CCU6_ADC_1

CCU6 ADC conversion triggering

AURIX™ TC2xx Microcontroller Training
V1.0.0

Scope of work

CCU6 timers T12 and T13 are used to trigger the Versatile Analog-to-Digital Converter (VADC).

The Timer T12 is defining the Analog-to-Digital Converter (ADC) sample rate. Due to the instability of the signals (e.g. the overshoot period), a delay is introduced by Timer T13 after each period match of Timer T12. This ensures that the measurement is done when the signal is already stable. UART communication is used to display all measured values on a terminal monitor.

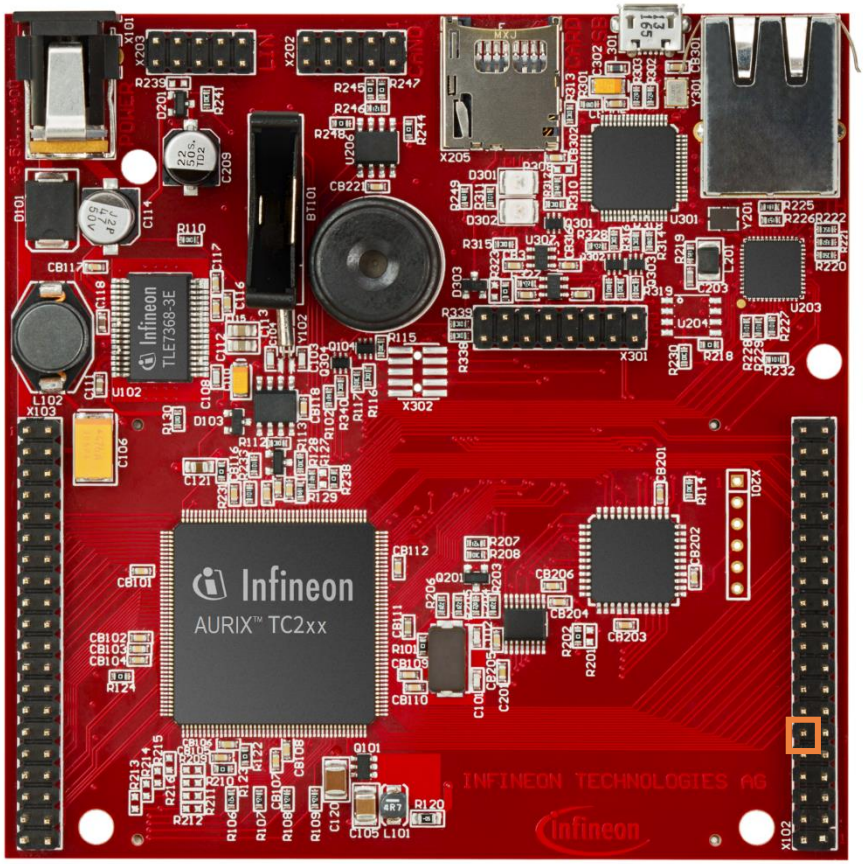
Introduction

- The Capture/Compare Unit 6 (CCU6) is a high-resolution 16-bit capture and compare unit with application specific modes, mainly used for AC drive control.
- The CCU6 unit is made up of a Timer T12 Block with three capture/compare channels and a Timer T13 Block with one compare channel. Timer T12 can also be used as a trigger for Timer T13.
- When a period match occurs, the timer can start counting again (continuous mode) or can stop and wait for another trigger (single shot mode). In this example, Timer T12 is used in continuous mode and it triggers Timer T13, which is configured in single shot mode.
- The CCU6 has four Service Request Output lines that can be used to trigger other peripherals like ADC, etc. without CPU intervention at different time intervals.

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.
- External peripherals like CCU6, GTM, etc. can trigger a sampling request.

Hardware setup



This code example has been developed for the board KIT_AURIX_TC297_TFT_BC-Step. The signal to be measured has to be connected to channel 0 of the VADC (port pin AN0).

