AN2582
Application note

STM32F10xxx USART application examples

Introduction

This application note is intended to provide practical application examples of the STM32F10xxx USART peripheral use.

This document, its associated firmware, and other such application notes are written to accompany the STM32F10xxx firmware library. These are available for download from the STMicroelectronics website: *www.st.com.*

Contents

1 STM32F10xxx USART hardware flow control communication

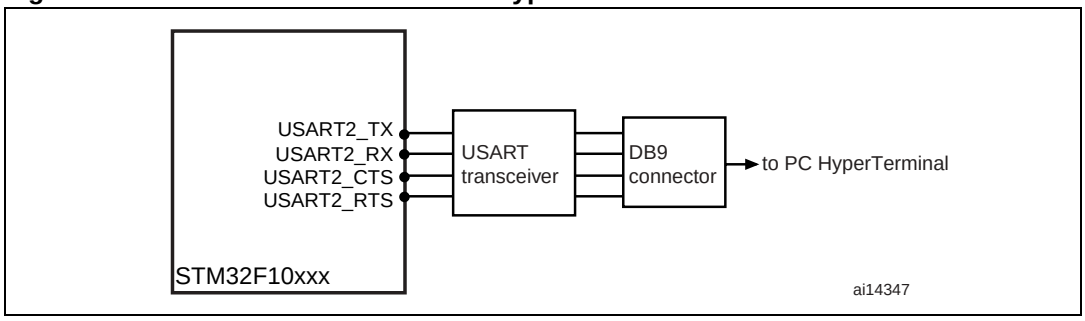
1.1 Overview

This section provides a description of how to use the USART with hardware flow control (RTS, CTS), and communicate with the HyperTerminal.

1.2 Hardware description

Figure 1 shows a typical interface between the STM32F10xxx USART and the PC HyperTerminal. All USART2 signals (Rx, Tx, RTS and CTS) must be connected to a DB9 connector using an RS232 transceiver. Moreover, a null-modem female/female RS232 cable is connected between the DB9 connector (CN5 on STM3210B-EVAL board) and the PC serial port.

Figure 1. STM32F10xxx USART and HyperTerminal interface



1.3 Firmware description

The provided firmware includes the USART driver that supports all USART communications through a set of functions. An example of use for most of these functions is provided.

First, USART2 sends a predefined buffer to the HyperTerminal and waits for a string from the HyperTerminal. The string is entered by the user and must end with the \r character (keypad ENTER button). It is stored into the receive buffer array. The maximum receive buffer size is defined in bytes in the *RxBufferSize* variable and can be configured by the user.
Each received byte is retransmitted to the HyperTerminal.

This firmware is provided as *USART example 1* in the STM32F10xxx firmware library, available from the STMicroelectronics microcontrollers website.

1.4 Conclusion

The STM32F10xxx USART has a modem function (CTS and RTS) that improves the application security regarding data transfers, and requires less software to control the data flow.

2 STM32F10xxx USART interrupt communication with HyperTerminal

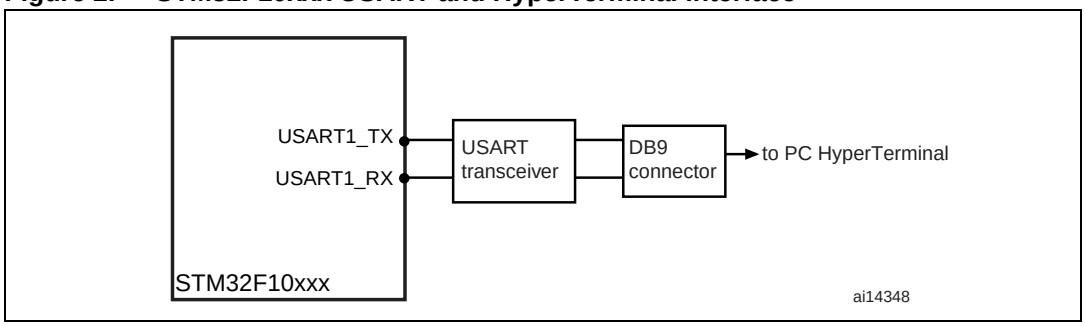
2.1 Overview

This section provides a description of how to use the USART interrupts to communicate with the HyperTerminal.

2.2 Hardware description

Figure 2 shows a typical interface between the STM32F10xxx USART and the PC HyperTerminal. The USART1 signals (Rx, Tx) must be connected to a DB9 connector using an RS232 transceiver. Moreover, a null-modem female/female RS232 cable is connected between the DB9 connector (CN6 on STM3210B-EVAL board) and the PC serial port.

Figure 2. STM32F10xxx USART and HyperTerminal interface



2.3 Firmware description

The provided firmware includes the USART driver that supports all USART communications through a set of functions. An example of use for most of these functions is provided.

First, USART1 sends a predefined buffer to the HyperTerminal and waits for a string from the HyperTerminal. The string length is defined by the user-configurable *RxBufferSize* variable. The communication is managed by Transmit and Receive interrupts. The user-entered string is stored into the receive buffer array. The receive buffer has a maximum size of *RxBufferSize* bytes.

