

# Interrupt\_Prio\_1 for KIT\_AURIX\_TC275\_LK

## Interrupts prioritization

AURIX™ TC2xx Microcontroller Training  
V1.0.1



[Please read the Important Notice and Warnings at the end of this document](#)

## Scope of work

---

**Three interrupts, with different priorities each, send a message on a terminal using UART communication.**

Each interrupt sends a message on a terminal when the ISR starts and ends. In this example, the interrupts are triggered by GPT12 module.

# Introduction

---

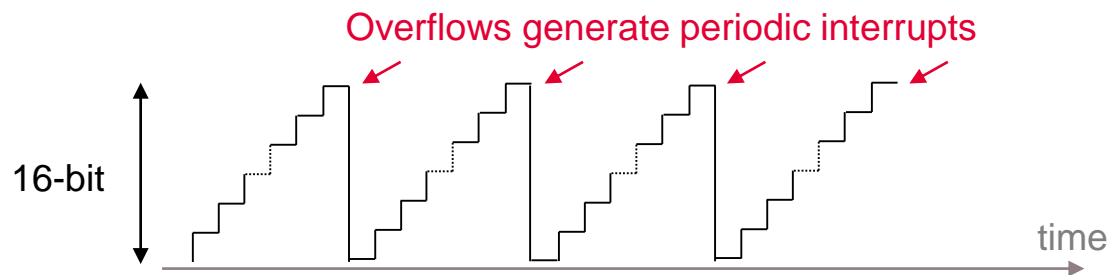
- › The interrupt system in the AURIX™ TC2xx devices is implemented in the **Interrupt Router (IR)**
  
- › **Interrupt Requests** (or Service Requests) can be serviced either by the CPUs or by the DMA module (both called Service Providers)
  
- › An interrupt can be triggered by:
  - Each module connected to the IR
  - External peripherals
  - Software via General Purpose Service Requests (GPSR)
  
- › Each Service Provider supports **up to 255** service priority levels:
  - 0 to disable the interrupt
  - 255 for highest priority

# Introduction

- › A triggered interrupt can be followed by an Interrupt Service Routine (ISR), a function which is called every time an interrupt is triggered
  
- › Example of ISR configuration:
  - Assign the ISR to a service provider and an interrupt priority  
***IFX\_INTERRUPT(functionA, 0, ISR\_Priority);***
  - ISR implementation  
***void functionA(void)***  
***{***  
***[...]***  
***}***
  
- › By default, an ISR cannot be interrupted by any other interrupt. IR waits until the function is finished before servicing any pending interrupt
  
- › To allow interrupting the execution of ISRs by higher priority service requests, the following iLLD function must be added at the beginning of the ISR:  
***IfxCpu\_enableInterrupts();***

