GPT12_Timer_Interrupt_1 for KIT_AURIX_TC397_TFT

GPT12 Timer Interrupt

AURIX™ TC3xx Microcontroller Training
V1.0.2

Scope of work

The GPT12 module generates an interrupt each 500 ms and toggles an LED.

The timer T3 of the GPT12 module is configured to trigger an interrupt each 500 ms. In its interrupt service routine, an LED is toggled.

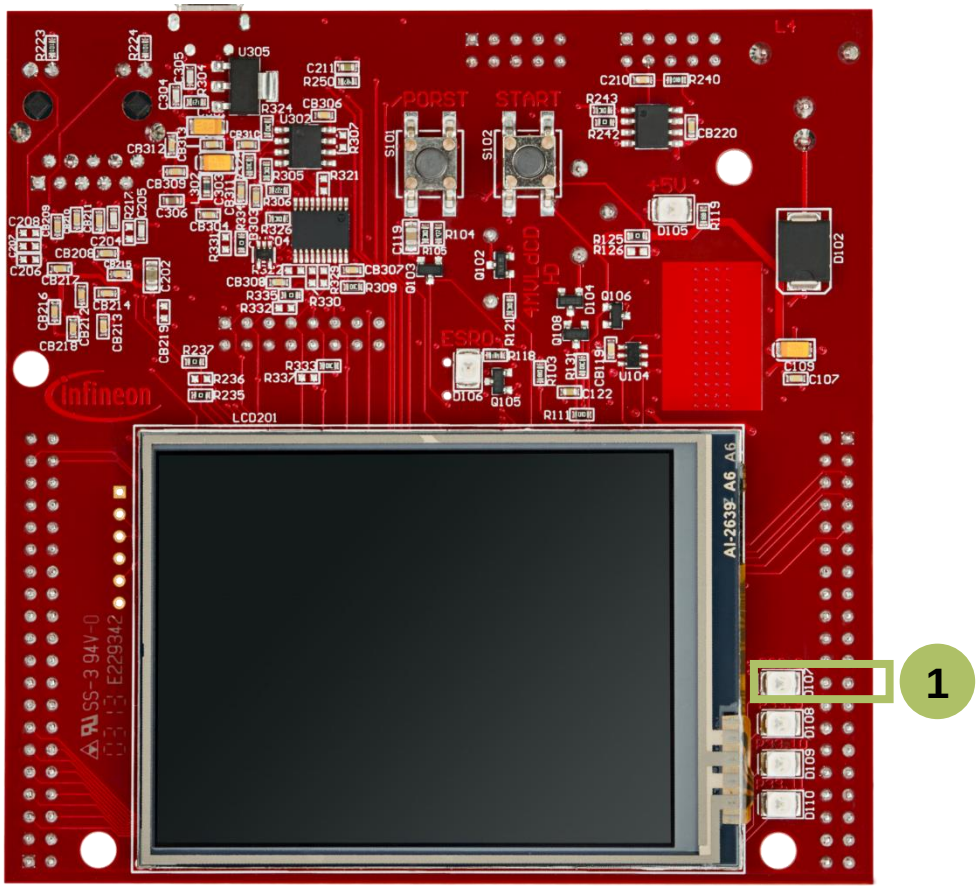
Introduction

- The General Purpose Timer Unit (GPT12) consists of two GPT blocks (GPT1 and GPT2)
- Each block has a multifunctional timer structure which incorporates several 16-bit timers
- Block GPT1 contains three timers: The core timer T3 and two auxiliary timers T2 and T4
- All timers of block GPT1 can run in one of four modes: Timer Mode, Gated Timer Mode, Counter Mode or Incremental Interface Mode
- In this example, the timer T3 is used in timer mode. The count direction is configured to “down-counting”. On an underflow event of timer T3 the value of timer T2 is transferred into timer T3

Hardware setup

This code example has been developed for the board KIT_A2G_TC397_5V_TFT.

LED D107 (1) is used for this example.



Implementation

Configuring GPT12 Module

Configuration of the GPT12 is done once in the function ***initGpt12Timer()*** by the following steps:

- Enable GPT12 module by calling the iLLD function ***IfxGpt12_enableModule()***
- Set the GPT1 prescaler with the iLLD function ***IfxGpt12_setGpt1BlockPrescaler()***
- Configure the mode of timer T3 by using the iLLD function ***IfxGpt12_T3_setMode()***
- Set the counting direction of the timer T3 by calling the iLLD function ***IfxGpt12_T3_setTimerDirection()***
- Set the prescaler of timer T3 by calling the iLLD function ***IfxGpt12_T3_setTimerPrescaler()***
- Set the timer value with the iLLD function ***IfxGpt12_T3_setTimerValue()*** and use an ***uint16*** number as parameter
- Start the timer T3 by calling the iLLD function ***IfxGpt12_T3_run()***

The above configuration functions are provided by the iLLD header ***IfxGpt12.h***.

Implementation

Configuring timer T2 to reload timer T3

Configuration of the timer T2 is done once in the function ***initGpt12Timer()*** by the following steps:

- Configure the timer T2 in reload mode by using the iLLD function ***IfxGpt12_T2_setMode()***
- Set the trigger event for reload of timer T3 by using the iLLD function ***IfxGpt12_T2_setReloadInputMode()***
- Set an ***uint16*** reload value with the iLLD function ***IfxGpt12_T2_setTimerValue()*** (the register has 16-bit length)

The above configuration functions are provided by the iLLD header ***IfxGpt12.h***.