	X102		
P14.5	40	39	P14.4
P20.10	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
AN32	8	7	AN33
AN20	6	5	AN21
GND	4	3	GND
V_UC(+5V)	2	1	VCC_IN

Implementation

Configure the CCU6 unit

Configuration of the CCU6 Timer is done in the ***init_ccu6()*** function by initializing an instance of the ***IfxCcu6_Timer_Config*** structure, which contains the following fields:

- ***timer*** – a parameter that allows to choose which of the two timers to configure. In this case, T13 is the master Timer and T12 is used as slave for its trigger.
- ***base*** – a structure that allows to set:
  - ***t12frequency*** – input clock frequency in Hz of the Timer T12
  - ***waitingTime*** – 16-bit register that determines the maximum count value for the Timer T12. It is used as value for the period register, which represents the waiting time in ticks before a new trigger event occurs for Timer T13
  - ***t13frequency*** – input clock frequency in Hz of the Timer T13
  - ***t13period*** – 16-bit register that determines the maximum count value for the Timer T13
- ***trigger*** – a structure for configuring the triggers for both timers
  - ***t13InSyncWithT12*** – allows synchronous operations between the two timers
- ***timer13*** – a structure that allows to:
  - ***t12SyncEvent*** – set the Timer T12 event that represents the Timer T13 trigger
  - ***t12SyncDirection*** – define if a trigger event is to be considered valid based on the counting direction of the Timer T12

Implementation

The functions used for CCU6 configuration are:

- ***IfxCcu6_Timer_initModuleConfig()*** – fills the configuration structure with default values
- ***IfxCcu6_Timer_initModule()*** – initializes the timer module with the user configuration
- ***IfxCcu6_enableInterrupt()*** and ***IfxCcu6_routeInterruptNode()*** – indicate which event generates the Interrupt Service Request and on which Service Request line. This enables the CCU6 to trigger an ADC conversion without the CPU intervention
- ***IfxCcu6_enableSingleShotMode()*** – configures the timer to count in single shot mode
- ***IfxCcu6_Timer_start()*** – starts the timer

The above functions can be found in the iLLD headers ***IfxCcu6_Timer.h*** and ***IfxCcu6.h***.

Implementation

Timer module frequency

In Timer Mode, the input clock of the timer module is derived from the internal module clock f_{CC6}.
By default, the internal module clock f_{CC6} is initialized by the iLLDs to run at 100 MHz.

Starting from this value, it is possible to set the timer module frequency to the following exact values:

100000000 Hz	= 100 MHz	(f_{CC6})
50000000 Hz	= 50 MHz	(f_{CC6} / 2)
25000000 Hz	= 25 MHz	(f_{CC6} / 4)
12500000 Hz	= 12.5 MHz	(f_{CC6} / 8)
6250000 Hz	= 6.25 MHz	(f_{CC6} / 16)
3125000 Hz	~ 3 MHz	(f_{CC6} / 32)
1562500 Hz	~ 1.5 MHz	(f_{CC6} / 64)
781250 Hz	~ 780 KHz	(f_{CC6} / 128)
390625 Hz	~ 390 KHz	(f_{CC6} / 256)
195312.5 Hz	~ 200 KHz	(f_{CC6} / 512)
97656.25 Hz	~ 100 KHz	(f_{CC6} / 1024)
48828.12 Hz	~ 50 KHz	(f_{CC6} / 2048)
24414.06 Hz	~ 25 KHz	(f_{CC6} / 4096)
12207.03 Hz	~ 12.5 KHz	(f_{CC6} / 8192)
6103.51 Hz	~ 6 KHz	(f_{CC6} / 16384)
3051.75 Hz	~ 3 KHz	(f_{CC6} / 32768)

Note: Any value can be set as ***frequency*** parameter, but the software will round up the chosen value to the nearest higher frequency listed in the above table (e.g. setting ***timerConfig.base.t12Frequency = 400000*** the timer will run at 781250 Hz).

Implementation

Period Match Frequency calculation

To calculate the actual Period Match frequency f_{PM} (the number of occurrences per second in which the timer counter value reaches the period value and is reset), two parameters have to be considered: **Timer clock frequency** and its **period**.

