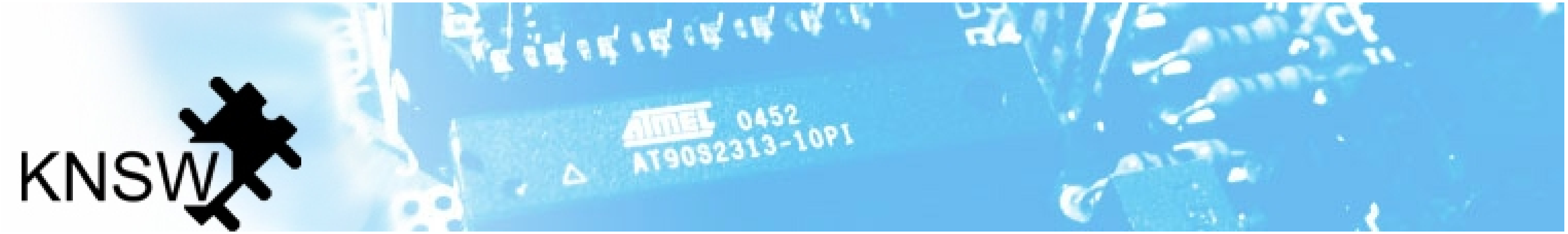


Wydział Elektroniki Politechniki Wrocławskiej

Warsztaty programowania układów mikroprocesorowych STM32

3. USART, RS-232C

Jacek Niepala,
Szymon Panecki

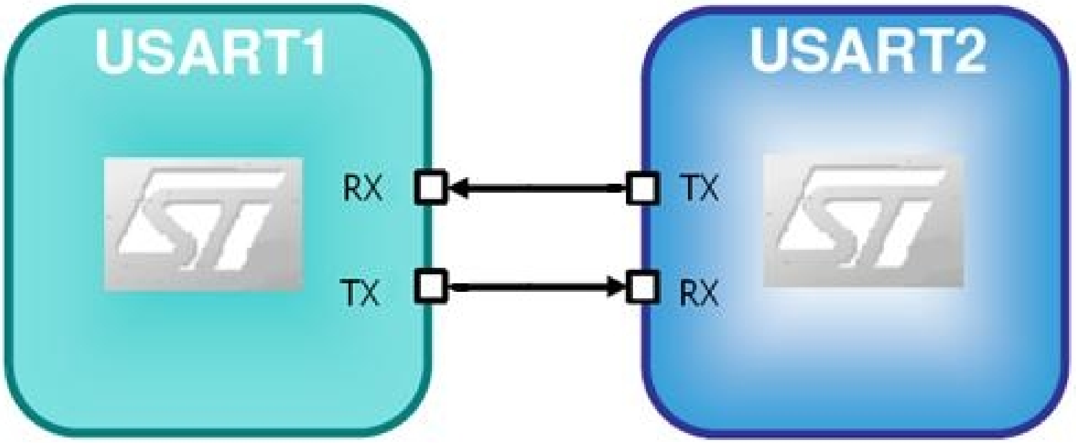
Koło Naukowe Systemów Wbudowanych

USART

USART - ang. Universal synchronous asynchronous receiver transmitter. Układ realizujący dwukierunkową, asynchroniczną lub synchroniczną transmisję szeregową.

Konfiguracja:

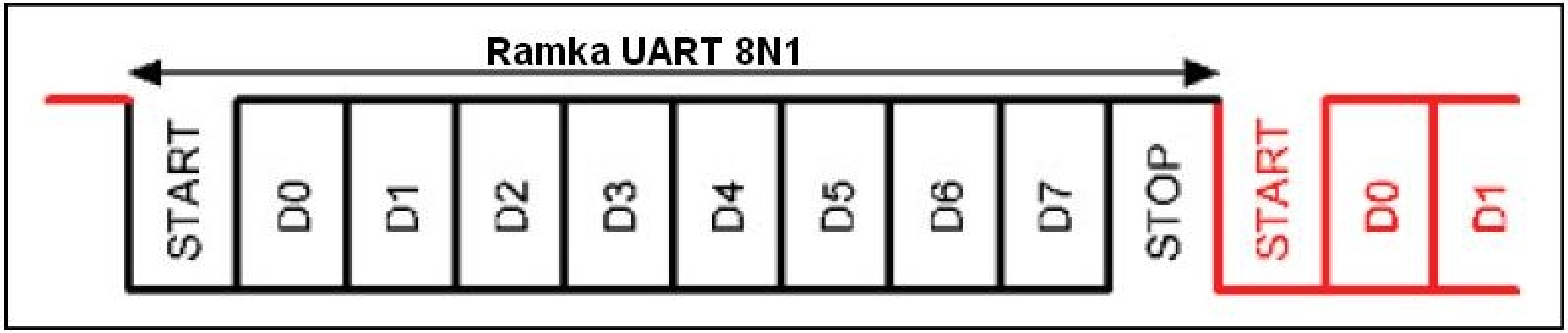
- □ baudrate
- □ bit/bity stopu
- □ parzystość
- □ długość słowa
- □ handshaking



Cechy:

- □ transmisja punkt-punkt
- □ krótki zasięg
- □ mała odporność na zakłócenia

USART - format ramki



- Najpopularniejsze prędkości transmisji[bit/s]: 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, 115200
- Więcej informacji:

www.maxim.com, APPLICATION NOTE 2141

KNSW

USART - wykorzystanie

- □ Komunikacja z układami scalonymi

  (FT232, RFID, RS-232, RS-485, LIN, Power Line, IRDA i inne)

