AN2824
Application note

STM32F101xx and STM32F103xx medium- and high-density devices: advanced I²C examples

Introduction

The aim of this application note is to provide examples that cover the four I^2C communication modes in which STM32F10xxx microcontrollers can operate, that is, slave transmitter, slave receiver, master transmitter and master receiver.

This application note applies to STM32F101xx and STM32F103xx medium- and high-density microcontrollers that feature two I^2C interfaces (controllers). In the rest of the document, these devices will be referred to as STM32F10xxx.

The application note is split into two parts. In the first part, the examples are focussed on the basic configuration of the four modes (data transfer involving the I^2C interface only). The second part gives practical and advanced examples in a “real” application (involving other STM32F10xxx-controlled resources: multislave communication, ADC conversion, data display, power modes, temperature sensor, etc.).

These examples are implemented with interrupts and bus error management.

Contents

List of figures

1 Example 1

Note: Please keep in mind that this application note applies to STM32F101xx and STM32F103xx medium- and high-density microcontrollers that feature two I^2C interfaces (controllers). In the rest of the document, these devices will be referred to as STM32F10xxx for convenience.

1.1 Overview

In this example, an I^2C communication is set up between two STM32F10xxx devices. The first device is operating as a master transmitter/receiver and the second one, as a slave transmitter/receiver.

Table 1: Interrupt events covered by example 1 gives the list of the interrupt events covered by this example.

In this example, “STM32F10xxx I2C1” or “I2C1” is used to refer to the I2C1 interface of the device while “STM32F10xxx I2C2” or “I2C2” refers to the I2C2 interface.

For more details, please refer to the medium- and high-density STM32F101xx and STM32F103xx datasheets and reference manual available from: *http://www.st.com/stm32*.

Table 1. Interrupt events covered by example 1

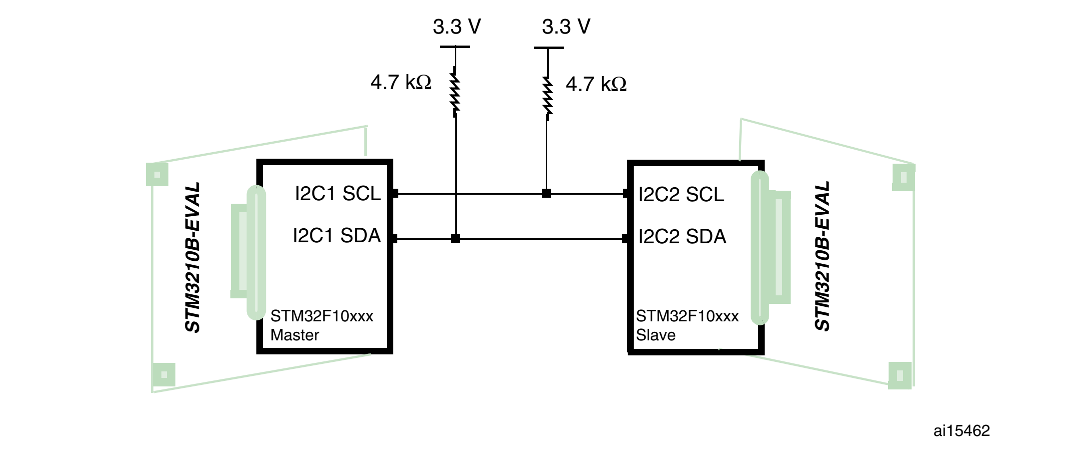
Interrupt event	Enable control bit
Start bit sent (Master)	ITEVFEN
Address sent (Master) or address matched (Slave)	
Stop received (Slave)	
Data byte transfer finished	
Receive buffer not empty	ITEVFEN and ITBUFEN
Transmit buffer empty	
Bus error	ITERREN
Acknowledge failure	

1.2 Hardware environment

Figure 1 shows the hardware connection between the I2C1 of an STM32F10xxx and the I2C2 of another STM32F10xxx. The I2C1 and I2C2 data lines (SDA) are connected together. The I2C1 and I2C2 clock (SCL) lines are also connected together. A pull-up resistor is connected to each line (I2C1 SDA and I2C1 SCL).