This firmware is provided as *USART example 2* in the STM32F10xxx firmware library, available from the STMicroelectronics microcontrollers website.

2.4 Conclusion

The STM32F10xxx USART interrupts provide a greater flexibility when sending and receiving data. Communication with the HyperTerminal was used for the demonstration purpose.

3 STM32F10xxx USART-USART communication using flags

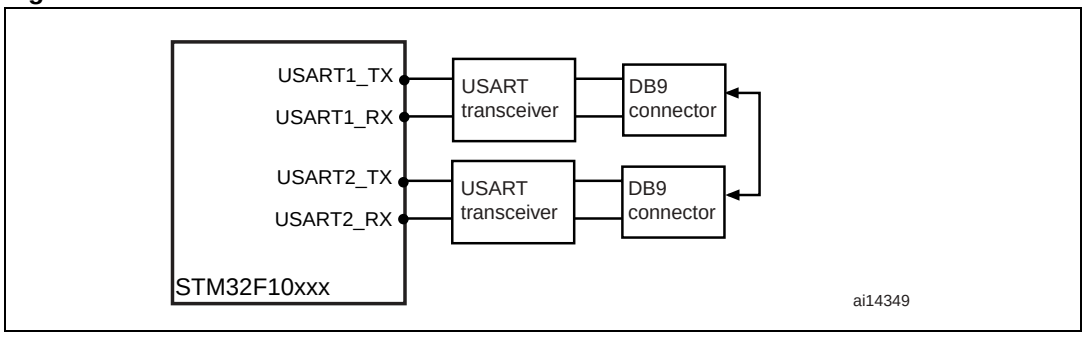
3.1 Overview

This section describes how to build a basic communication between USART1 and USART2 using flags.

3.2 Hardware description

Figure 3 shows a typical interface between the STM32F10xxx USART1 and USART2. The USART1 signals (Rx, Tx) must be connected to a DB9 connector using an RS232 transceiver and the same connection must be done for the USART2 signals. Moreover, a null-modem female/female RS232 cable is connected between the two DB9 connectors (CN6 and CN5 on STM3210B-EVAL board).

Figure 3. STM32F10xxx USART-USART interface



3.3 Firmware description

The provided firmware includes the USART driver that supports all USART communications through a set of functions. An example of use for most of these functions is provided.

First, USART1 sends a predefined buffer to USART2. USART2 reads the received data and stores it into its receive buffer. The received data are then compared with the sent data and the result of this comparison is stored into the *TransferStatus* variable.

This firmware is provided as *USART example 3* in the STM32F10xxx firmware library, available from the STMicroelectronics microcontrollers website.

3.4 Conclusion

The STM32F10xxx USART flags can easily be used to control a communication between two USARTs.

4 STM32F10xxx USART-USART communication using interrupts

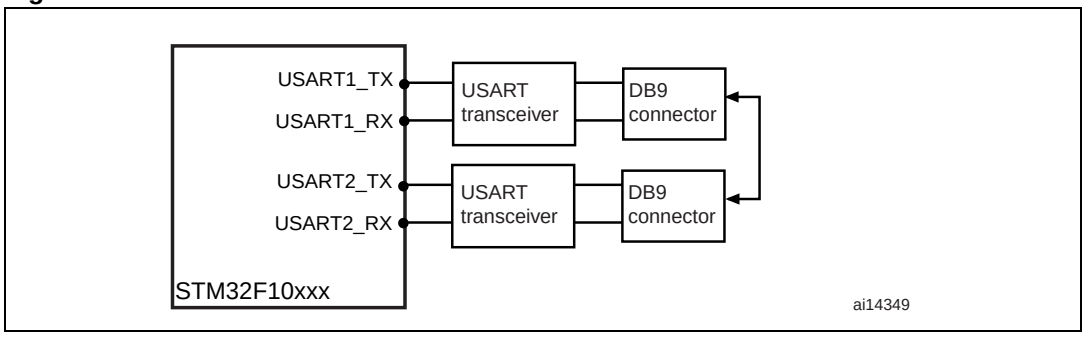
4.1 Overview

This section describes how to build a basic communication between USART1 and USART2 using interrupts.

4.2 Hardware description

Figure 4 shows a typical interface between the STM32F10xxx USART1 and USART2. The USART1 signals (Rx, Tx) must be connected to a DB9 connector using an RS232 transceiver and the same connection must be done for the USART2 signals. Moreover, a null-modem female/female RS232 cable is connected between the two DB9 connectors (CN6 and CN5 on STM3210B-EVAL board).

Figure 4. STM32F10xxx USART-USART interface



4.3 Firmware description

The provided firmware includes the USART driver that supports all USART communications through a set of functions. An example of use for most of these functions is provided.

First, two predefined buffers are sent, one from USART1 to USART2 and the other, from USART2 to USART1. The received data are stored into the USART2 and USART1 receive buffers, respectively. The data transfers are managed in the *stm32f10x.c* files in the USART1 and USART2 interrupt service routines. The sent and the received data buffers are then compared.