- The **Timer clock frequency** can be freely selected as described in the previous slide
- The **period** value is stored in a 16-bit register, which limits its maximum value to 65535

The Period Match frequency and period can then be calculated as:

$$f_{PM} = (\textbf{Timer clock frequency}) / (\textbf{period} + 1)$$
$$T_{PM} = (\textbf{period} + 1) / (\textbf{Timer clock frequency})$$

In this example, a clock frequency equal to 48828 Hz is chosen for both timers, while the periods are set to have a T12 period match frequency equal to 1 Hz (ADC sample rate, 1 sample/sec) and a T13 period match (delay) time equal to 0.5 seconds (ADC sample delay).

Implementation

Configuration of the VADC

The configuration of the VADC is done in the ***init_vadc()*** function in three different steps:

- Configuration of the **VADC module**
- Configuration of the **VADC group**
- Configuration of the **VADC channel**

Configuration of the VADC module

The functions used for configuring the VADC module are:

- ***IfxVadc_Adc_initModuleConfig()*** – initializes the VADC module configuration structure with the default values
- ***IfxVadc_Adc_initModule()*** – initializes the VADC module with the user configuration, which in this case is the default configuration

Implementation

Configuration of the VADC group

Configuration of the VADC group is done by initializing an instance of the ***IfxVadc_Adc_GroupConfig*** structure, which contains the following fields:

- ***groupId*** – a parameter that allows to choose which of the groups to configure
- ***master*** – indicates which group is the master. In this example only one channel of one group is used, therefore the chosen group is also the master
- ***arbiter*** – a structure that represents the enabled request sources (which can be Scan, Queue and/or Background sources; in this example the Scan source is used)
- ***scanRequest*** – a structure to set the Scan Request Source configuration
  - ***autoscanEnabled*** – specifies if Autoscan mode is active
  - ***triggerConfig*** – specifies the trigger configuration

The functions used for configuring the VADC group are:

- ***IfxVadc_Adc_initGroupConfig()*** – fills the group configuration structure with default values
- ***IfxVadc_Adc_initGroup()*** – initializes the VADC group specified in the parameters with the user configuration

Implementation

Configuration of the VADC channel

Configuration of the VADC channel is done by initializing an instance of the ***IfxVadc_Adc_ChannelConfig*** structure, which contains the following fields:

- ***channelId*** – a parameter that allows to choose which of the channels to configure
- ***resultRegister*** – indicates the register where the sample value is stored
- ***resultPriority*** – the priority of the result trigger interrupt
- ***resultServProvider*** – interrupt service provider for the result trigger interrupt. This can be any of the available CPUs or the DMA. In this example, the interrupt is used to print the measurements through UART communication

The functions used for configuring the VADC channel are:

- ***IfxVadc_Adc_initChannelConfig()*** – fills the channel configuration structure with default values
- ***IfxVadc_Adc_initChannel()*** – initializes the channel with the user configuration
- ***IfxVadc_Adc_setScan()*** – add the channel used for the measurement to the scan sequence