There are no specific boards (single printed-circuit board) fitted with two STM32F10xxx devices, so we simply connected two STM3210B-EVAL boards together as shown in *Figure 1: Hardware connection.*

Figure 1. Hardware connection



1.3 Example description

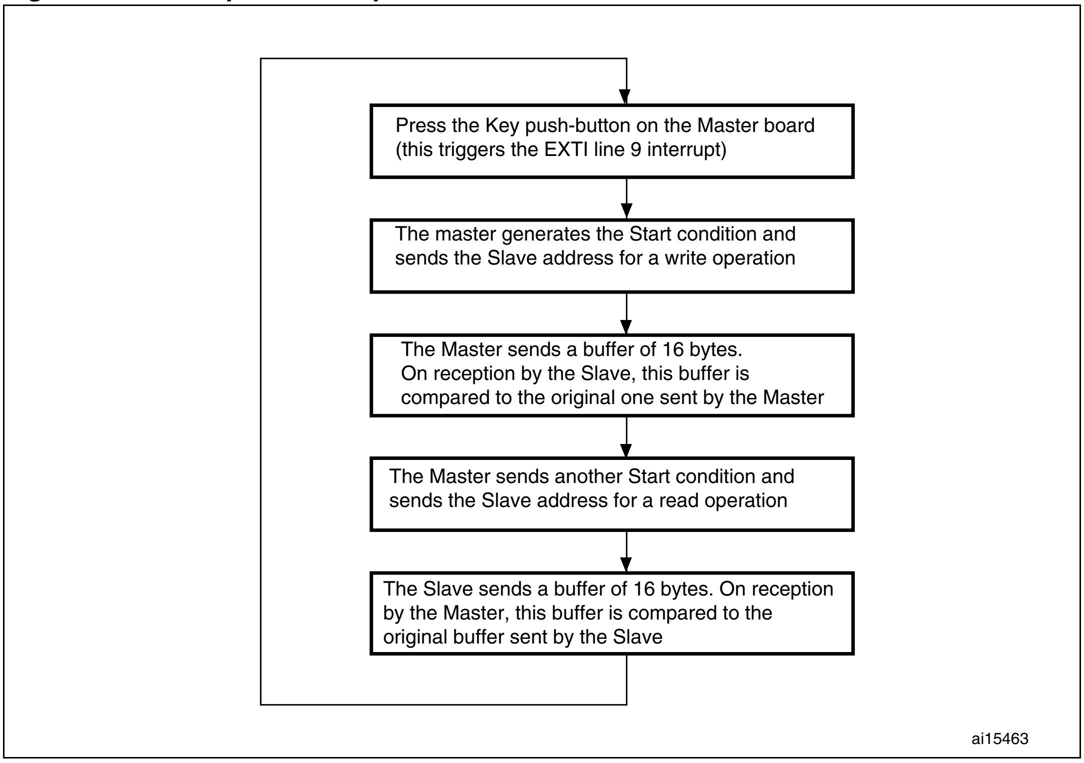
The Master transmits a buffer of bytes to the Slave. The buffer size is predetermined by the user (in the provided firmware, the buffer size is 16 bytes).

The buffer received by the Slave is compared with the originally sent buffer and then their elements are incremented by 2 and re-sent to the Master.

The buffer received by the Master is compared with the buffer sent by the Slave.

This sequence can be relaunched by pressing the Key push-button connected to EXTI Line9 on the STM3210B-EVAL board operating as Master.