Implementation

Reload Value calculation example

Calculate the reload value to adjust to a period of 500 ms. This value needs to fit into 16-bit. In order to get a high resolution, the prescaler values should be as low as possible.

- GPT12 module base frequency f_{GPT} = 100 MHz
- 500 ms period corresponds to a frequency f of 2 Hz
- The GPT1 block prescaler is set to ***IfxGpt12_Gpt1BlockPrescaler_16***, therefore f_{GPT} is divided by **16**
- The timer prescaler is set to ***IfxGpt12_TimerInputPrescaler_64***, therefore f_{GPT} is divided by **64**

$$\text{Reload Value} = \frac{f_{GPT}}{\textbf{Gpt1BlockPrescaler} * \textbf{TimerInputPrescaler} * f}$$

$$\text{Reload Value} = \frac{100\ MHz}{16\ * 64\ * 2\ Hz} = 48828$$

Implementation

Configuring the interrupt for GPT12

Configuration of the interrupt is done once in the function ***initGpt12Timer()*** by the following steps:

- Get the address of timer T3 service request with the iLLD function ***IfxGpt12_T3_getSrc()***

- Initialize GPT12 interrupt by calling ***IfxSrc_init()*** with a pointer to the address of the timer T3 service request, the interrupt provider and the interrupt priority number as parameters

- Enable GPT12 interrupt with the iLLD function ***IfxSrc_enable()*** and the pointer to the address of timer T3 service request as parameter

The function ***IfxGpt12_T3_getSrc()*** is provided by the header ***IfxGpt12.h*** and the functions ***IfxSrc_init()*** and ***IfxSrc_enable()*** are provided by the header ***IfxSrc.h***.

Implementation

Configuring the LED

The LED is configured and toggled by controlling the port pin to which it is connected.

In the setup phase, the port pin is configured as **output push-pull mode** using the function ***IfxPort_setPinModeOutput()***.

The Interrupt Service Routine (ISR)

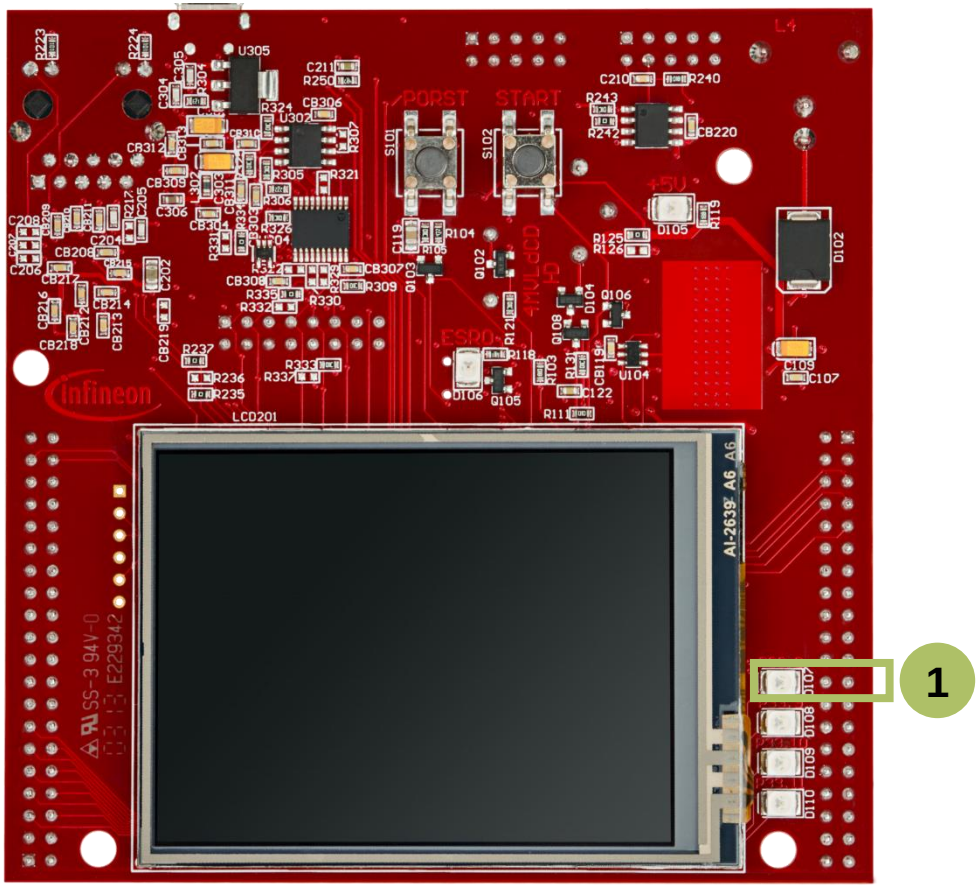
The ISR implemented in this example contains the following:

- Toggle the LED by calling the iLLD function ***IfxPort_togglePin()***

Both functions are provided by the iLLD header ***IfxPort.h***.

Run and Test

After code compilation and flashing the device, observe the **LED D107** (1), which should be blinking with a frequency of 1 Hz.



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

Revision history

Revision	Description of change
V1.0.2	Fixed function name to configure the port pin
V1.0.1	Update of version to be in line with the code example’s version
V1.0.0	Initial version

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

Edition 2021-06
Published by
Infineon Technologies AG
81726 Munich, Germany

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

Document reference
GPT12_Timer_Interrupt_1
_KIT_TC397_TFT

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.