MULTICAN_1
for KIT_AURIX_TC297_TFT
MULTICAN data transmission

AURIX™ TC2xx Microcontroller Training
V1.0.2

Scope of work

MULTICAN is used to exchange data between two nodes, implemented in the same device using Loop-Back mode.

A CAN message is sent from CAN node 0 to CAN node 1 using Loop-Back mode. After the CAN message transmission, an interrupt will be generated and LED1 will be turned on to confirm successful message transmission. Once the CAN message is successfully received by the CAN node 1, an interrupt will be generated. Inside the interrupt service routine the content of the received CAN message will be compared to the content of the transmitted CAN message. In case of a success, LED2 will be turned on to confirm successful message reception.

Introduction

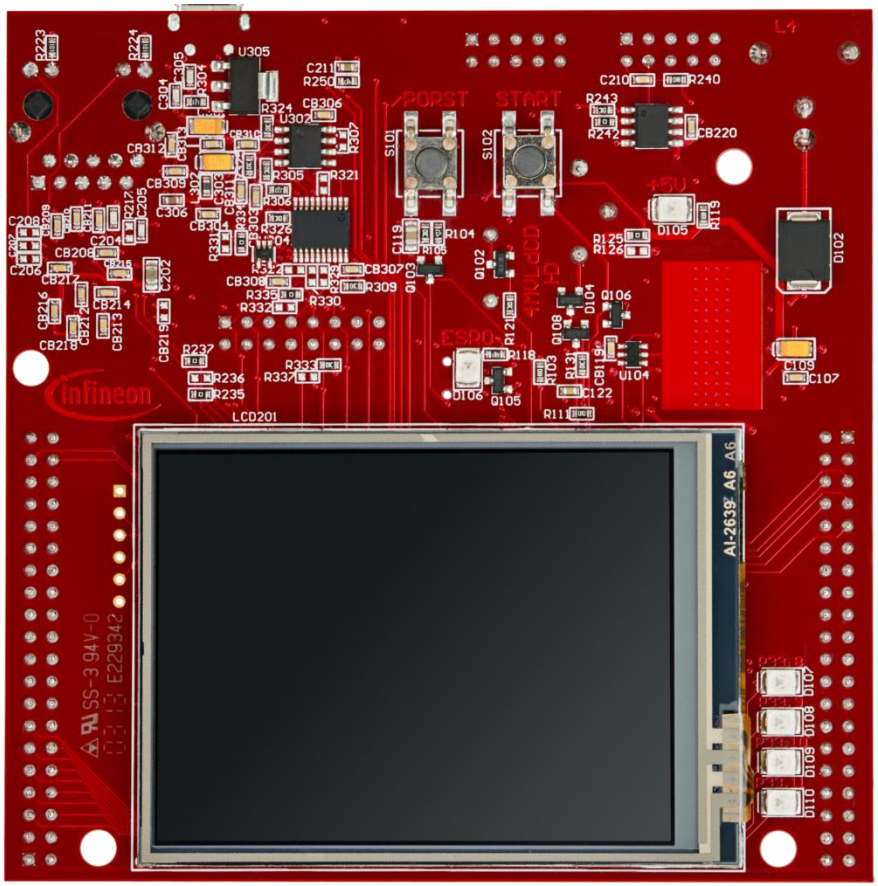
- The MultiCAN+ module provides a communication interface which is **fully compliant with CAN specification V2.0B (active)** and to **CAN FD ISO11898-1 DIS version 2014**, providing communication up to **1 Mbit/s in Classical CAN** (ISO 11898-1:2003(E)mode) and/or **CAN FD up to 5 Mbit/s** (dependent on frequency and nodes).

- The MultiCAN+ module consists of several **CAN nodes** (in case of AURIX™ TC29x device, 4 nodes) which are **CAN FD capable**. Each CAN node communicates over two pins (TXD and RXD). Additionally, there is an internal **Loop-Back Mode** functionality available for test purposes.