Figure 2. Example 1 description

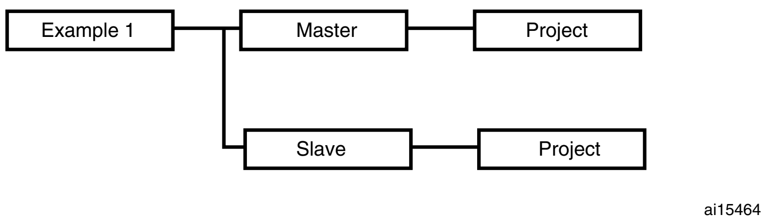


1.4 Firmware details

The example 1 firmware is structured as follows:

- **Slave** folder: contains the Slave project and the corresponding source and header files.
  - *main.c*: file in which the system clocks, I2C2 and interrupts are configured
  - *stm32f10x_it.c*: file in which the I2C2 event (slave transmitter/receiver events) and error (acknowledge failure and bus error) interrupts are handled
  - *stm32f10x_it.h*: headers of the interrupt handlers
  - *stm32f10x_conf.h*: configuration file
- **Master** folder: contains the Master project and the corresponding source and header files.
  - *main.c*: file in which the system clocks, I2C1, EXTI line 9 and standard interrupts are configured
  - *stm32f10x_it.c*: file in which the I2C1 event (master transmitter/receiver events) interrupts are handled
  - *stm32f10x_it.h*: Headers of the interrupt handlers
  - *stm32f10x_conf.h*: configuration file

Figure 3. Example1 directory structure



1.5 How to run the example

In order to run the example successfully, go through the following steps:

- Load the Slave project into board 1
- Load the Master project into board 2
- Run the code in board 1
- Run the code in board 2
- Press the PB9 push-button on board 2 in order to launch the communication.

You can either use an oscilloscope to visualize the data frames or simply read the buffers received by the Slave and the Master using the debugger.

2 Example 2

2.1 Overview

In this example, I^2C communication is performed between three STM32F10xxx microcontrollers. One device is operating as the Master receiver and the other two are operating as Slave transmitters.

As mentioned in *Section 1.1: Overview*, “STM32F10xxx I2C1” or “I2C1” is used to refer to the I2C1 interface of an STM32F10xxx microcontroller with two I^2C controllers and “STM32F10xxx I2C2” or “I2C2” refers to the I2C2 interface of another STM32F10xxx microcontroller also featuring two I^2C controllers.

2.2 Hardware environment

Figure 4: Hardware connection shows the connection between an STM32F10xxx I2C1 and two STM32F10xxx I2C2s.

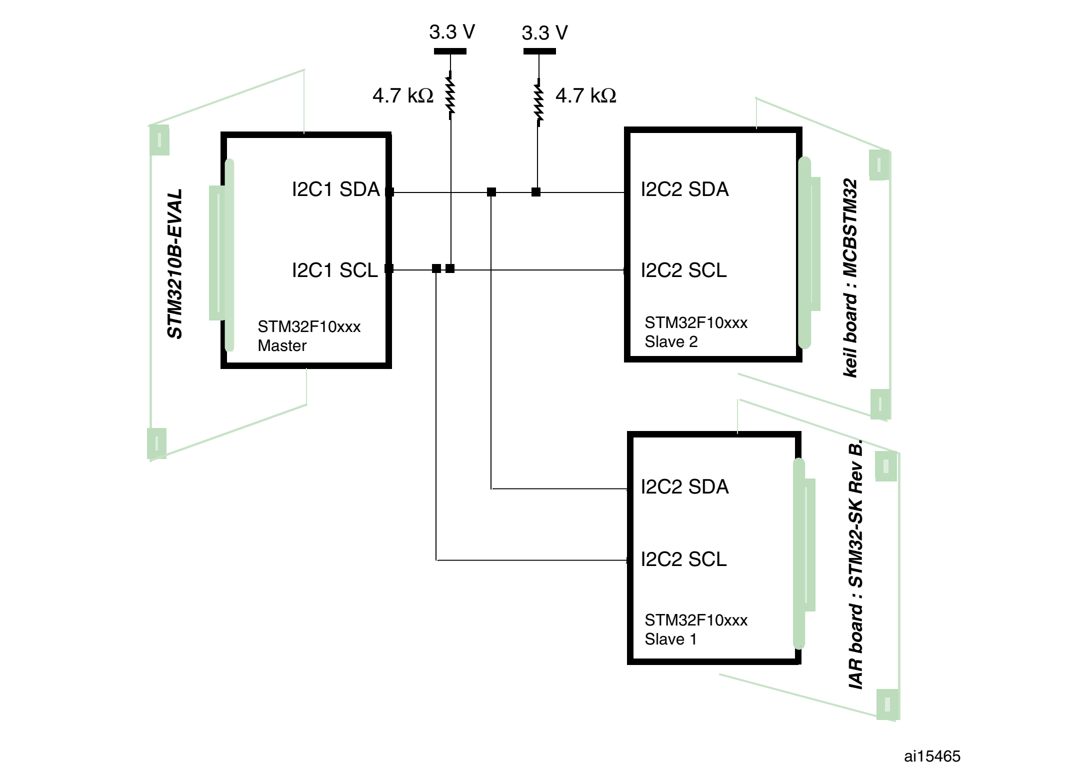
The I2C1 and I2C2 data (SDA) lines are connected together. The I2C1 and I2C2 clock (SCL) lines are also connected together. A pull-up resistor is connected to each line (SDA and SCL).

