# AVR32100: Using the AVR32 USART

32-bit AVR® Microcontrollers

Application Note

## Features

- **Supports character length from 5 to 9 bits**
- **Interrupt Generation**
- **Parity, Framing and Overrun Error Detection**
- **Programmable Baud Rate Generator**
- **Line Break Generation and Detection**
- **Protocol ISO7816 T=0 and T=1**
- **Modem, Handshaking (Hardware and Software) and RS485 signals**
- **Infrared Data Association (IrDA) 115.2 kbps**
- **Test modes for debugging**
    - **Automatic echo, local loopback and remote loopback**
- **Odd, even, mark, space or no parity bit**
- **1, 1.5 or 2 stop bits**
    - **Also provide support for a time guard up to 255 bit periods long**

## 1 Introduction

The AVR®32 microcontroller has independent USARTs. These are widely configurable, and can be set to operate in several different modes. This application note describes these modes and provides the drivers to communicate with the USART.

Rev. 32006A-AVR-04/06

# 2 Operation

## 2.1 Operation mode overview

When the USART is initialized several parameters are passed, one of which select the operation mode of the USART. The USART supports the RS232, RS485, modem and ISO7816 (T=0 and T=1) protocols, along with software and hardware handshaking, and IrDA.

### 2.1.1 Normal mode (RS232)

Most devices using serial communication support the RS232 protocol. Two devices are connected using a separate wire for transmit and receive in order to achieve full duplex communication. Usually a ground wire is included for a total of three wires. The characters transmitted may be from 5 to 9 bits, and are wrapped with a start bit and at least one stop bit. An additional parity bit may also be included to form the packet transmitted.

### 2.1.2 Handshaking

There are two available types of handshaking; software and hardware. In the software mode two reserved characters are sent from the receiver to the transmitter, one signalizing that the receivers buffers are full, causing the transmitter to stop transmitting. The other tells the transmitter to resume the transmission. Hardware handshaking require extra wiring. One wire is used to make a Request To Send (RTS), another to signal Clear To Send (CTS). Whenever a character is to be transmitted the source will set its RTS high, when the CTS is received from the target, the character is transmitted.

### 2.1.3 Multidrop mode (RS485)

The main difference between RS485 and RS232, is the ability to connect several devices on the same wires, forming a serial bus. When in RS485 mode the USART is able to tristate (disconnect) it's output, thus granting some other device the usage of the bus. In most configurations all but one of the devices on the bus, are slaves. The master will address the other devices using a special address character. If a reply is anticipated, release the bus for some time.

### 2.1.4 ISO7816 protocol

The ISO7816 is a half-duplex protocol for communication with smart cards. The USART must be configured to provide a clock signal to the smart card (enable clock output). Since only one wire is used for communication, the USART can not enable both the transmitter and the receiver simultaneously. After initialization, both are disabled. In order to transmit, usart_send_enable() should be called. To receive data, usart_send_disable() must be run. The ISO7816 protocol is subdivided into two modes; T=0 and T=1. In the T=1 mode, the USART will communicate like in normal mode, but only half-duplex. The T=0 mode provide some additional features. In this mode the receiver will send a NACK, Non Acknowledge, bit to report error in the transmission. The max_it field gives the maximum number of re-sends the transmitter will perform before giving up. On the receiver side, this field gives the maximum number of NACKs to send for each character. When the maximum number of repetitions has been reached on either side, a flag is asserted (see usart_max_it_reached()). To avoid repetitive transmission, the inack bit should be set in the receiver.

### 2.1.5 IrDA protocol

The USART supports half-duplex wireless communication by providing an IrDA mode. This mode complies with IrDA specification version 1.1, and supports data rates from 2.4 kbps to 115.2 kbps. When transmitting, the RZI modulation scheme is used. Logical “0” is transmitted by emitting light for 3/16 of a bit period, while logical “1” is represented by no emission for a bit period. On the receiving end received light will cause RXD to drop low, a demodulator will use a counter to separate 1 from 0. Whenever a falling edge is detected on the RXD pin, the value set by `IrDA_filter` in the options struct is loaded into a counter. This counter decreases with one at every cycle of the master clock. If no rising edge has been detected by the time the counter reaches 0, the input is read as logical “0”. However, if a rising edge is detected, the counter is reloaded, and the signal read as logical “1”. (For IrDA reception the `filter` value in the options struct will have to be set).