This firmware is provided as *USART example 4* in the STM32F10xxx firmware library, available from the STMicroelectronics microcontrollers website.

4.4 Conclusion

The use of USART interrupts within a communication further reduces the code density and makes it easy to transmit and receive data.

5 STM32F10xxx USART-USART communication using DMA

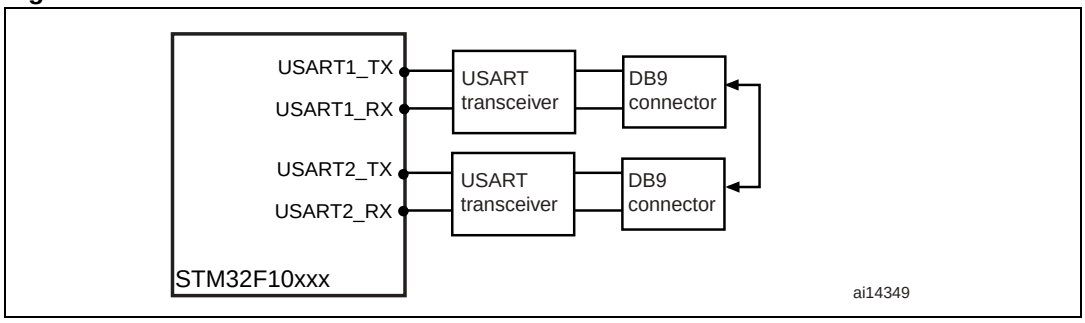
5.1 Overview

This section describes how to build a basic communication between USART1 and USART2 using the DMA capability.

5.2 Hardware description

Figure 5 shows a typical interface between the STM32F10xxx USART1 and USART2. The USART1 signals (Rx, Tx) must be connected to a DB9 connector using an RS232 transceiver and the same connection must be done for the USART2 signals. Moreover, a null-modem female/female RS232 cable is connected between the two DB9 connectors (CN6 and CN5 on STM3210B-EVAL board).

Figure 5. STM32F10xxx USART-USART interface



5.3 Firmware description

The provided firmware includes the USART driver that supports all USART communications through a set of functions. An example of use for most of these functions is provided.

First, DMA is used to transfer data from a predefined transmit buffer to the USART2 Transmit data register, then the data are sent to USART1. The data received by USART1 are transferred by DMA and stored into a predefined receive buffer, then compared with the sent data and the result of the comparison is stored into the *TransferStatus1* variable.

Then, DMA is used to transfer data from a predefined transmit buffer to the USART1 Transmit data register, then the data are sent to USART2. The data received by USART2 are transferred by DMA and stored into a predefined receive buffer, then compared with the sent data and the result of the comparison is stored into the *TransferStatus2* variable.

This firmware is provided as *USART example 5* in the STM32F10xxx firmware library, available from the STMicroelectronics microcontrollers website.

5.4 Conclusion

The use of DMA within a communication further reduces the code density and execution time. It is also easier to transmit and receive data.

6 USART-USART communication using DMA, flags and interrupts

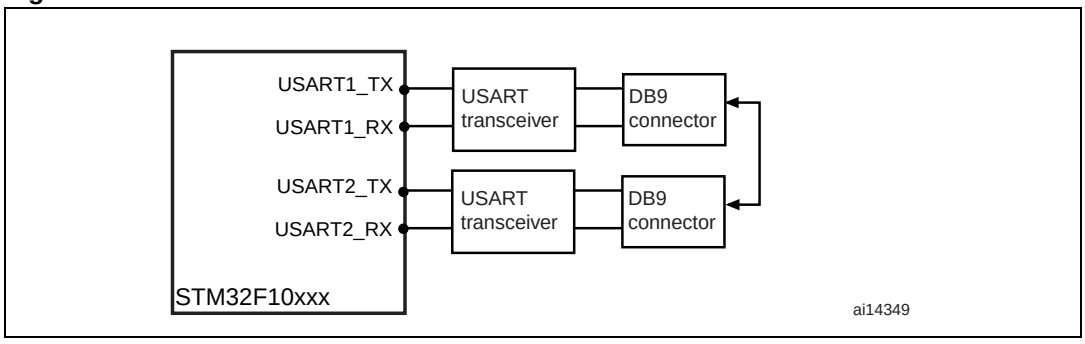
6.1 Overview

This section describes how to build a basic communication between USART1 and USART2 using DMA, flags and interrupts.

6.2 Hardware description

Figure 6 shows a typical interface between the STM32F10xxx USART1 and USART2. The USART1 signals (Rx, Tx) must be connected to a DB9 connector using an RS232 transceiver and the same connection must be done for the USART2 signals. Moreover, a null-modem female/female RS232 cable is connected between the two DB9 connectors (CN6 and CN5 on STM3210B-EVAL board).

Figure 6. STM32F10xxx USART-USART interface



6.3 Firmware description

The provided firmware includes the USART driver that supports all USART communications through a set of functions. An example of use for most of these functions is provided.