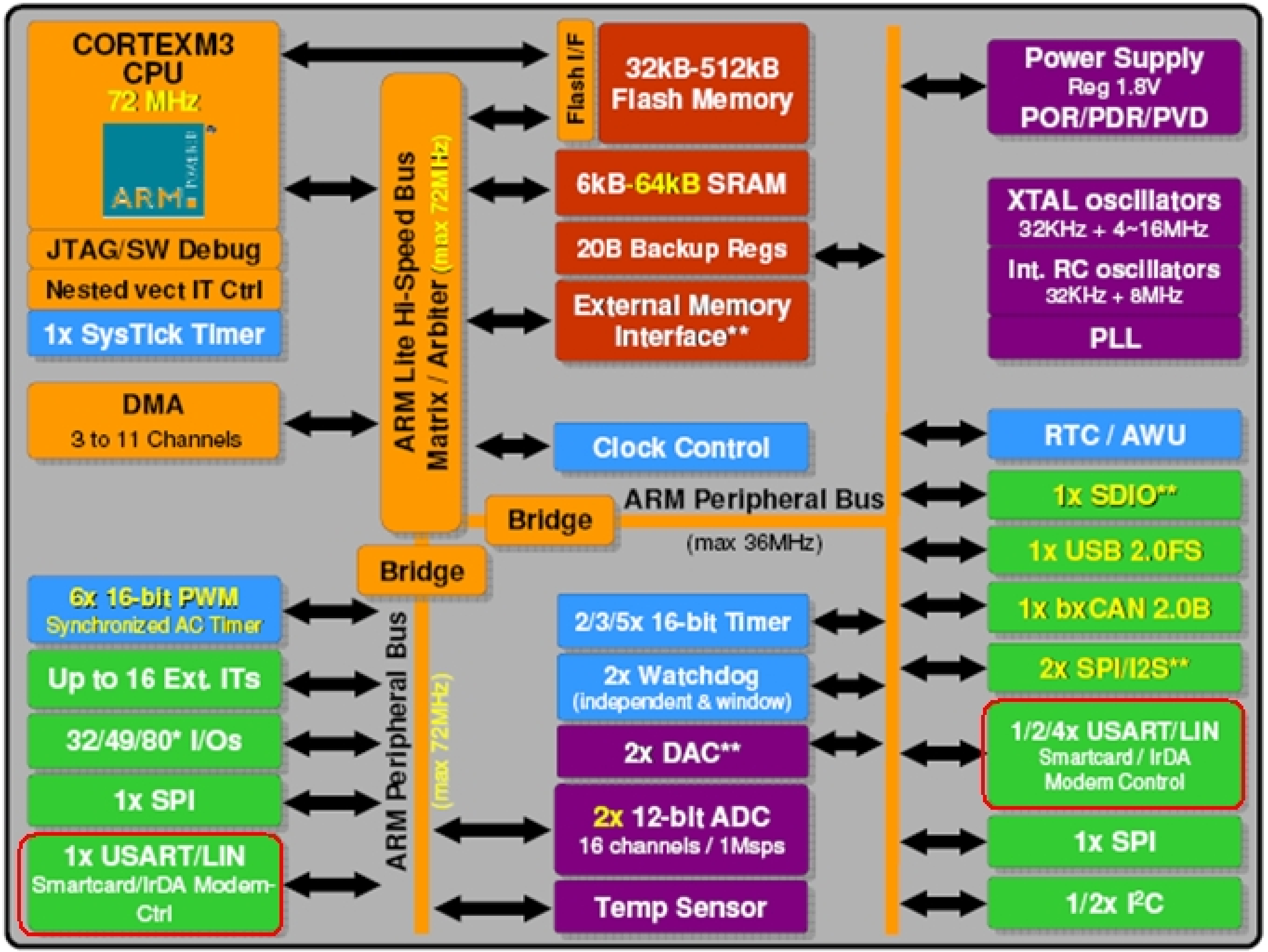
- □ Komunikacja pomiędzy uC np. STM32(Multiprocessor communication) AVR(Multi-processor Communication Mode)

USART w STM32

- Trzy moduły USART: USART1 na High speed APB2 oraz USART2/3 na Low speed APB1
- W pełni programowalna charakterystyka interfejsu:
  - 8 lub 9 bitów danych
  - Odd, Even lub no-parity
  - 12-bitowy generator prędkości transmisji
  - Hardware Flow Control (CTS, RTS)
- Detekcja transmisji poprzez flagi oraz przerwania
- DMA(Direct memory access):
  - Recieve DMA
  - Transmit DMA

KNSW

USART w STM32 c.d.



CORTEXM3 CPU 72 MHz

JTAG/SW Debug

Nested vect IT Ctrl

1x SysTick Timer

DMA 3 to 11 Channels

ARM Lite Hi-Speed Bus Matrix / Arbiter (max 72MHz)

Flash I/F

32kB-512kB Flash Memory

6kB-64kB SRAM

20B Backup Regs

External Memory Interface**

Clock Control

Bridge

ARM Peripheral Bus (max 36MHz)

Bridge

ARM Peripheral Bus (max 72MHz)

6x 16-bit PWM Synchronized AC Timer

Up to 16 Ext. ITs

32/49/80* I/Os

1x SPI

1x USART/LIN Smartcard/IrDA Modem-Ctrl

2/3/5x 16-bit Timer

2x Watchdog (independent & window)

2x DAC**

2x 12-bit ADC 16 channels / 1Msps

Temp Sensor

Power Supply Reg 1.8V POR/PDR/PVD

XTAL oscillators 32KHz + 4~16MHz

Int. RC oscillators 32KHz + 8MHz

PLL

RTC / AWU

1x SDIO**

1x USB 2.0FS

1x bxCAN 2.0B

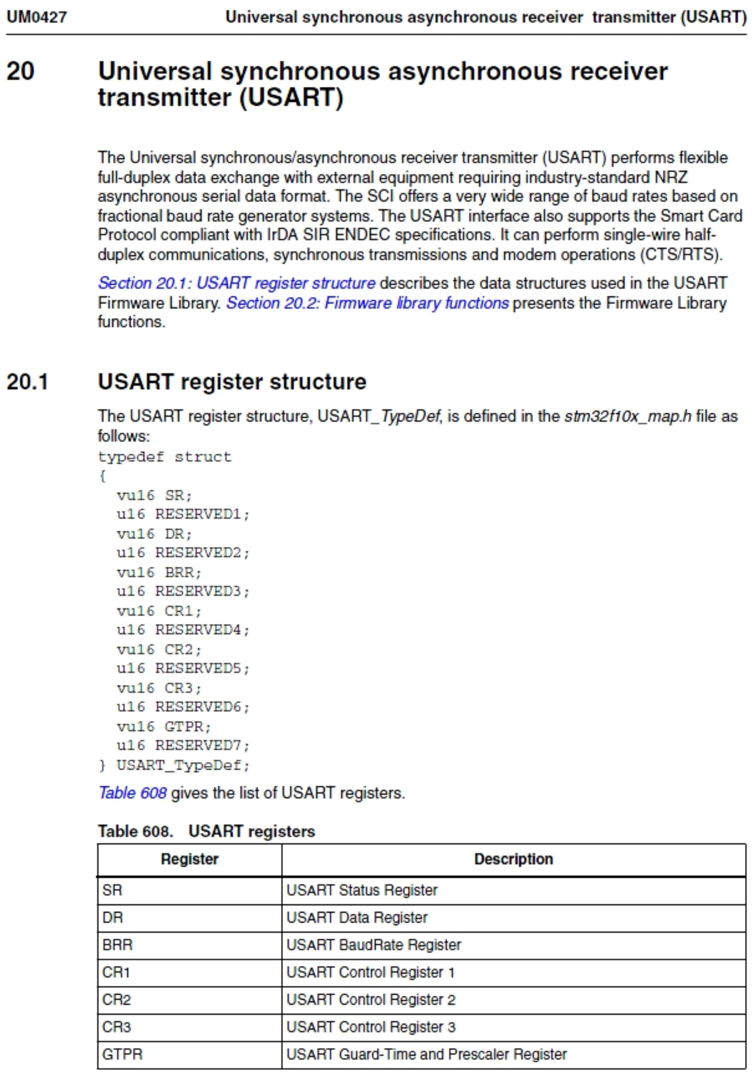
2x SPI/I2S**

1/2/4x USART/LIN Smartcard / IrDA Modem Control

1x SPI

1/2x I^2C

„User manual”



20 Universal synchronous asynchronous receiver transmitter (USART)

The Universal synchronous/asynchronous receiver transmitter (USART) performs flexible full-duplex data exchange with external equipment requiring industry-standard NRZ asynchronous serial data format. The SCI offers a very wide range of baud rates based on fractional baud rate generator systems. The USART interface also supports the Smart Card Protocol compliant with IrDA SIR ENDEC specifications. It can perform single-wire half-duplex communications, synchronous transmissions and modem operations (CTS/RTS).

