposes, connect V_UC or GND to the channels.

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

X103

VCC_IN	1	2	V_UC
GND	3	4	GND
P21.2	5	6	P21.3
P14.8	7	8	P14.7
P14.6	9	10	P20.0
P21.4	11	12	P21.5
P02.0	13	14	P02.1
P02.2	15	16	P02.3
P02.4	17	18	P02.5
P02.6	19	20	P02.7
P02.8	21	22	P00.0
P00.1	23	24	P00.2
P00.3	25	26	P00.4
P00.5	27	28	P00.6
P00.7	29	30	P00.8
P00.9	31	32	P00.10
P00.11	33	34	P00.12
AN19	35	36	AN18
AN17	37	38	AN16
AN25	39	40	AN24

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.

Run and Test

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

- Run the code and then pause it
- Repeat step number one to see that the result is changing accordingly to the signal you measure, AN17 is ***g_results[0]***, AN20 is ***g_results[1]*** and AN21 is ***g_results[2]***.

Expression	Type	Value
(x)= g_results[0].B.RESULT	unsigned short	327
(x)= g_results[1].B.RESULT	unsigned short	2
(x)= g_results[2].B.RESULT	unsigned short	4095
Add new expression		

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 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_Queued_Scan_1_KIT_TC397_TFT