First, DMA is used to transfer data from a predefined transmit buffer to the USART2 Transmit data register, then the data are sent to USART1. The data received by USART1 are transferred using the RxNE flag, and stored into a predefined receive buffer. They are then compared with the sent data and the result of this comparison is stored into the *TransferStatus1* variable.

Then, DMA is used to transfer data from a predefined transmit buffer to the USART1 Transmit data register, then the data are sent to USART2. The data received by USART2 are transferred using the Receive interrupt, and stored into a predefined receive buffer. They are then compared with the sent data and the result of this comparison is stored into the *TransferStatus2* variable.

This firmware is provided as *USART example 6* in the STM32F10xxx firmware library, available from the STMicroelectronics microcontrollers website.

6.4 Conclusion

In multibuffer communication, the STM32F10xxx USART triggers DMA on send/receive requests that can be used for data transfer. This leaves the CPU free to perform other tasks.

7 STM32F10xxx USART retargeting of C *printf* function

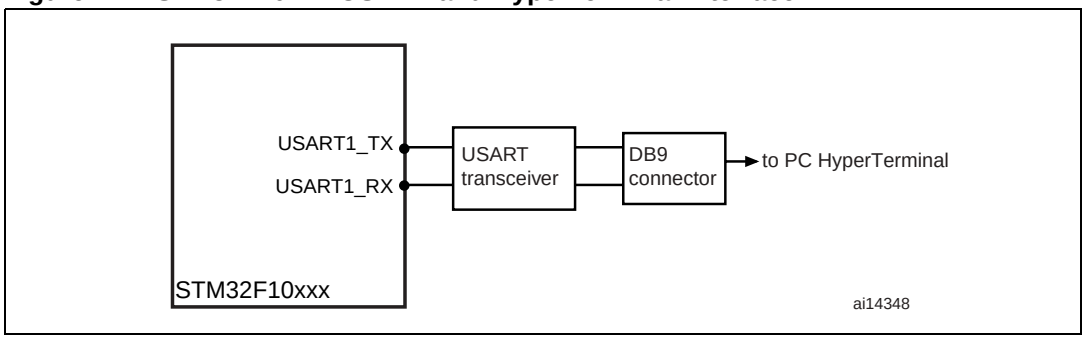
7.1 Overview

This section shows how use the USART to retarget the C library *printf* function.

7.2 Hardware description

Figure 7 shows a typical interface between the STM32F10xxx USART and the PC HyperTerminal. The USART1 signals (Rx, Tx) must be connected to a DB9 connector using an RS232 transceiver. Moreover, a null-modem female/female RS232 cable is connected between the DB9 connector (CN6 on STM3210B-EVAL board) and the PC serial port.

Figure 7. STM32F10xxx USART and HyperTerminal interface



7.3 Firmware description

The provided firmware includes the USART driver that supports all USART communications through a set of functions. An example of use for most of these functions is provided.

This implementation outputs the *printf* message on the HyperTerminal using USARTx. USARTx can be USART1, USART2 or USART3; to select the USART interface to be used uncomment the `#define USE_USARTx` line in the *main.h* file.

This firmware is provided as *USART example 7* in the STM32F10xxx firmware library, available from the STMicroelectronics microcontrollers website.

7.4 Conclusion

The STM32F10xxx USART can be used to retarget the C *printf* function. The user can display the message on the HyperTerminal.

8 STM32F10xxx USART synchronous mode (SPI mode)

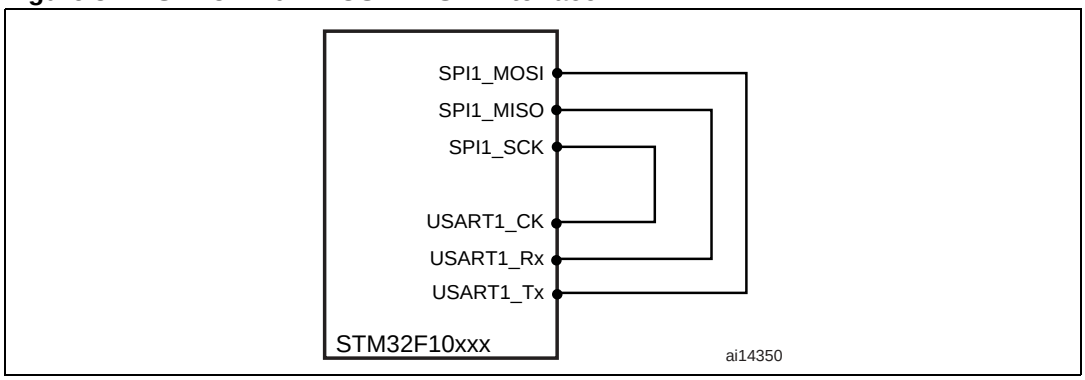
8.1 Overview

This section describes how to build a basic communication between USART1 (Synchronous mode) and SPI1 using flags.

8.2 Hardware description

Figure 8 shows a typical interface between the STM32F10xxx USART1 and SPI1. USART1_Tx (PA9) is connected to SPI1_MOSI (PA7), USART1_Rx (PA10) is connected to SPI1_MISO (PA6) and USART1_CK (PA8) is connected to SPI1_SCK(PA5).

