8-bit AVR Microcontrollers

AVR306: Using the AVR USART on tinyAVR and megaAVR devices

APPLICATION NOTE

Introduction

This application note provides code examples to use the USART/UART on AVR devices in asynchronous mode. Most of the tinyAVR® and megaAVR® devices have USART peripheral which can perform both asynchronous and synchronous transfers.

For AVR devices with UART peripheral which do not have capability to perform synchronous serial data transfer, separate examples are provided in the zip file available with this application note.

Features

- Setup and use of the AVR USART/UART in asynchronous mode
- Code examples for polled and interrupt controlled USART/UART

Table of Contents

1. USART Transfer Methods

There are two methods in which the CPU transfers data through a USART/UART - Polled Transfer and Interrupt Controlled Transfer.

Table 1-1. Properties of Polled/Interrupt Controlled USART Routines

Polled USART	Interrupt Controlled USART
Compact code	Reasonable code size
Application busy while communicating	Application free while communicating

1.1. Polled Transfer

The application continuously checks the UDRE bit in the USART Status Register to identify when the USART has finished sending a byte. Then the next byte can be loaded to the data register. When receiving data, the application continuously checks the RXC bit in the USART Status Register to identify when the USART has completed receiving a byte. In this case, the CPU waits for the reception of a byte and then proceed with further processing.

1.2. Interrupt Controlled Transfer

In interrupt controlled transfer, the USART generates an interrupt when the USART has finished transmitting or receiving a byte. Thus the CPU can perform other functions while the USART module transmit or receive a byte. Interrupts corresponding to receive and transmit functionality of the USART module should be enabled for interrupt controlled transfers. In addition to this the Global Interrupt Enable bit in SREG must be set for the interrupts to be enabled.

2. BAUD Rate Generator

The USART Baud Rate Register (UBRRxx) and the down-counter connected to it function as a programmable prescaler or baud rate generator. The down-counter, running at system clock (f_{osc}), is loaded with the UBRRxx value each time the counter has counted down to zero or when the UBRRxx register is written. A clock is generated each time the counter reaches zero.

3. Examples of Baud Rate Setting

The following table contains equations for calculating the baud rate (in bits per second) and for calculating the UBRR value for each mode of operation using an internally generated clock source.

Table 3-1. Equations for Calculating Baud Rate Register Setting

Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRR Value
Asynchronous Normal mode (U2X = 0)	$\mathrm{BAUD} = \frac{f_{\mathrm{OSC}}}{16(\mathbf{UBRR}+1)}$	$\mathbf{UBRR} = \frac{f_{\mathrm{OSC}}}{16\mathrm{BAUD}} - 1$
Asynchronous Double Speed mode (U2X = 1)	$\mathrm{BAUD} = \frac{f_{\mathrm{OSC}}}{8(\mathbf{UBRR}+1)}$	$\mathbf{UBRR} = \frac{f_{\mathrm{OSC}}}{8\mathrm{BAUD}} - 1$
Synchronous Master mode	$\mathrm{BAUD} = \frac{f_{\mathrm{OSC}}}{2(\mathbf{UBRR}+1)}$	$\mathbf{UBRR} = \frac{f_{\mathrm{OSC}}}{2\mathrm{BAUD}} - 1$

Note: 1. The baud rate is defined to be the transfer rate in bit per second (bps).

BAUD	Baud rate (in bits per second, bps).
f_{OSC}	System oscillator clock frequency.
UBRR	Contents of the UBRRH and UBRRL Registers, (0-4095).

For standard crystal and resonator frequencies, the most commonly used baud rates for asynchronous operation can be generated by using the UBRR settings as listed in the following table. For details on clock options supported by a device, refer the device datasheet.

UBRR values which yield an actual baud rate differing less than 0.5% from the target baud rate, are bold in the table. Higher error ratings are acceptable, but the Receiver will have less noise resistance when the error ratings are high, especially for large serial frames. The error values are calculated using the following equation:

$$Error\left[\%\right] = \left(\frac{\mathrm{BaudRate}_{\mathrm{Closest\ Match}}}{\mathrm{BaudRate}} - 1\right) \times 100\,\%$$

Table 3-2. Examples of UBRR Settings for Commonly Used Oscillator Frequencies

Baud Rate [bps]	f_{osc} = 1.0000MHz				f_{osc} = 1.8432MHz				f_{osc} = 2.0000MHz			
	U2X = 0		U2X = 1		U2X= 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	25	0.2%	51	0.2%	47	0.0%	95	0.0%	51	0.2%	103	0.2%
4800	12	0.2%	25	0.2%	23	0.0%	47	0.0%	25	0.2%	51	0.2%
9600	6	-7.0%	12	0.2%	11	0.0%	23	0.0%	12	0.2%	25	0.2%
14.4k	3	8.5%	8	-3.5%	7	0.0%	15	0.0%	8	-3.5%	16	2.1%
19.2k	2	8.5%	6	-7.0%	5	0.0%	11	0.0%	6	-7.0%	12	0.2%
28.8k	1	8.5%	3	8.5%	3	0.0%	7	0.0%	3	8.5%	8	-3.5%