### 2.1.6 Modem protocol

When set to the modem mode, the USART enables additional inputs and outputs. The USART behaves as a Data Terminal Equipment, DTE, as it drives DTR and RTS and can detect change on DSR, DCD, CTS and RI. Special modem mode functions are included for this.

# 3 Initialization

Before using the USART it has to be initialized. This is accomplished by running the appropriate initialization function. These function sets up the USART according to the arguments given to them. If some of the arguments do not correspond to a valid operational mode, an error will be returned. In this case no initialization will be performed, and the USART should not be expected to work at any level. Reinitializing the USART will erase any previous initialization data, and completely reset the USART. As a result, all needed parameters must be passed on every reset. If a parameter must be changed, a complete re-initialization should be performed. Different initialization functions are available for each of the different operation modes available for the USART.

### 3.1.1 Mode options

When initializing the USART to operate in i.e RS232 mode (and most of the other asynchronous modes) a `usart_options_t` struct has to be passed. This struct contains settings that control the communication. All the members of the struct must be set to a suitable value before passing it to an initialization function or an error will be returned, and the USART will not work as expected.

Please note that this struct is used for every usart mode, except the ISO7816 mode, which has its own initialization different from the other modes.

Calling an initialization function for either one of the modes available in the USART module will initialize the appropriate USART. The initialization function will return 0 upon success.

Each USART is uniquely initialized and can only support one mode at any given time. If a mode change is required, the previous mode will no longer be supported by the USART.

# 4 Functions

All the functions available in the driver are standalone functions. In many situations these functions may be implemented in another fashion to improve performance. This could incorporate interrupts, performance counters and timers.

To use an USART in any mode, the correct mode must first be initialized. If an USART is initialized, only functions associated with that mode should be used. In most cases the functions themselves can be called from any state or function in an application. There are a few functions that require a specific function to be called in advance. This is clearly stated for these functions.

## 4.1 I/O functions

The USART have several functions, which can operate independently on each of the USARTs. Calling these functions is not depending on the function or state of a certain program. This means that i.e. an interrupt can be set to trigger reading of an USART. All diagrams are therefore indifferent to the current running function or state. These functions are used in all 5 of the different modes in each of the USART instances available.

### 4.1.1 Read and write

After one or more USARTs have been initialized, functions for transmitting and receiving data are needed. Reception of messages can be either setup as interrupts or by polling the transmit buffer. Writing or reading a character from the usart consists of calling the appropriate function, either `usart_read_char` or `usart_write_char`.

### 4.1.2 Break

A break state can be entered by calling `usart_start_break()` and no transmission on the USART can be made. To unlock the break state, the function `usart_stop_break()` must be called. To test whether the USART is in the break state or not, the function `usart_test_brk()` can be called. This function will return either true or false depending on the current state of the specified USART. All these functions take an USART base address as an argument.

## 4.2 Timeout handling

All USARTs have support for timeout handling. There are 3 timeout functions available for each USART:

- `usart_start_timeout_now( usart_t usart, int timeout )`
  - This function will start a timeout immediately.
- `usart_start_timeout_after_transfer(usart_t usart, int timeout)`
  - Timeout will start automatically after the next reception of a character of the actual USART.
- `usart_has_timed_out( usart_t usart )`
  - Returns true or false.