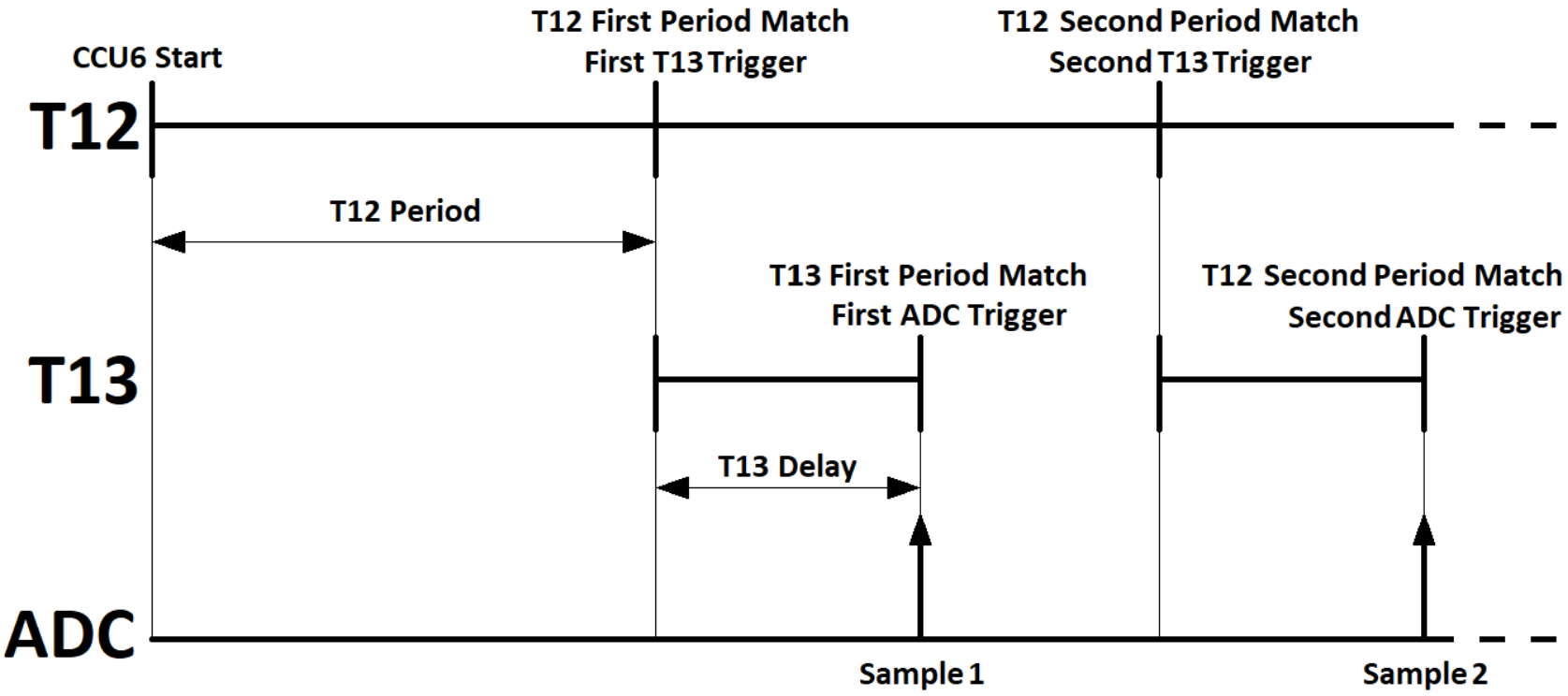
When both the CCU6 and VADC modules are configured, a single scan sequence is triggered by the CCU6 timer T13 period match event.

All the functions used for configuring the VADC module, group and channel can be found in the iLLD header ***IfxVadc_Adc.h***.

Implementation

Events timing

The graph shows the temporal evolution of events: T12 period matches, T13 period matches and ADC sampling instants.



Implementation

Configuration of the UART

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

The ***init_uart()*** function initializes the UART communication.

The iLLD function ***IfxAsclin_Asc_initModuleConfig()*** fills the configuration structure ***ascConf*** with the default values. Then, the parameters used to configure the module are set, depending on the needed connection: baudrate, interrupts, Tx and Rx buffers and port pins configuration.