- All CAN nodes share a common set of 256 **message objects**. Each message object can be individually allocated to one of the CAN nodes. Besides serving as a **storage container for incoming and outgoing frames**, message objects can be combined to build **gateways** between the CAN nodes or to setup a **FIFO buffer**.

Hardware setup

This code example has been developed for the board KIT_AURIX_TC297_TFT_BC-Step.



Implementation

Application code can be separated into three segments:

- Initialization of the MultiCAN+ module with the accompanying node and message objects initialization, implemented in the ***initMultican()*** function.
- Initialization of the pins that are connected to the LEDs. LEDs will be used to verify the success of a CAN message transmission and reception. This is done inside the ***initLed()*** function.
- Transmission of the configured CAN message, implemented in the ***transmitCanMessage()*** function.

Additionally, two interrupt service routines (ISRs) are implemented:

- On TX interrupt the LED1 is turned on to indicate successful CAN message transmission (implemented in ***canIsrTxHandler()***).
- On RX interrupt the ISR verifies the received CAN message and turns on the LED2 to indicate successful reception (implemented in ***canIsrRxHandler()***).

Implementation

MultiCAN+ module initialization

Initialization is performed in three phases:

- A default CAN module configuration is loaded into the configuration structure by using the function ***IfxMultican_Can_initModuleConfig()***.
  Afterwards, the initialization of the CAN module with the user configuration is done with the function ***IfxMultican_Can_initModule()***.

- A default CAN node configuration is loaded into the configuration structure by using the function ***IfxMultican_Can_Node_initConfig()***. Initialization of the CAN nodes (CAN node 0 and 1) with the different CAN node ID values and definition of Loop-Back Mode usage for both nodes is done with the function ***IfxMultican_Can_Node_init()***. CAN node 0 is defined as “source node” while CAN node 1 represents a “destination node”.

Implementation

MultiCAN+ module initialization

- A default CAN message object configuration is loaded into the configuration structure by using the function ***IfxMultican_Can_MsgObj_initConfig()***.
  The initialization of the CAN message objects (CAN message object 0 and 1) with different CAN message object ID value and different frame direction is done afterwards. CAN message object 0 is defined as „transmit message object“ while CAN message object 1 is defined as „receive message object“. Additionally, generation of TX and RX interrupt respectively is enabled and configured (***IfxMultican_Can_MsgObj_init()*** function).

All functions used for the MultiCAN+ module initialization are declared in the iLLD header ***IfxMultican_Can.h***.

Implementation

Initialization of the pins connected to the LEDs

LEDs are used to verify the success of a CAN message transmission and reception. Before using the LEDs, the port pins to which the LEDs are connected must be configured.

- First step is to set the port pins to level “HIGH”; this keeps the LEDs turned off as a default state (***IfxPort_setPinHigh()*** function).
- Second step is to set the port pins to push-pull output mode with the ***IfxPort_setPinModeOutput()*** function.
- Finally, the pad driver strength is defined through the function ***IfxPort_setPinPadDriver()***.

All functions are declared in the iLLD header ***IfxPort.h***.

Implementation

CAN message transmission

Before a CAN message is transmitted, two messages need to be initialized. TX message (message that will be transmitted) is initialized with the predefined content. RX message (message where the received CAN message will be stored) is initialized with some invalid data (after successful CAN transmission the data will be replaced with the valid data).

- Initialization of both TX and RX messages is done by using ***IfxMultican_Message_init()***.
- A CAN message is transmitted by using the ***IfxMultican_Can_MsgObj_sendMessage()*** function. A CAN message will be continuously transmitted as long as the returned status is ***IfxMultican_Status_notSentBusy*** (this status occurs if there is a pending transmit request).

The function ***IfxMultican_Message_init()*** is declared in the iLLD header ***IfxMultican.h*** while the ***IfxMultican_Can_MsgObj_sendMessage()*** function is declared in the iLLD header ***IfxMultican_Can.h***.