Section 20.1: USART register structure describes the data structures used in the USART Firmware Library. *Section 20.2: Firmware library functions* presents the Firmware Library functions.

20.1 USART register structure

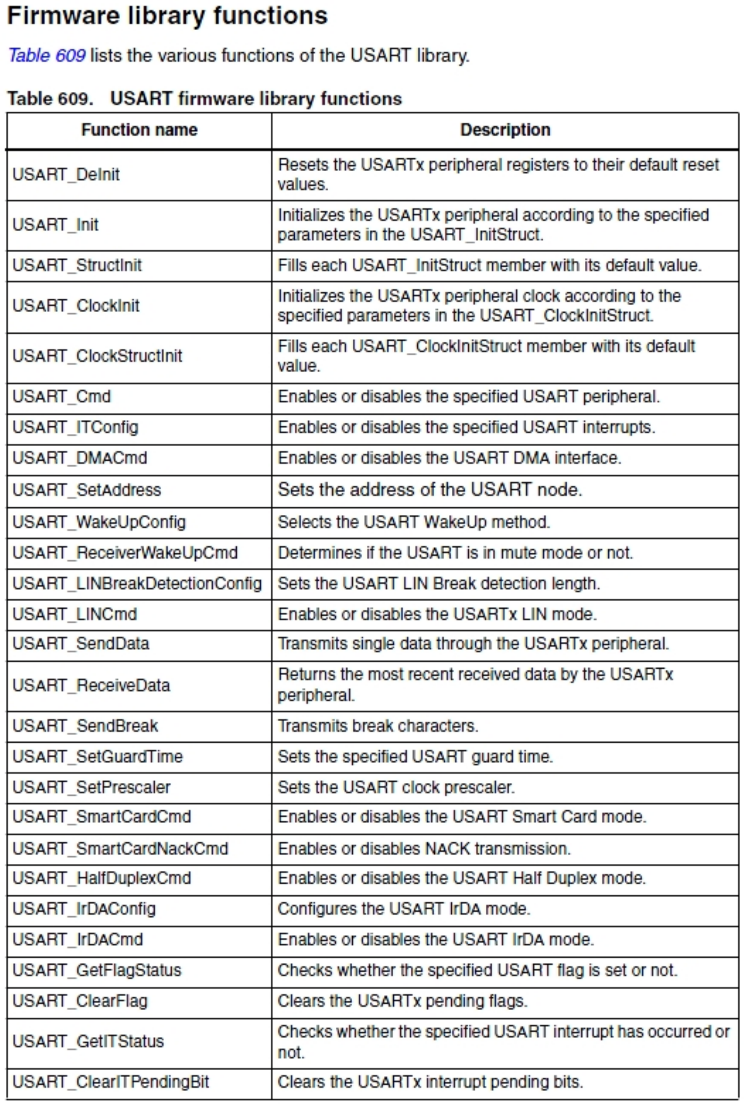
The USART register structure, USART_*TypeDef*, is defined in the *stm32f10x_map.h* file as follows:

```
typedef struct
{
  vu16 SR;
  u16 RESERVED1;
  vu16 DR;
  u16 RESERVED2;
  vu16 BRR;
  u16 RESERVED3;
  vu16 CR1;
  u16 RESERVED4;
  vu16 CR2;
  u16 RESERVED5;
  vu16 CR3;
  u16 RESERVED6;
  vu16 GTPR;
  u16 RESERVED7;
} USART_TypeDef;
```

Table 608 gives the list of USART registers.

Table 608. USART registers

Register	Description
SR	USART Status Register
DR	USART Data Register
BRR	USART BaudRate Register
CR1	USART Control Register 1
CR2	USART Control Register 2
CR3	USART Control Register 3
GTPR	USART Guard-Time and Prescaler Register



Firmware library functions

Table 609 lists the various functions of the USART library.

Table 609. USART firmware library functions

Function name	Description
USART_DeInit	Resets the USARTx peripheral registers to their default reset values.
USART_Init	Initializes the USARTx peripheral according to the specified parameters in the USART_InitStruct.
USART_StructInit	Fills each USART_InitStruct member with its default value.
USART_ClockInit	Initializes the USARTx peripheral clock according to the specified parameters in the USART_ClockInitStruct.
USART_ClockStructInit	Fills each USART_ClockInitStruct member with its default value.
USART_Cmd	Enables or disables the specified USART peripheral.
USART_ITConfig	Enables or disables the specified USART interrupts.
USART_DMACmd	Enables or disables the USART DMA interface.
USART_SetAddress	Sets the address of the USART node.
USART_WakeUpConfig	Selects the USART WakeUp method.
USART_ReceiverWakeUpCmd	Determines if the USART is in mute mode or not.
USART_LINBreakDetectionConfig	Sets the USART LIN Break detection length.
USART_LINCmd	Enables or disables the USARTx LIN mode.
USART_SendData	Transmits single data through the USARTx peripheral.
USART_ReceiveData	Returns the most recent received data by the USARTx peripheral.
USART_SendBreak	Transmits break characters.
USART_SetGuardTime	Sets the specified USART guard time.
USART_SetPrescaler	Sets the USART clock prescaler.
USART_SmartCardCmd	Enables or disables the USART Smart Card mode.
USART_SmartCardNackCmd	Enables or disables NACK transmission.
USART_HalfDuplexCmd	Enables or disables the USART Half Duplex mode.
USART_IrDAConfig	Configures the USART IrDA mode.
USART_IrDACmd	Enables or disables the USART IrDA mode.
USART_GetFlagStatus	Checks whether the specified USART flag is set or not.
USART_ClearFlag	Clears the USARTx pending flags.
USART_GetITStatus	Checks whether the specified USART interrupt has occurred or not.
USART_ClearITPendingBit	Clears the USARTx interrupt pending bits.

UART - konfiguracja

```
typedef struct
{
  u32 USART_BaudRate;
  u16 USART_WordLength;
  u16 USART_StopBits;
  u16 USART_Parity;
  u16 USART_HardwareFlowControl;
  u16 USART_Mode;
} USART_InitTypeDef;
```

USART_Parity	Description
USART_Parity_No	Parity Disable
USART_Parity_Even	Even Parity
USART_Parity_Odd	Odd Parity

USART_WordLength	Description
USART_WordLength_8b	8 bits Data
USART_WordLength_9b	9 bits Data