Figure 8. STM32F10xxx USART-SPI interface



8.3 Firmware description

The provided firmware includes the USART driver that supports all USART communications through a set of functions. An example of use for most of these functions is provided.

First, USART1 sends data from a predefined transmit buffer to SPI1 using the TxE flag. The data, received by SPI1 using the RxNE flag, are stored into a predefined receive buffer, then compared with the sent data and the result of the comparison is stored into the *TransferStatus1* variable.

Then, SPI1 sends data from a predefined transmit buffer to USART1 using the TxE flag. The data received by USART1 using the RxNE flag, is stored into a predefined receive buffer, then compared with the sent data and the result of the comparison is stored into the *TransferStatus2* variable.

This firmware is provided as *USART example 8* in the STM32F10xxx firmware library, available from the STMicroelectronics microcontrollers website.

8.4 Conclusion

In synchronous mode, with its clock output, the STM32F10xxx USART can communicate with an SPI interface. The USART can only be the master.

9 STM32F10xxx USART half-duplex mode

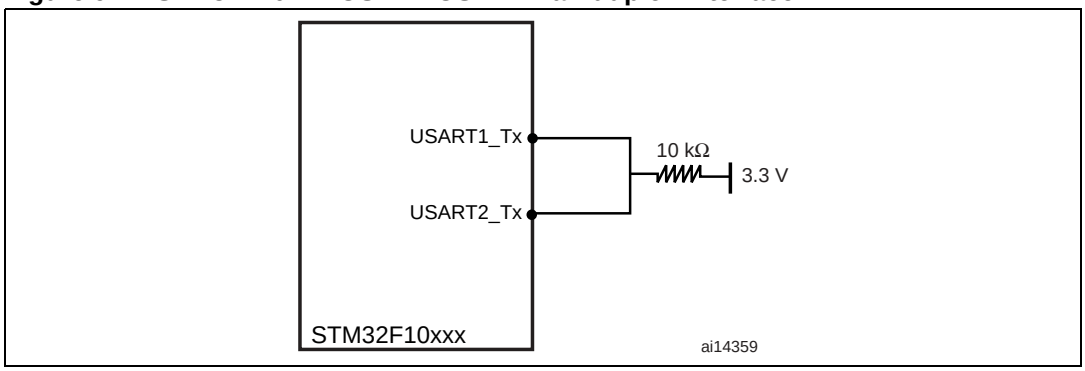
9.1 Overview

This section describes how to build a basic communication between USART1 and USART2 in half-duplex mode using flags.

9.2 Hardware description

Figure 9 shows a typical interface between the STM32F10xxx USART1 and USART2 in half-duplex mode. USART1_Tx (PA9) is connected to USART2_Tx (PD5) and a pull-up resistor is connected to the line (10 kΩ).

Figure 9. STM32F10xxx USART-USART half-duplex interface



9.3 Firmware description

The provided firmware includes the USART driver the supports all USART communications through a set of functions. An example of use for most of these functions is provided.

First, USART1 sends data from a predefined transmit buffer to USART2 using the TxE flag. The data received by USART2 using RxNE flag, is stored into a predefined receive buffer, then compared with the sent data, and the result of the comparison is stored into the *TransferStatus1* variable.

Then, USART2 sends data from a predefined transmit buffer to USART1 using the TxE flag. The data received by USART1 using the RxNE flag is stored into a predefined receive buffer, then compared with the sent data and the result of this comparison is stored into the *TransferStatus2* variable.

This firmware is provided as *USART example 9* in the STM32F10xxx firmware library, available from the STMicroelectronics microcontrollers website.

9.4 Conclusion

The STM32F10xxx USART can communicate in half-duplex mode. In this mode, the USARTx_Rx pin is no longer used.

10 STM32F10xxx USART IrDA mode

The example 10 provides two IrDA programs: transmitter & receiver. To run the full demonstration, two boards are required:

- one board to act as the IrDA transmitter
- one board to act as the IrDA receiver