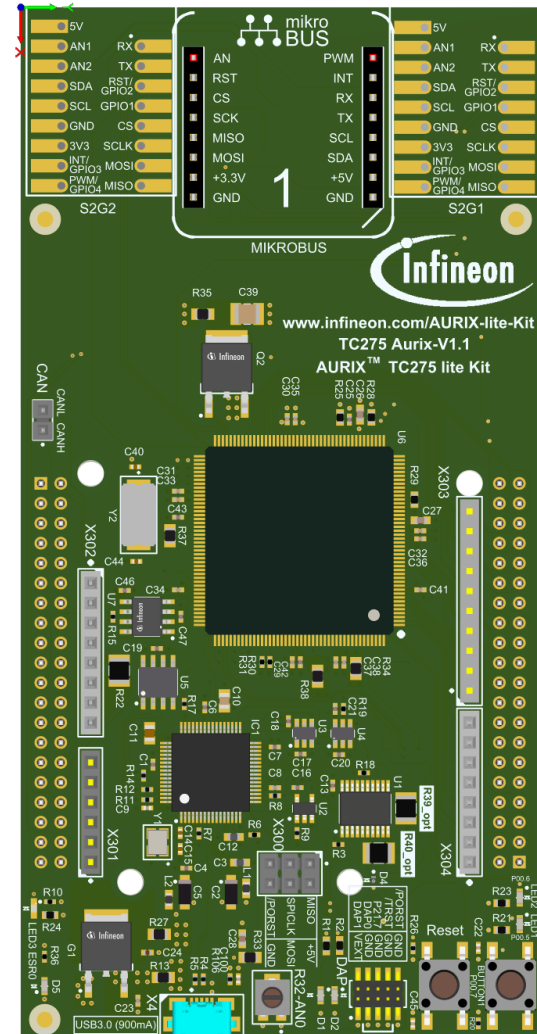
# Introduction

- › The **General Purpose Timer (GPT12)** module has very flexible multifunctional timer structures which can be used for timing, event counting, pulse width measurement, pulse generation, frequency multiplication, and other purposes
- › The GPT12 module incorporates **five 16-bit timers** that are grouped into two timer blocks GPT1 and GPT2. Each timer can operate independently in a number of different modes such as Timer mode, Gated Timer mode, Counter mode, or can be concatenated with another timer of the same block
- › In this example, the timers are used in **timer mode**:



# Hardware setup

This code example has been developed for the board `KIT_AURIX_TC275_LITE`.



# Implementation

## Configure the Timers

Three timers are configured in order to periodically generate three different interrupts. The GPT12 module methods come from the iLLD header ***IfxGPT12.h***.

First, the module is enabled via ***IfxGpt12\_enableModule()***.

All three timers are set in **timer mode** using the iLLD function: ***IfxGpt12\_Tx\_setMode()***, where  $x = 1,2,3$ .

To improve the observability during the process of interrupts handling, the timers are set to the slowest frequency. This is done via the prescalers of both the GPT1 block and the individual timers:

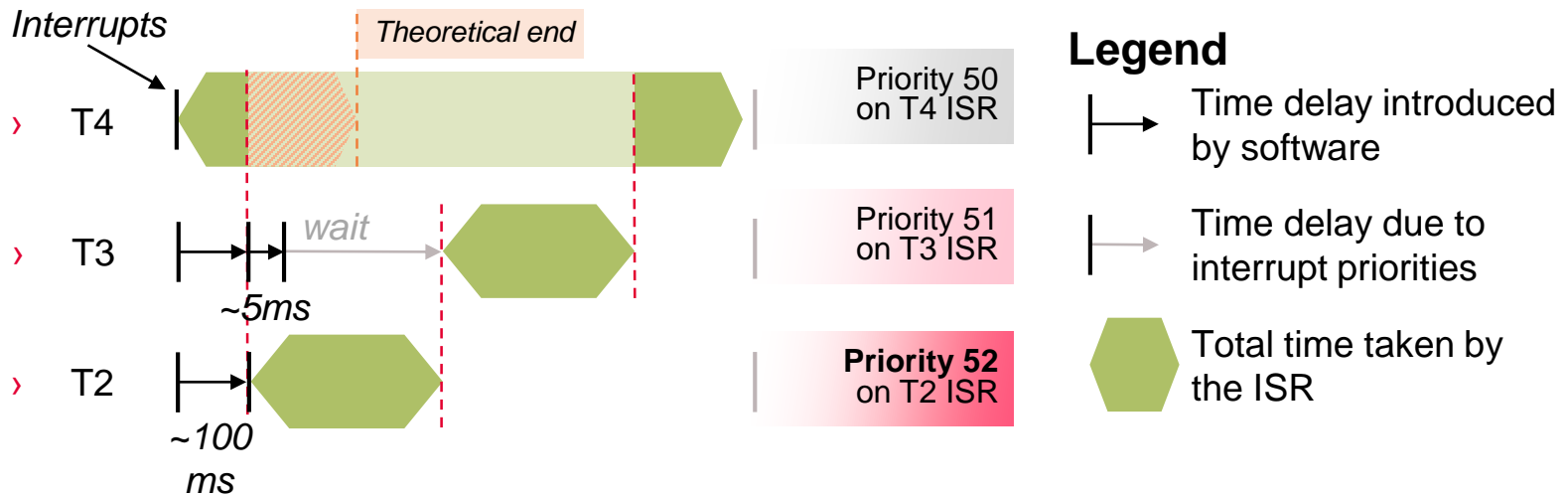
- › ***IfxGpt12\_setGpt1BlockPrescaler()***
  - with parameter ***IfxGpt12\_Gpt1BlockPrescaler\_32*** for  $f_{GPT}/32$
- › ***IfxGpt12\_Tx\_setTimerPrescaler()***
  - with parameter ***IfxGpt12\_TimerInputPrescaler\_128*** for  $f_{GPT1}/128$