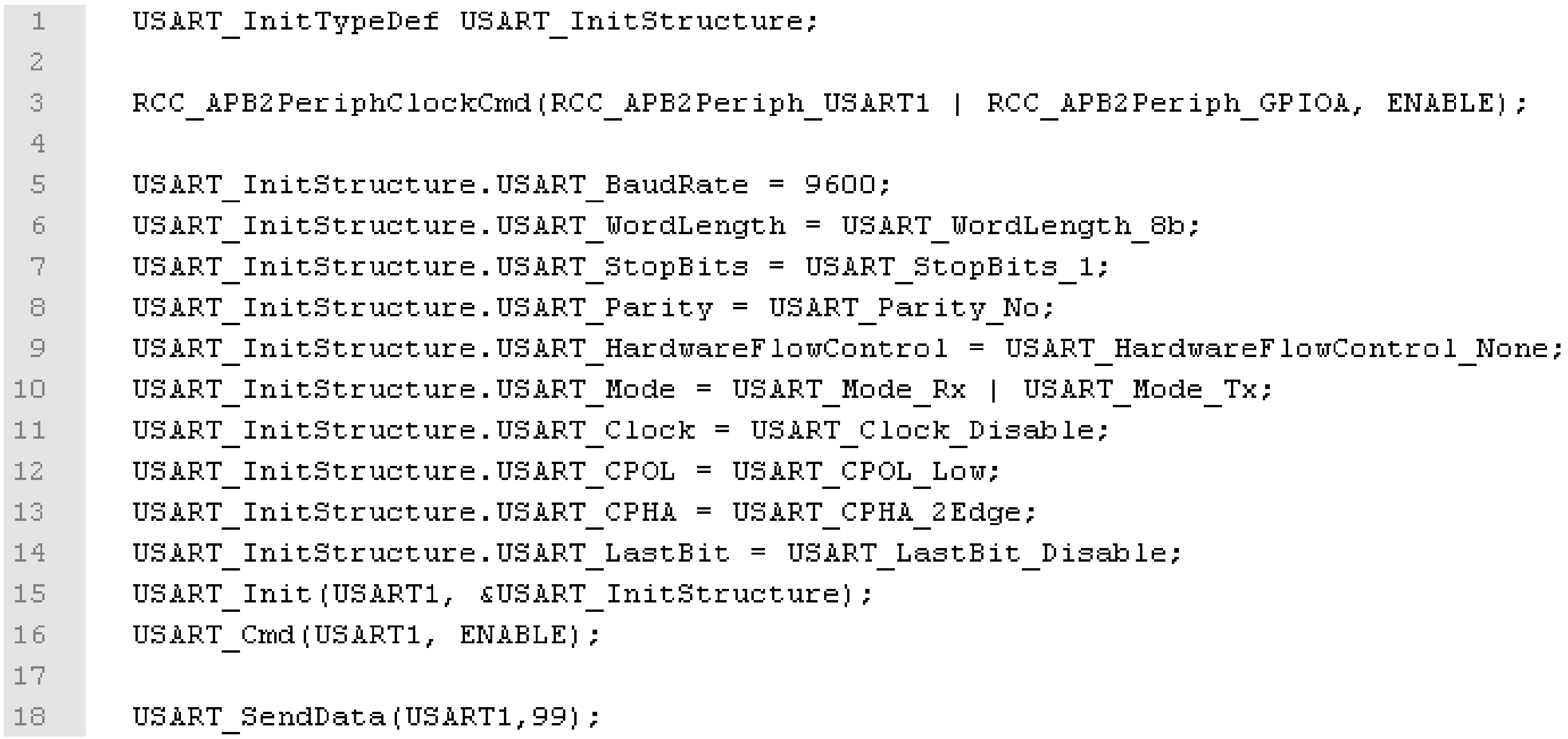
USART_HardwareFlowControl	Description
USART_HardwareFlowControl_None	HFC Disabled
USART_HardwareFlowControl_RTS	RTS enabled
USART_HardwareFlowControl_CTS	CTS enabled
USART_HardwareFlowControl_RTS_CTS	RTS and CTS enabled

USART_StopBits	Description
USART_StopBits_1	1 stop bit is transmitted at the end of frame
USART_StopBits_0_5	0.5 stop bit is transmitted at the end of frame
USART_StopBits_2	2 stop bits are transmitted at the end of frame
USART_StopBits_1_5	1.5 stop bit is transmitted at the end of frame

USART_Mode	Description
USART_Mode_Tx	Transmit enabled
USART_Mode_Rx	Receive enabled

KNSW

Przykład



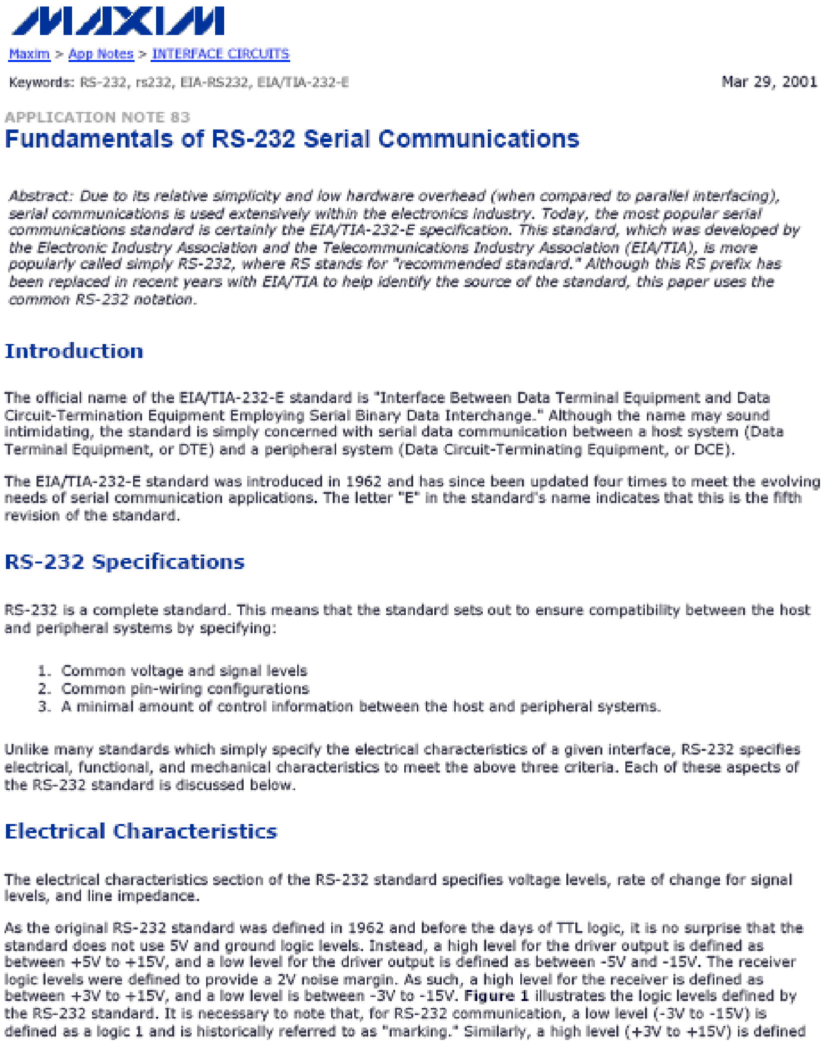
```
USART_InitTypeDef USART_InitStructure;

RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1 | RCC_APB2Periph_GPIOA, ENABLE);

USART_InitStructure.USART_BaudRate = 9600;
USART_InitStructure.USART_WordLength = USART_WordLength_8b;
USART_InitStructure.USART_StopBits = USART_StopBits_1;
USART_InitStructure.USART_Parity = USART_Parity_No;
USART_InitStructure.USART_HardwareFlowControl = USART_HardwareFlowControl_None;
USART_InitStructure.USART_Mode = USART_Mode_Rx | USART_Mode_Tx;
USART_InitStructure.USART_Clock = USART_Clock_Disable;
USART_InitStructure.USART_CPOL = USART_CPOL_Low;
USART_InitStructure.USART_CPHA = USART_CPHA_2Edge;
USART_InitStructure.USART_LastBit = USART_LastBit_Disable;
USART_Init(USART1, &USART_InitStructure);
USART_Cmd(USART1, ENABLE);

USART_SendData(USART1,99);
```