10.1 STM32F10xxx USART IrDA Transmit mode

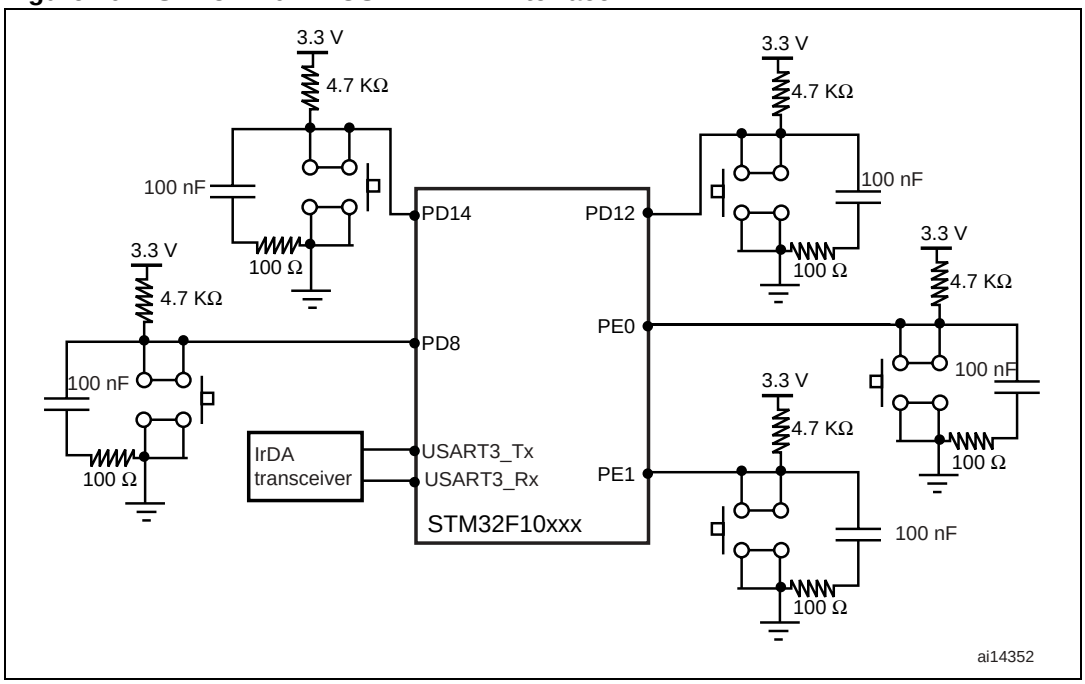
10.1.1 Overview

This section describes how to build a basic communication with USART3 in IrDA transmit mode.

10.1.2 Hardware description

Figure 10 shows the STM32F10xxx IrDA interface.

Figure 10. STM32F10xxx USART IrDA interface



10.1.3 Firmware description

The provided firmware includes the USART driver that supports all USART communications through a set of functions. An example of use for most of these functions is provided.

Five pins are used to select the byte to be sent. Each pressed key transmits an associated byte on the USART3_Tx pin

These bytes are:

- 0x00 if no key pressed
- 0x01 if PD12 pin state changes
- 0x02 if PE0 pin state changes
- 0x03 if PE1 pin state changes
- 0x04 if PD8 pin state changes
- 0x05 if PD14 pin state changes

This firmware is provided as *USART example 10 Transmit* in the STM32F10xxx firmware library, available from the STMicroelectronics microcontrollers website.

10.1.4 Conclusion

The STM32F10xxx USART IrDA mode can be used to transmit data at the maximum baudrate of 115200 bps in accordance with the IrDA standard. The USART also supports the IrDA low-power mode.

10.2 STM32F10xxx USART IrDA Receive mode

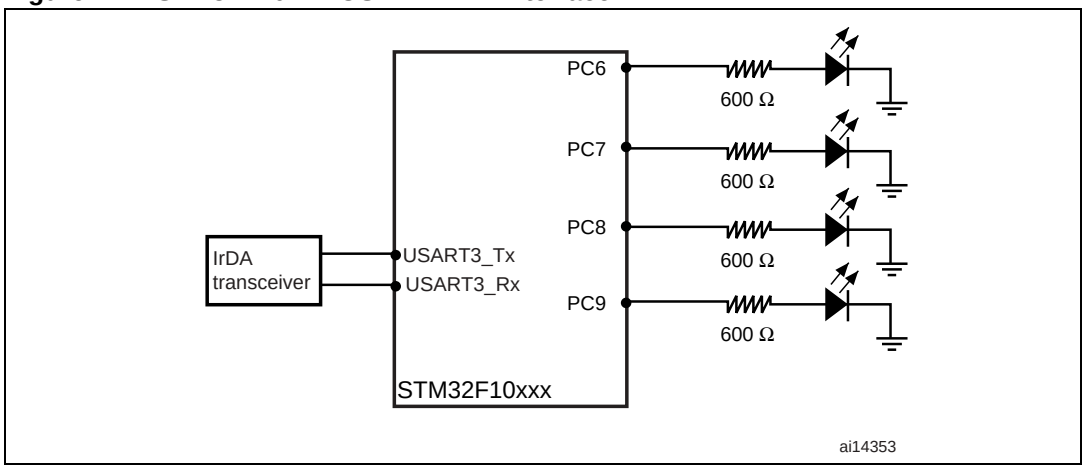
10.2.1 Overview

This section describes how to build a basic communication with USART3 in IrDA Receive mode.

10.2.2 Hardware description

Figure 11 shows the STM32F10xxx IrDA interface.

Figure 11. STM32F10xxx USART IrDA interface



10.2.3 Firmware description

The provided firmware includes the USART driver that supports all USART communications through a set of functions. An example of use for most of these functions is provided.

The USART is waiting for a byte and depending on which byte is received, an LED will toggle. Four LEDs are used to indicate which byte is received.