## 4.3 Error status functions

There are 4 error functions available for each USART:

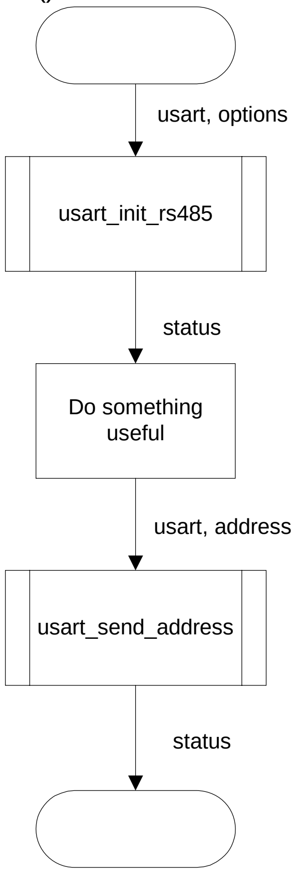
- `usart_parity_error( usart_t usart )`
- `usart_framing_error( usart_t usart )`
- `usart_overrun_error( usart_t usart )`
- `usart_reset_status( usart_t usart )`

The three first functions will return true or false if an error of the given type has occurred since the last reset of the actual USART. The function `usart_reset_status()` will clear all errors of the given USART that has occurred since the last reset of the USART.

## 4.4 Addressing functions (multidrop mode)

If using rs485, each node must have an address associated with the actual node. To set up this address, the function `usart_send_address()` is used. This address takes the USART base address and the address of the node as arguments. To successfully set an address, the function `usart_init_rs485()` must be called first. See chapter 3 for more details.

**Figure 1.** usart_send_address()



## 4.5 ISO7816 mode functions

A series of functions are specified for the ISO7815 mode. All these functions require the base address for the actual USART to function appropriately. These functions can furthermore be connected independently of each other to suit specific implementation needs.

### 4.5.1 Error functions

- usart_number_of_errors( usart_t usart )
    - returns the number of transfer errors. The counter overflows at 255 and is reset upon each read access.
- usart_max_iterations_reached( usart_t usart )
    - Returns true if the maximum number of resends (iterations) have been reached.
- usart_reset_iteration_counter( usart_t usart )
    - Resets the iteration bit described above.
- usart_reset_nack( usart_t usart )
    - Resets the nack bit for the USART.

Please note that the function `usart_number_of_errors()` utilizes hardware to keep track of the number of errors that might have occurred. Due to the fact that this counter overflows at 255 errors, design considerations should be made to ensure that the counter does not overflow and hides possible overflows.

### 4.5.2 Transmission functions

- usart_send_enable( usart_t usart )
    - Enables transmissions and disables receiver
- usart_send_disable( usart_t usart )
    - Enables the receiver, disables the transmitter

## 4.6 Modem mode functions

For modem communication a series of function are available. The functions are flexible and can be used in any order necessary to satisfy implementation specifications.

- usart_modem_pins()
    - Returns the bitfield for all changes to the modem i/o pins.
- usart_set_dtr()
    - Change *data terminal ready*. Takes the new pin_value as an argument.
- usart_set_rts()
    - This function controls the *request to send* output. The new value is sent as an argument to the function.

# 5 Package information

Included with the application note is a driver package. This package contains drivers, example code and documentation.

## 5.1 Drivers

Drivers are available in the package. These drivers are written to be independent of a specific compiler and are successfully tested on gcc and IAR Embedded Workbench.

## 5.2 Examples

Examples are available from the corresponding driver package. All functionality is divided into libraries and an example that utilizes the library.

## 5.3 Documentation

Function specific documentation is available in the package. Refer to readme.html in the source code directory.

## Atmel Corporation

2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

## Regional Headquarters

***Europe***
Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
Tel: (41) 26-426-5555
Fax: (41) 26-426-5500

***Asia***
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

***Japan***
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

## Atmel Operations

***Memory***
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

***Microcontrollers***
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
Tel: (33) 2-40-18-18-18
Fax: (33) 2-40-18-19-60

***ASIC/ASSP/Smart Cards***
Zone Industrielle
13106 Rousset Cedex, France
Tel: (33) 4-42-53-60-00
Fax: (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
Tel: (44) 1355-803-000
Fax: (44) 1355-242-743

***RF/Automotive***
Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
Tel: (49) 71-31-67-0
Fax: (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

***Biometrics/Imaging/Hi-Rel MPU/***
***High Speed Converters/RF Datacom***
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
Tel: (33) 4-76-58-30-00
Fax: (33) 4-76-58-34-80

---

***Literature Requests***
www.atmel.com/literature