RS-232

- Zastosowanie:
  - PC, mierniki, kasy fiskalne, drukarki, wagi, motoryzacja, sieci przemysłowe itp.
  - moduły GSM/GPS
- Układy:
  - Maxim, TI, ST, Sipex, National itp.
  - Układy na makietach: ST3232 lub SP3232ECP

KNSW

Datasheets



Maxim > App Notes > INTERFACE CIRCUITS

Keywords: RS-232, rs232, EIA-RS232, EIA/TIA-232-E

Mar 29, 2001

APPLICATION NOTE 83

Fundamentals of RS-232 Serial Communications

Abstract: Due to its relative simplicity and low hardware overhead (when compared to parallel interfacing), serial communications is used extensively within the electronics industry. Today, the most popular serial communications standard is certainly the EIA/TIA-232-E specification. This standard, which was developed by the Electronic Industry Association and the Telecommunications Industry Association (EIA/TIA), is more popularly called simply RS-232, where RS stands for "recommended standard." Although this RS prefix has been replaced in recent years with EIA/TIA to help identify the source of the standard, this paper uses the common RS-232 notation.

Introduction

The official name of the EIA/TIA-232-E standard is "Interface Between Data Terminal Equipment and Data Circuit-Termination Equipment Employing Serial Binary Data Interchange." Although the name may sound intimidating, the standard is simply concerned with serial data communication between a host system (Data Terminal Equipment, or DTE) and a peripheral system (Data Circuit-Terminating Equipment, or DCE).

The EIA/TIA-232-E standard was introduced in 1962 and has since been updated four times to meet the evolving needs of serial communication applications. The letter "E" in the standard's name indicates that this is the fifth revision of the standard.

RS-232 Specifications

RS-232 is a complete standard. This means that the standard sets out to ensure compatibility between the host and peripheral systems by specifying:

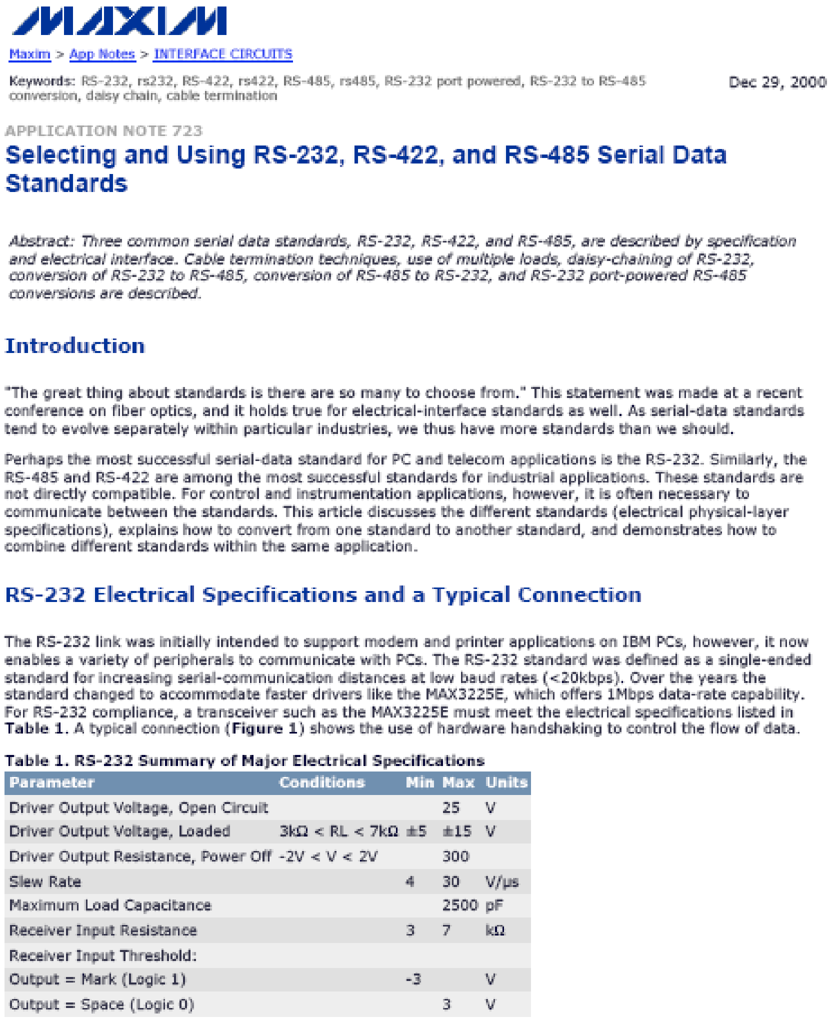
1. Common voltage and signal levels
2. Common pin-wiring configurations
3. A minimal amount of control information between the host and peripheral systems.

Unlike many standards which simply specify the electrical characteristics of a given interface, RS-232 specifies electrical, functional, and mechanical characteristics to meet the above three criteria. Each of these aspects of the RS-232 standard is discussed below.

Electrical Characteristics

The electrical characteristics section of the RS-232 standard specifies voltage levels, rate of change for signal levels, and line impedance.

As the original RS-232 standard was defined in 1962 and before the days of TTL logic, it is no surprise that the standard does not use 5V and ground logic levels. Instead, a high level for the driver output is defined as between +5V to +15V, and a low level for the driver output is defined as between -5V and -15V. The receiver logic levels were defined to provide a 2V noise margin. As such, a high level for the receiver is defined as between +3V to +15V, and a low level is between -3V to -15V. **Figure 1** illustrates the logic levels defined by the RS-232 standard. It is necessary to note that, for RS-232 communication, a low level (-3V to -15V) is defined as a logic 1 and is historically referred to as "marking." Similarly, a high level (+3V to +15V) is defined



Maxim > App Notes > INTERFACE CIRCUITS

Keywords: RS-232, rs232, RS-422, rs422, RS-485, rs485, RS-232 port powered, RS-232 to RS-485 conversion, daisy chain, cable termination

Dec 29, 2000

APPLICATION NOTE 723

Selecting and Using RS-232, RS-422, and RS-485 Serial Data Standards

Abstract: Three common serial data standards, RS-232, RS-422, and RS-485, are described by specification and electrical interface. Cable termination techniques, use of multiple loads, daisy-chaining of RS-232, conversion of RS-232 to RS-485, conversion of RS-485 to RS-232, and RS-232 port-powered RS-485 conversions are described.

Introduction

"The great thing about standards is there are so many to choose from." This statement was made at a recent conference on fiber optics, and it holds true for electrical-interface standards as well. As serial-data standards tend to evolve separately within particular industries, we thus have more standards than we should.

Perhaps the most successful serial-data standard for PC and telecom applications is the RS-232. Similarly, the RS-485 and RS-422 are among the most successful standards for industrial applications. These standards are not directly compatible. For control and instrumentation applications, however, it is often necessary to communicate between the standards. This article discusses the different standards (electrical physical-layer specifications), explains how to convert from one standard to another standard, and demonstrates how to combine different standards within the same application.