Implementation

Interrupt Service Routines (ISRs)

Two interrupt services routines are implemented: one ISR that is triggered with the successful CAN message transmission and a second one that is triggered with the successful CAN message reception.

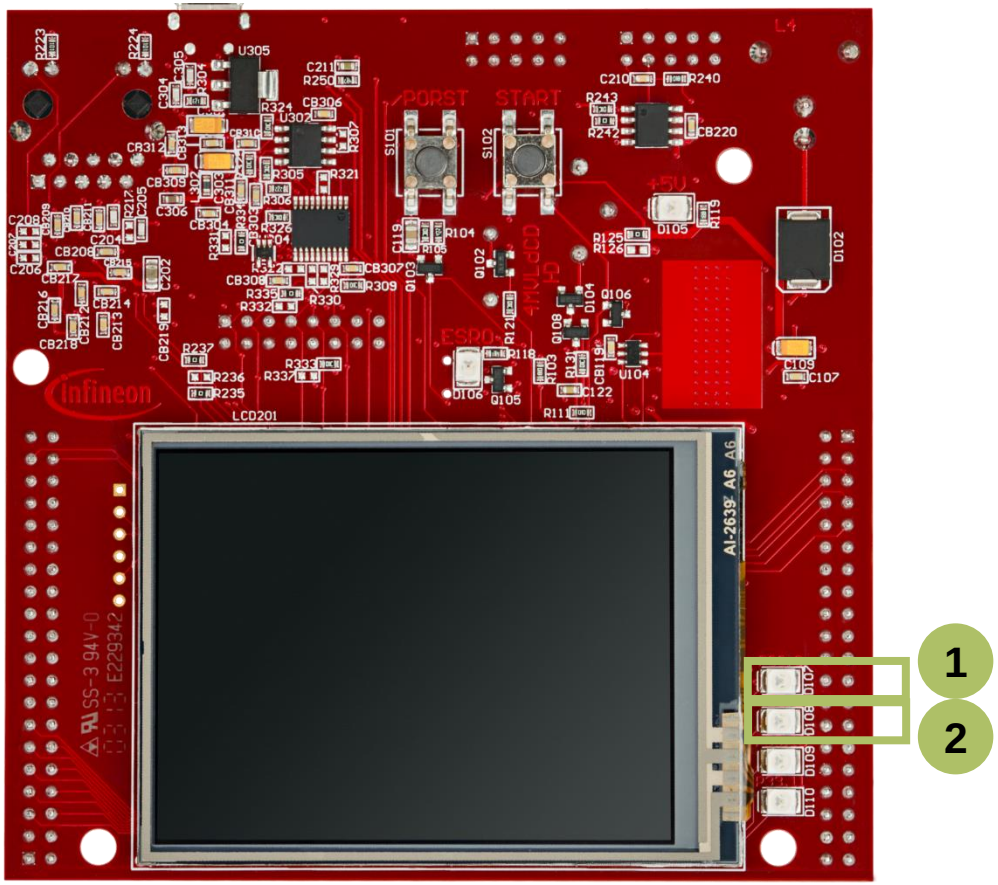
- TX interrupt service routine indicates that the CAN message has been transmitted successfully by turning on LED1.
- RX interrupt service routine reads the received CAN message with the ***IfxMultican_Can_MsgObj_readMessage()*** function. Based on the return status, the code execution can end up in an infinite loop due to the erroneous return status. If the non-erroneous return status is present, then a received data is compared against the transmitted data. In case of success, the LED2 is turned on to indicate that the received message is correct.

The function is declared in the iLLD header ***IfxMultican_Can.h***.

Run and Test

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

- Check that LED1 D107 (1) is turned on (successful CAN message transmission by CAN node 0)
- Check that LED2 D108 (2) is turned on (successful CAN message reception by CAN node 1)



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	Update of version to be in line with the code example’s version
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 2020-12
Published by
Infineon Technologies AG
81726 Munich, Germany

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

Document reference
MULTICAN_1_KIT_TC297_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.