Like in the case of example 1, there is not a specific board (single PCB) fitted with three STM32F10xxx devices, so we chose to connect an STM3210B-EVAL board (Master) to an IAR board (STM32-SK Rev B) (Slave 1) and a Keil board (MCBSTM32 v1.1) (Slave 2).

For more details about the boards used in these examples, please refer to the following links:

- *http://www.keil.com* for the MCBSTM32 board
- *http://www.iar.com* for the STM32-SK board
- *http://www.st.com/stm32* for the STM3210B-EVAL board

Figure 4. Hardware connection



2.3 Example 2 description

Slave 1 (IAR board) sends the converted values of the ADC1 channel 15 (ADC1_CH15) to the Master. ADC1_CH15 is connected to the potentiometer on the board.

Slave 2 sends the temperature values to the Master, using the on-chip temperature sensor that is internally connected to ADC1_CH16.

The Start condition is generated within a timer interrupt triggered once every second. Consequently, both Slaves send data alternately every second, approximately. Once the Master has generated the Start condition and sent the Slave address:

- if Slave1 is addressed, ADC1_CH15 conversions are launched. The number of conversions is to be fixed by the user. In the example, 16 conversions are carried out. DMA1 channel 1 is used to transfer the data to the I2C2_Buffer_Tx buffer, which is then sent to the Master.
- if Slave2 is addressed, ADC1_CH16 conversions are launched. The number of conversions is to be fixed by the user. In the example, 16 conversions are carried out. DMA1 channel 1 is used to transfer the data to a table named Buffer. Then the I2C2_Buffer_Tx buffer is filled with the temperature values using elements in the Buffer table and the formula provided in the device reference manual. I2C2_Buffer_Tx buffer is then sent to the Master.