RS-232 Electrical Specifications and a Typical Connection

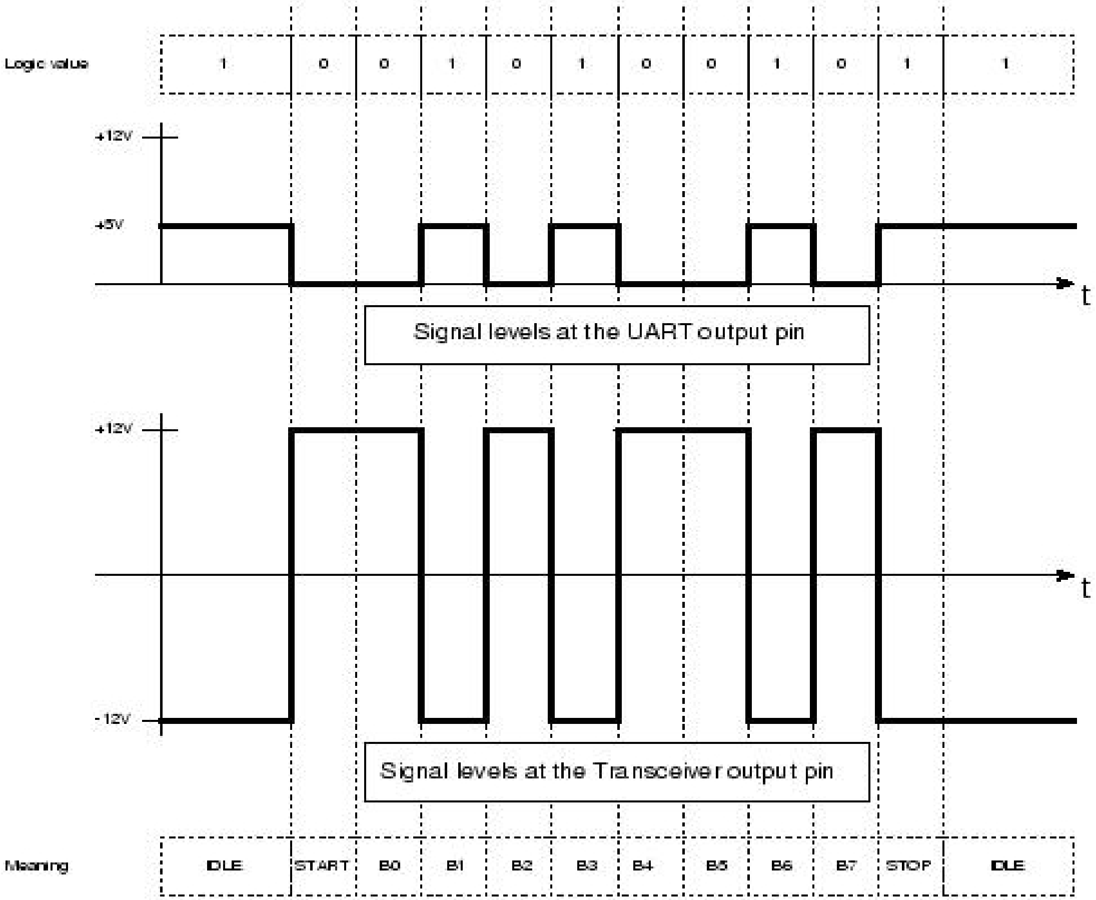
The RS-232 link was initially intended to support modem and printer applications on IBM PCs, however, it now enables a variety of peripherals to communicate with PCs. The RS-232 standard was defined as a single-ended standard for increasing serial-communication distances at low baud rates (<20kbps). Over the years the standard changed to accommodate faster drivers like the MAX3225E, which offers 1Mbps data-rate capability. For RS-232 compliance, a transceiver such as the MAX3225E must meet the electrical specifications listed in **Table 1**. A typical connection (**Figure 1**) shows the use of hardware handshaking to control the flow of data.

Table 1. RS-232 Summary of Major Electrical Specifications

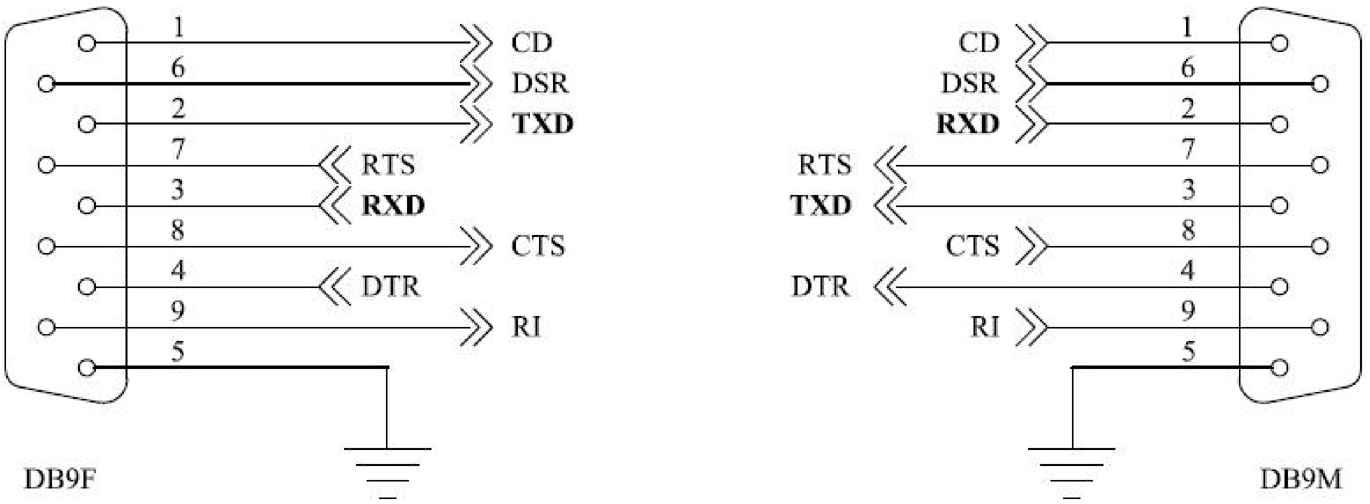
Parameter	Conditions	Min	Max	Units
Driver Output Voltage, Open Circuit			25	V
Driver Output Voltage, Loaded	3kΩ < RL < 7kΩ	±5	±15	V
Driver Output Resistance, Power Off	-2V < V < 2V		300	
Slew Rate		4	30	V/µs
Maximum Load Capacitance			2500	pF
Receiver Input Resistance		3	7	kΩ
Receiver Input Threshold:				
Output = Mark (Logic 1)		-3		V
Output = Space (Logic 0)			3	V