$$f_{GPT} = 100\text{MHz} \rightarrow f_{T2} = f_{T3} = f_{T4} = \underline{24.4\text{kHz}}$$

$$T_{T2} = T_{T3} = T_{T4} = \underline{2.684\text{s}}$$

# Implementation

- > The timer with the lowest ISR priority, T4, is launched first
- > The timers T2 and T3 are launched shortly after in order to ensure that the lowest priority ISR has already started
- > All timers are running with the same frequency



- > All cases are covered:
  - Higher prioritized ISR **interrupts** lower prioritized ISR already running
  - Lower prioritized ISR **waits** the end of higher prioritized ISR in case of simultaneous or belated trigger



# Implementation

## Configure the ASCLIN

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

Configuration of the ASCLIN module for UART communication is done in the setup phase by initializing an instance of the ***IfxAsclin\_Asc\_Config*** structure with the following parameters:

- › ***baudrate*** – structure to set the actual communication speed in bit/s
- › ***interrupt*** – structure to set:
  - interrupt priorities for transmit, receive and error events (***txPriority***, ***rxPriority*** and ***erPriority***)
  - ***typeOfService*** – defines which service provider is responsible for handling the interrupt, which can be any of the available CPUs, or the DMA
- › ***pins*** – structure to set which GPIOs port pins are used for the communication
- › ***rxBuffer***, ***rxBufferSize***, ***txBuffer***, ***txBufferSize*** – to configure the buffers that will hold the incoming/outgoing data

The function ***IfxAsclin\_Asc\_initModuleConfig()*** fills the configuration structure with default values and ***IfxAsclin\_Asc\_initModule()*** initializes the module with the user configuration.

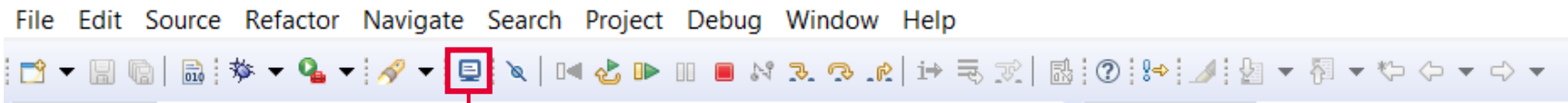
The standard interface is configured with the function ***IfxAsclin\_Asc\_stdIfDPipelnit()***.

All the above functions can be found in the iLLD header ***IfxAsclin\_Asc.h***.