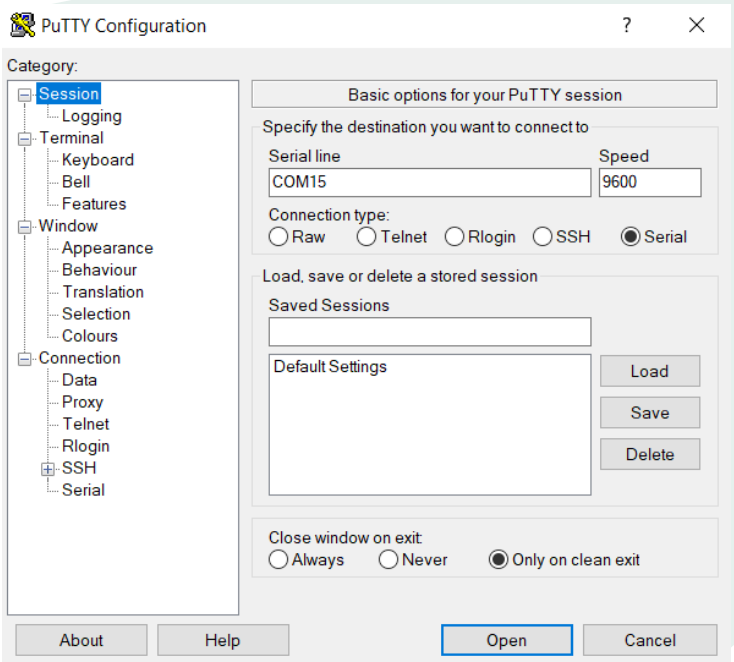
Finally, ***IfxAsclin_Asc_initModule()*** initializes the module with the user configuration and ***IfxAsclin_Asc_stdIfDPipeInit()*** initializes the standard interface to use the ASC module.

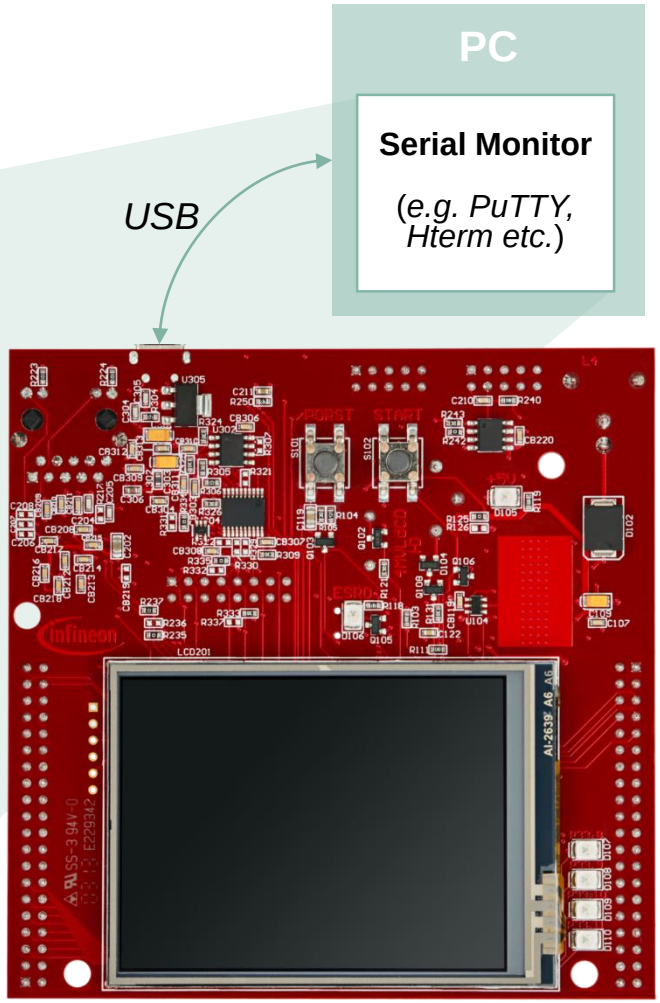
The above functions can be found in the iLLD header ***IfxAsclin_Asc.h***.

Software setup

Serial monitor settings:

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





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 (can be seen in the Device Manager)
- Check the ADC conversions sampled every one second in the serial monitor console

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 2020-01
Published by
Infineon Technologies AG
81726 Munich, Germany

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

Document reference
CCU6_ADC_1

IMPORTANT NOTICE
The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics ("Beschaffenheitsgarantie") .

With respect to any examples, hints or any typical values stated herein and/or any information regarding the application of the product, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation warranties of non-infringement of intellectual property rights of any third party.

In addition, any information given in this document is subject to customer's compliance with its obligations stated in this document and any applicable legal requirements, norms and standards concerning customer's products and any use of the product of Infineon Technologies in customer's applications.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office (www.infineon.com).

WARNINGS
Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.