Sygnały logiczne

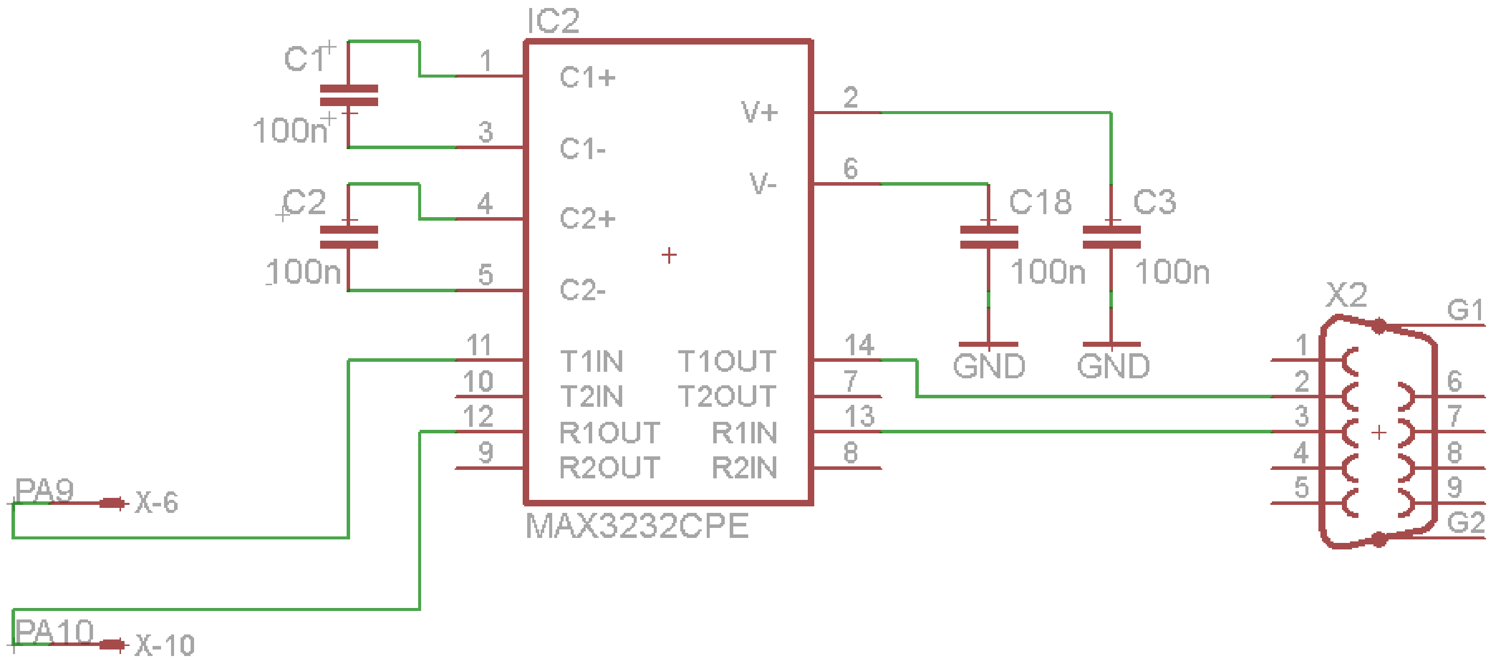
- USART:
  - „0”→ 0V
  - „1”→ 5V
- RS232C
  - „0”→ 3V – 12V
  - „1”→ -3V – -12V