- when 0x04 is received the LED connected to PC6 toggles
- when 0x05 is received the LED connected to PC7 toggles
- when 0x03 is received the LED connected to PC8 toggles
- when 0x02 is received the LED connected to PC9 toggles
- when 0x01 is received the LEDs connected to PC6, PC7, PC8 and PC9 toggle

This firmware is provided as *USART example 10 Receive* in the STM32F10xxx firmware library, available from the STMicroelectronics microcontrollers website.

10.2.4 Conclusion

The STM32F10xxx USART IrDA mode can be used to receive data at the maximum baudrate of 115200 bps in accordance with the IrDA standard. The USART also supports the IrDA low-power mode.

11 STM32F10xxx USART multiprocessor communication

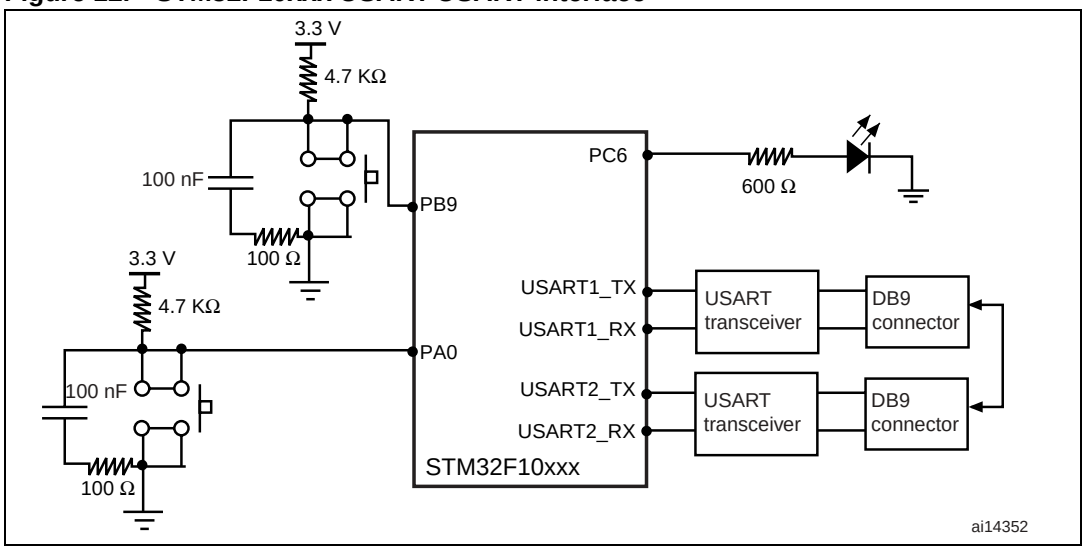
11.1 Overview

This section provides a description of how to use the USART in multiprocessor mode.

11.2 Hardware description

Figure 12 shows a typical interface between the STM32F10xxx USART1 and USART2. The USART1 signals (Rx, Tx) must be connected to a DB9 connector using an RS232 transceiver and the same connection must be done for the USART2 signals. Moreover, a null-modem female/female RS232 cable is connected between the two DB9 connectors (CN6 and CN5 on STM3210B-EVAL board).

Figure 12. STM32F10xxx USART-USART interface



11.3 Firmware description

The provided firmware includes the USART driver that supports all USART communications through a set of functions. An example of use for most of these functions is provided.

First, the USART1 and USART2 addresses are set to 0x1 and 0x2, respectively. USART1 continuously sends the 0x33 character to USART2. USART2 toggles an LED connected to the PC6 while receiving 0x33.

When a falling edge is applied on EXTI Line9 (PB 9), an interrupt is generated and in the EXTI9_5_IRQ Handler routine, USART2 enters the mute mode and remains in this mode (no LED toggling) until a rising edge is applied on EXTI Line0 (PA0). In this interrupt routine, USART1 sends the address mark character (0x102) to wake up USART2. Then the LED restarts toggling.

This firmware is provided as *USART example 11* in the STM32F10xxx firmware library, available from the STMicroelectronics microcontrollers website.

11.4 Conclusion

In multiprocessor configurations, it is often desirable that only the intended message recipient should actively receive the full message contents, thus reducing redundant USART service overhead for all non-addressed receivers.

12 STM32F10x USART Smartcard mode

12.1 Overview

This section provides a description of how to use the USART in Smartcard mode. The example only gives the possibility to read the ATR (answer to reset) and decode it into a predefined buffer.

12.2 Hardware description

Figure 13 shows a typical interface between the STM32F10xxx and a Smartcard that supports the ISO 7816-3 character-oriented (T=0) protocol.