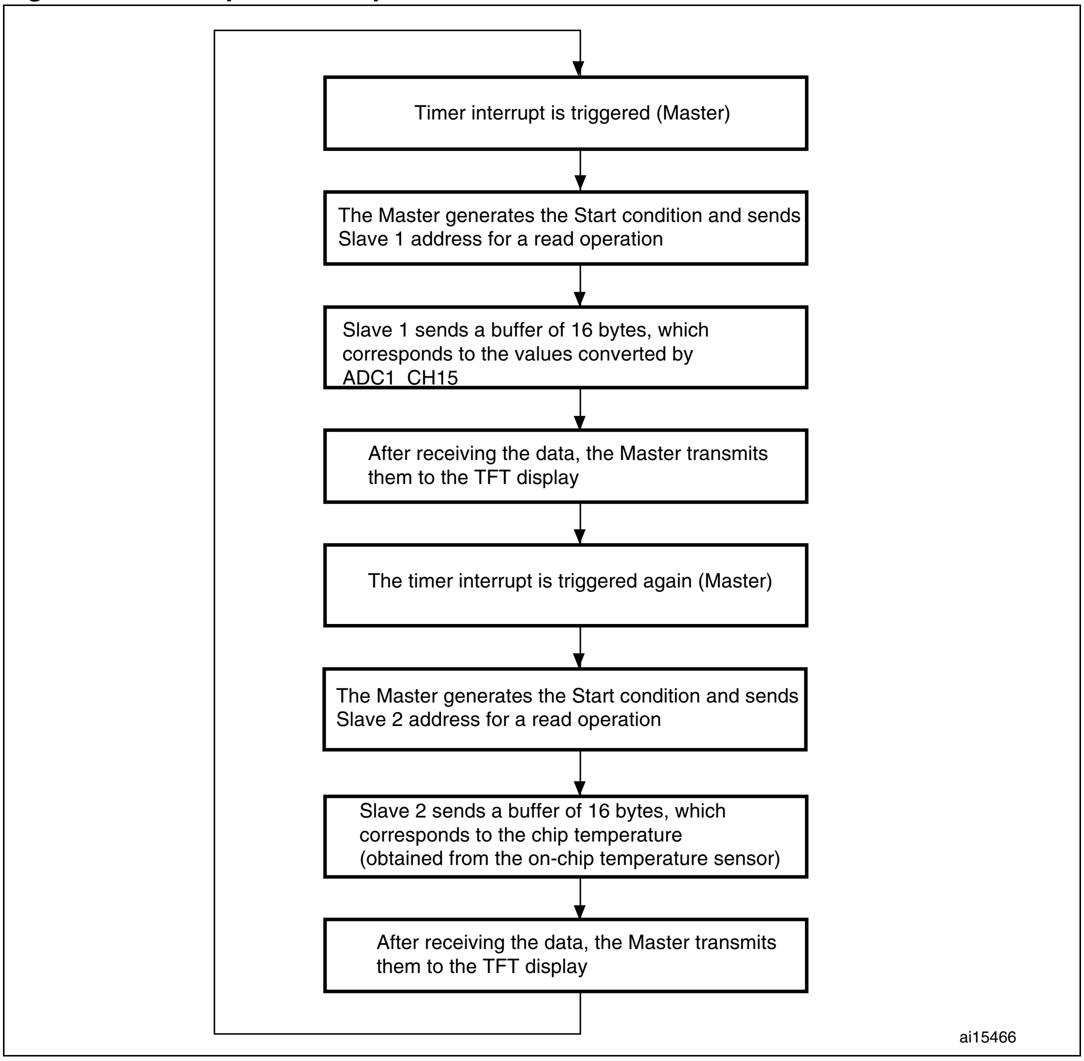
When the transmission is complete, the Slaves enter the Sleep mode using WFI.

When a Start condition is generated by the Master and the Slave address is sent, the Slave address match will trigger an I^2C interrupt that will wake up the corresponding Slave.

After receiving the data, the Master transmits them to the display interface.

Note: The display on the STM3210B-EVAL is connected to the SPI whereas on the STM3210E-EVAL (high-density evaluation board) it is connected to the FSMC. So to be able to use the provided firmware with the STM3210E-EVAL board, it is first necessary to update it.