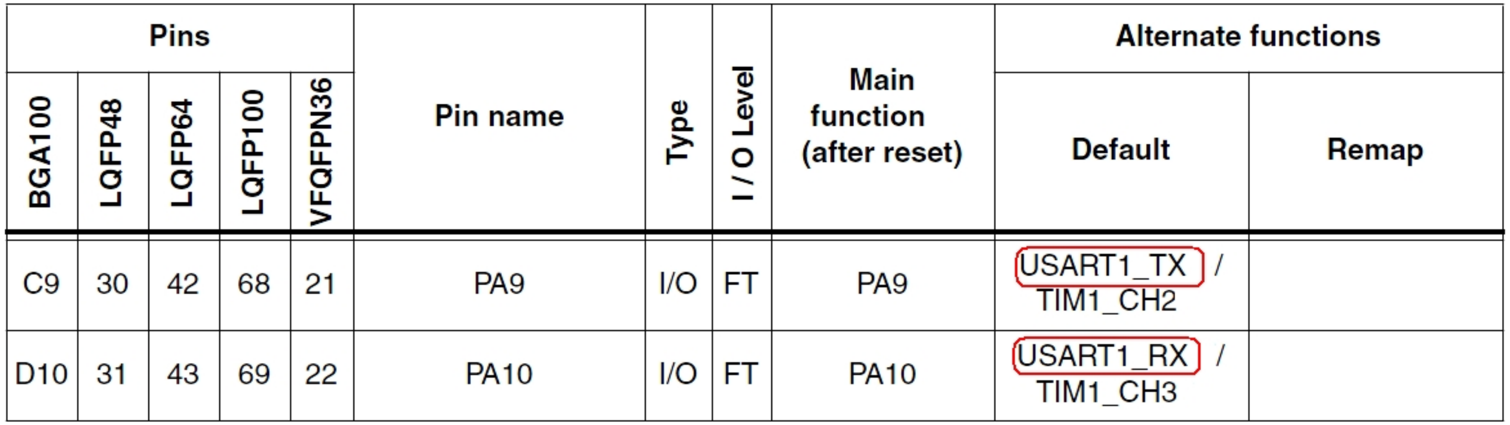


DB9F, DB9M


1
CD
6
DSR
2
TXD
7
RTS
3
RXD
8
CTS
4
DTR
9
RI
5
DB9F
CD
1
DSR
6
RXD
2
RTS
7
TXD
3
CTS
8
DTR
4
RI
9
5
DB9M

Przykład RS-232

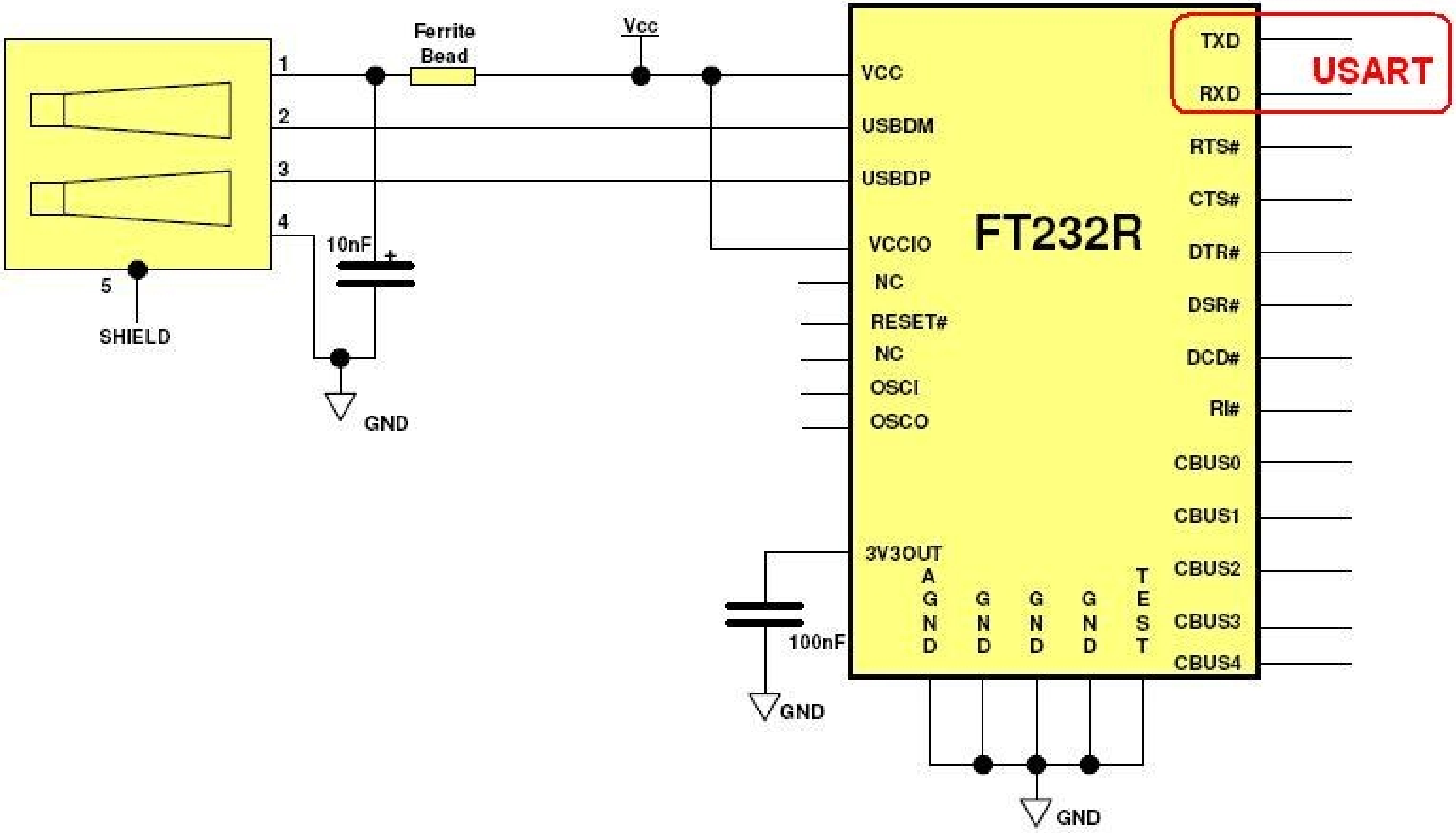




Pins					Pin name	Type	I / O Level	Main function (after reset)	Alternate functions	
BGA100	LQFP48	LQFP64	LQFP100	VFQFPN36					Default	Remap
C9	30	42	68	21	PA9	I/O	FT	PA9	USART1_TX / TIM1_CH2	
D10	31	43	69	22	PA10	I/O	FT	PA10	USART1_RX / TIM1_CH3	

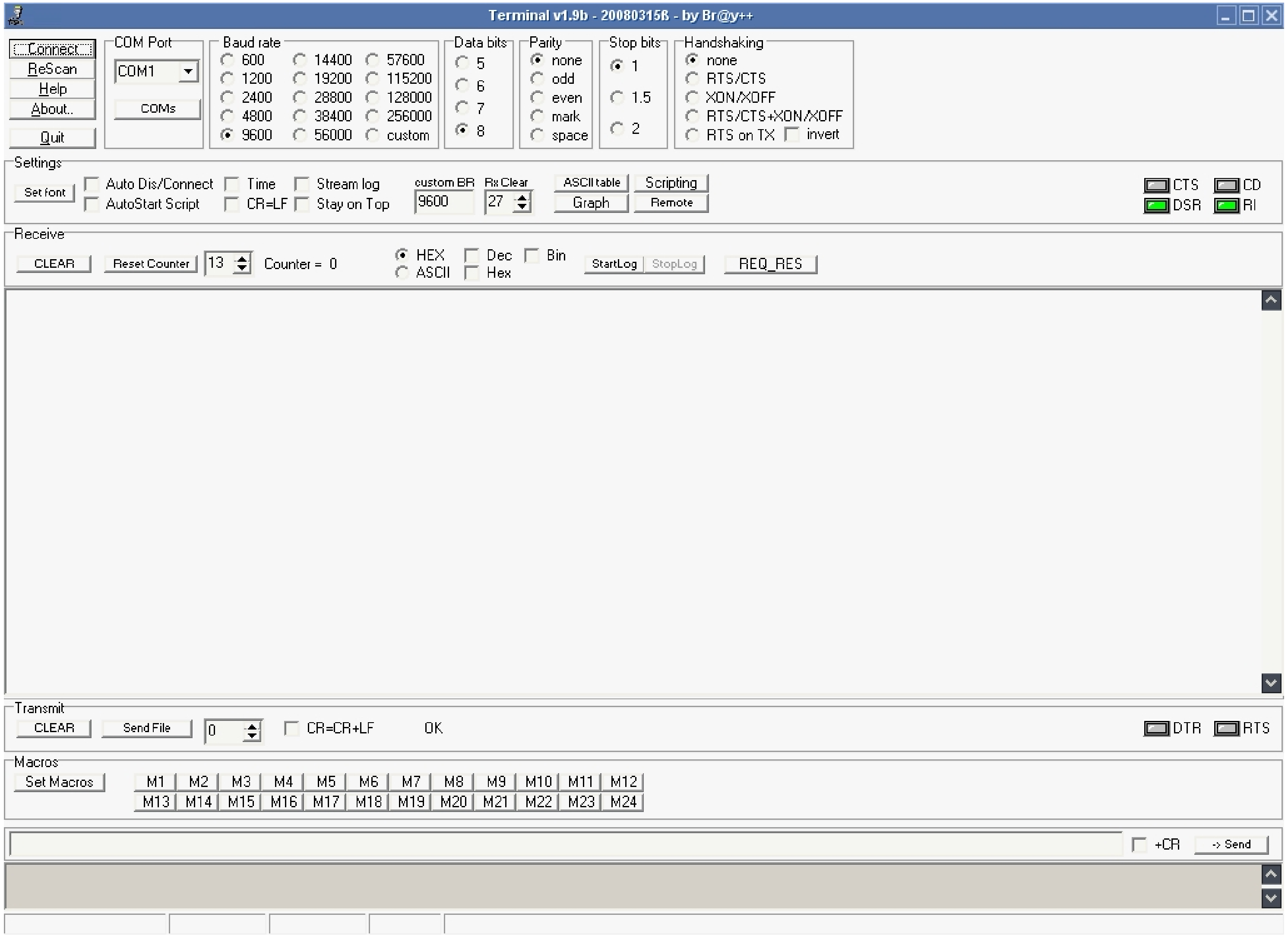
KNSW

FT232 – Virtual COM Port



KNSW

Terminal v1.9b



Terminal v1.9b - 20080315ß - by Br@y++

COM Port: COM1

Baud rate: ○ 600 ○ 1200 ○ 2400 ○ 4800 ● 9600 ○ 14400 ○ 19200 ○ 28800 ○ 38400 ○ 56000 ○ 57600 ○ 115200 ○ 128000 ○ 256000 ○ custom

Data bits: ○ 5 ○ 6 ○ 7 ● 8

Parity: ● none ○ odd ○ even ○ mark ○ space

Stop bits: ● 1 ○ 1.5 ○ 2

Handshaking: ● none ○ RTS/CTS ○ XON/XOFF ○ RTS/CTS+XON/XOFF ○ RTS on TX □ invert

Settings

□ Auto Dis/Connect □ AutoStart Script □ Time □ CR=LF □ Stream log □ Stay on Top

custom BR: 9600 Rx Clear: 27

Receive

13 Counter = 0

● HEX ○ ASCII □ Dec □ Hex □ Bin

Transmit

0 □ CR=CR+LF OK

Macros

□ +CR

Koniec