12.3 Firmware description

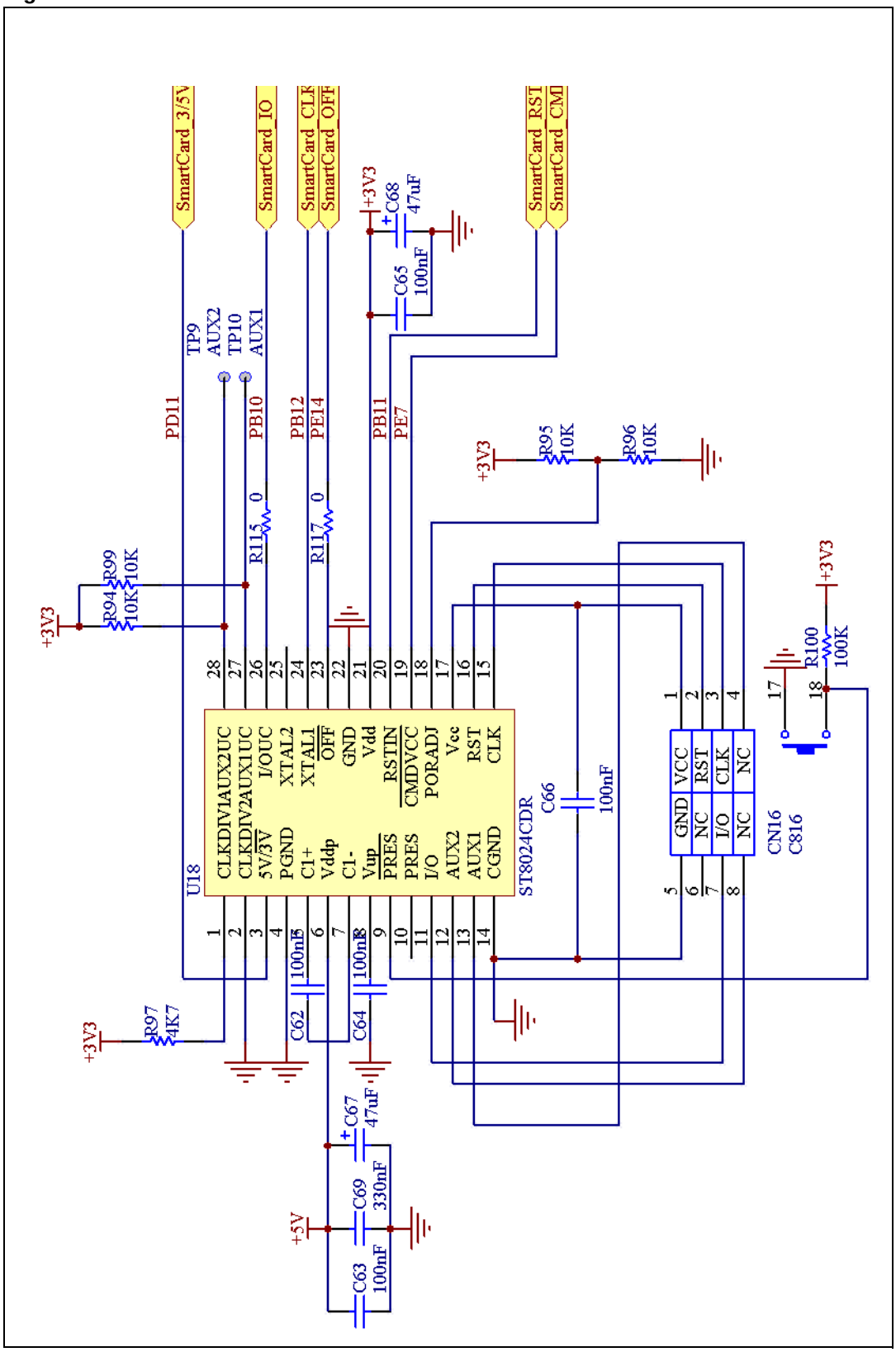
The provided firmware includes the USART driver that supports all USART communications through a set of functions. An example of use for most of these functions is provided.

First, the code is waiting for a card insertion. If a card is detected through an EXTI Line interrupt, a reset signal is applied to the card through its reset pin. In response to this reset, the card transmits the ATR. Once received, the ATR is decoded and stored into a specific structure (SC_A2R). The card protocol type is stored into a variable.

Then a test is performed to check if the inserted card is ISO 7816-3 T=0 compatible.

This firmware is provided as *USART example 12* in the STM32F10xxx firmware library, available from the STMicroelectronics microcontrollers website.

Figure 13. STM32F10x Smartcard interface



13 Revision history

Table 1. Document revision history

Date	Revision	Changes
26-Jun-2007	1	Initial release.

Please Read Carefully:

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST’s terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

UNLESS OTHERWISE SET FORTH IN ST’S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.

UNLESS EXPRESSLY APPROVED IN WRITING BY AN AUTHORIZED ST REPRESENTATIVE, ST PRODUCTS ARE NOT RECOMMENDED, AUTHORIZED OR WARRANTED FOR USE IN MILITARY, AIR CRAFT, SPACE, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS, NOR IN PRODUCTS OR SYSTEMS WHERE FAILURE OR MALFUNCTION MAY RESULT IN PERSONAL INJURY, DEATH, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE. ST PRODUCTS WHICH ARE NOT SPECIFIED AS "AUTOMOTIVE GRADE" MAY ONLY BE USED IN AUTOMOTIVE APPLICATIONS AT USER’S OWN RISK.

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.

Information in this document supersedes and replaces all information previously supplied.

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

www.st.com