Figure 5. Example 2 description



2.4 Firmware details

The example 2 firmware is structured as follows:

- **Slave1** folder: it contains the Slave 1 project and the corresponding source and header files:
  - *main.c*: file in which the system clocks, I2C2, DMA1 channel 1, ADC1_CH15 and interrupts are configured
  - *stm32f10x_it.c*: file where the I2C2 event (slave receiver events) and error (acknowledge failure and bus error) interrupts are handled
  - *stm32f10x_it.h*: headers of the interrupt handlers
  - *stm32f10x_conf.h*: configuration file
- **Slave2** folder: it contains the Slave 2 project and the corresponding source and header files:
  - *main.c*: file in which the system clocks, I2C2, DMA1 Channel1, ADC1 channel 16 and interrupts are configured.
  - *stm32f10x_it.c*: file where the I2C2 event (slave receiver events) and error (acknowledge failure and bus error) interrupts are handled
  - *stm32f10x_it.h*: headers of the interrupt handlers
  - *stm32f10x_conf.h*: configuration file
- **Master** folder: it contains the Master project and the corresponding source and header files. It is subdivided into three subfolders, the:
  - **Source** subfolder contains:

    main.c: file where the system clocks, I2C1, TIM2, LCD and interrupts are configured.

    stm32f10x_it.c: file where I2C1 events (master receiver events) and TIM2 interrupts are programmed.

    lcd.c: it contains some LCD routines for the AM-240320LTNQW00H liquid crystal display module of the STM3210B-EVAL.

    lcd_config.c:
  - **Include** subfolder contains:

    main.h: header file for *main.c*

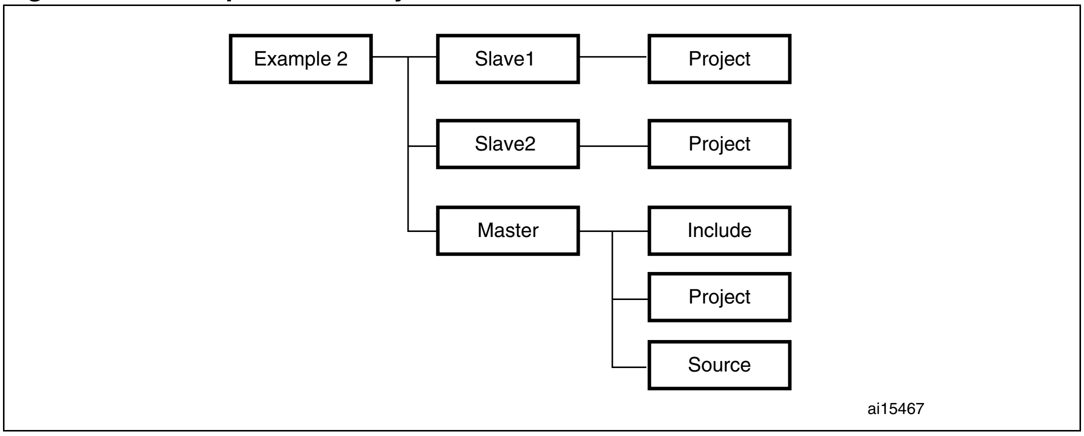
    stm32f10x_it.h: header file for *stm32f10x_it.c*

    lcd.h: header file for *lcd.c*

    lcd_config.h: header file for *lcd_config.c*

    fonts.h: it contains the lcd fonts size definition
  - **Project** subfolder contains preconfigured projects.

Figure 6. Example 2 directory structure



2.5 How to run the example

To run the example successfully, go through the following steps:

- Load the Slave 1 driver into board 1
- Load the Slave 2 driver into board 2
- Load the Master driver into board 3
- Run the code in board 1
- Run the code in board 2
- Run the code in board 3

Now, the values converted by Slave1 ADC1_CH15 and the temperature values provided by Slave 2 are displayed alternately and continuously, every second approximately, on the master board TFT display.

Note: The user can change the firmware so as to launch the I^2C communication with a different periodicity (in the example, periodicity = T = 1 s). Be careful when selecting T because it takes time to display the received bytes on the TFT display. In fact, the selected periodicity should be less than or equal to T otherwise a mixture of ADC1_CH15 (Slave 1) and temperature values (Slave 2) will be displayed.

3 Revision history

Table 2. Document revision history

Date	Revision	Changes
18-Sep-2008	1	Initial release.