**Note:** Tx, Rx and Er interrupts have to be higher priority than GPT interrupts for a correct display.

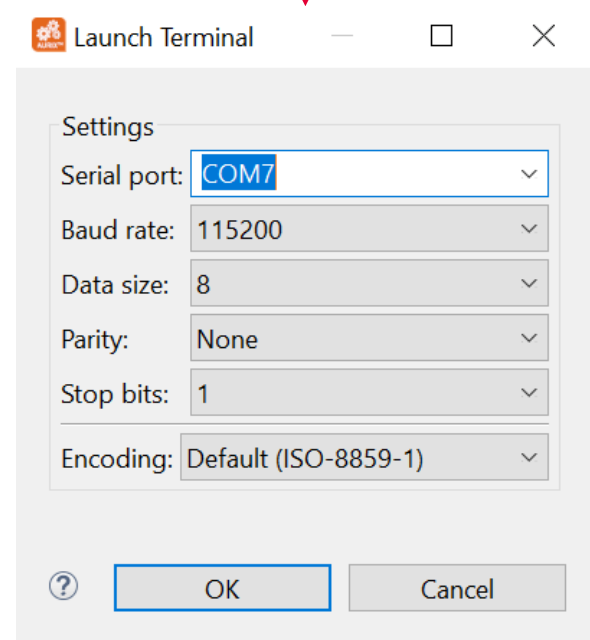
# Run and Test

- > For this training, a serial monitor is required for visualizing the interrupt prioritization process. The monitor can be opened inside the AURIX™ Development Studio using the following icon:



- > The serial monitor must be configured with the following parameters to enable the communication between the board and the PC:

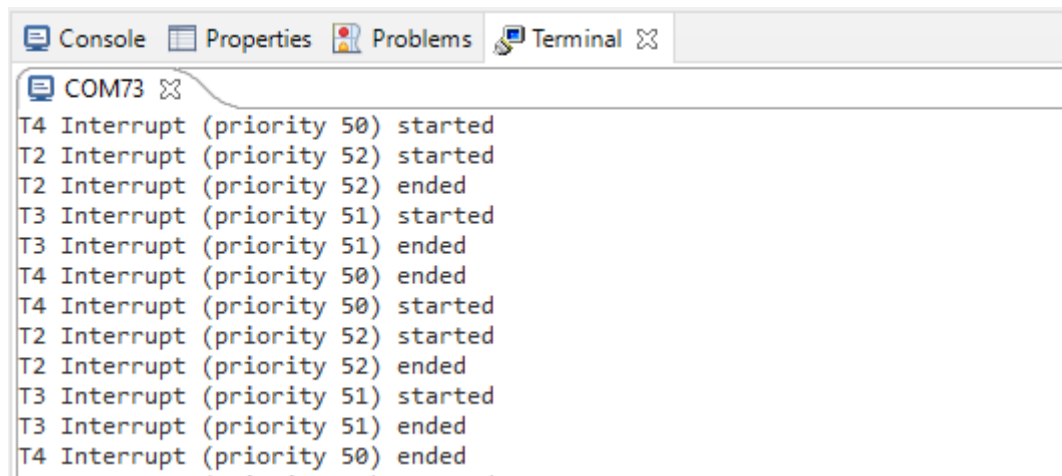
- Speed (baud): 115200
- Data bits: 8
- Stop bit: 1



# Run and Test

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

- › The board must be connected to PC via the USB cable
- › Open a serial monitor with the previous configuration and connect
- › In the terminal, it can be observed when an interrupt starts and ends



```

Console Properties Problems Terminal
COM73
T4 Interrupt (priority 50) started
T2 Interrupt (priority 52) started
T2 Interrupt (priority 52) ended
T3 Interrupt (priority 51) started
T3 Interrupt (priority 51) ended
T4 Interrupt (priority 50) ended
T4 Interrupt (priority 50) started
T2 Interrupt (priority 52) started
T2 Interrupt (priority 52) ended
T3 Interrupt (priority 51) started
T3 Interrupt (priority 51) ended
T4 Interrupt (priority 50) ended

```

**Note:** Interrupts with higher priority suspend the lower priority interrupts. The latter continue after the end of the higher priority interrupts.

# 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](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>

# Revision history

---

<b>Revision</b>	<b>Description of change</b>
V1.0.1	Fixed timeline image
V1.0.0	Initial version

## Trademarks

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

**Edition 2023-03**

**Published by**

**Infineon Technologies AG  
81726 Munich, Germany**

**© 2023 Infineon Technologies AG.  
All Rights Reserved.**

**Do you have a question about this document?**

**Email: [erratum@infineon.com](mailto:erratum@infineon.com)**

**Document reference**

**Interrupt\_Prio\_1\_KIT\_TC275\_LK**

## 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](http://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.