Baud Rate [bps]	f_{osc} = 1.0000MHz				f_{osc} = 1.8432MHz				f_{osc} = 2.0000MHz			
	U2X = 0		U2X = 1		U2X= 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
38.4k	1	-18.6%	2	8.5%	2	0.0%	5	0.0%	2	8.5%	6	-7.0%
57.6k	0	8.5%	1	8.5%	1	0.0%	3	0.0%	1	8.5%	3	8.5%
76.8k	–	–	1	-18.6%	1	-25.0%	2	0.0%	1	-18.6%	2	8.5%
115.2k	–	–	0	8.5%	0	0.0%	1	0.0%	0	8.5%	1	8.5%
230.4k	–	–	–	–	–	–	0	0.0%	–	–	–	–
250k	–	–	–	–	–	–	–	–	–	–	0	0.0%
Max(1)	62.5kbps		125kbps		115.2kbps		230.4kbps		125kbps		250kbps	

Note: 1. UBRR = 0, Error = 0.0%

Table 3-3. Examples of UBRR Settings for Commonly Used Oscillator Frequencies (Continued)

Baud Rate [bps]	f_{osc} = 3.6864MHz				f_{osc} = 4.0000MHz				f_{osc} = 7.3728MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	95	0.0%	191	0.0%	103	0.2%	207	0.2%	191	0.0%	383	0.0%
4800	47	0.0%	95	0.0%	51	0.2%	103	0.2%	95	0.0%	191	0.0%
9600	23	0.0%	47	0.0%	25	0.2%	51	0.2%	47	0.0%	95	0.0%
14.4k	15	0.0%	31	0.0%	16	2.1%	34	-0.8%	31	0.0%	63	0.0%
19.2k	11	0.0%	23	0.0%	12	0.2%	25	0.2%	23	0.0%	47	0.0%
28.8k	7	0.0%	15	0.0%	8	-3.5%	16	2.1%	15	0.0%	31	0.0%
38.4k	5	0.0%	11	0.0%	6	-7.0%	12	0.2%	11	0.0%	23	0.0%
57.6k	3	0.0%	7	0.0%	3	8.5%	8	-3.5%	7	0.0%	15	0.0%
76.8k	2	0.0%	5	0.0%	2	8.5%	6	-7.0%	5	0.0%	11	0.0%
115.2k	1	0.0%	3	0.0%	1	8.5%	3	8.5%	3	0.0%	7	0.0%
230.4k	0	0.0%	1	0.0%	0	8.5%	1	8.5%	1	0.0%	3	0.0%
250k	0	-7.8%	1	-7.8%	0	0.0%	1	0.0%	1	-7.8%	3	-7.8%
0.5M	–	–	0	-7.8%	–	–	0	0.0%	0	-7.8%	1	-7.8%
1M	–	–	–	–	–	–	–	–	–	–	0	-7.8%
Max.(1)	230.4kbps		460.8kbps		250kbps		0.5Mbps		460.8kbps		921.6kbps	

Note: 1. UBRR = 0, Error = 0.0%

Table 3-4. Examples of UBRR Settings for Commonly Used Oscillator Frequencies (Continued)

Baud Rate [bps]	f_{osc} = 8.0000MHz				f_{osc} = 11.0592MHz				f_{osc} = 14.7456MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	207	0.2%	416	-0.1%	287	0.0%	575	0.0%	383	0.0%	767	0.0%
4800	103	0.2%	207	0.2%	143	0.0%	287	0.0%	191	0.0%	383	0.0%
9600	51	0.2%	103	0.2%	71	0.0%	143	0.0%	95	0.0%	191	0.0%
14.4k	34	-0.8%	68	0.6%	47	0.0%	95	0.0%	63	0.0%	127	0.0%
19.2k	25	0.2%	51	0.2%	35	0.0%	71	0.0%	47	0.0%	95	0.0%
28.8k	16	2.1%	34	-0.8%	23	0.0%	47	0.0%	31	0.0%	63	0.0%
38.4k	12	0.2%	25	0.2%	17	0.0%	35	0.0%	23	0.0%	47	0.0%
57.6k	8	-3.5%	16	2.1%	11	0.0%	23	0.0%	15	0.0%	31	0.0%
76.8k	6	-7.0%	12	0.2%	8	0.0%	17	0.0%	11	0.0%	23	0.0%
115.2k	3	8.5%	8	-3.5%	5	0.0%	11	0.0%	7	0.0%	15	0.0%
230.4k	1	8.5%	3	8.5%	2	0.0%	5	0.0%	3	0.0%	7	0.0%
250k	1	0.0%	3	0.0%	2	-7.8%	5	-7.8%	3	-7.8%	6	5.3%
0.5M	0	0.0%	1	0.0%	–	–	2	-7.8%	1	-7.8%	3	-7.8%
1M	–	–	0	0.0%	–	–	–	–	0	-7.8%	1	-7.8%
Max.(1)	0.5Mbps		1Mbps		691.2kbps		1.3824Mbps		921.6kbps		1.8432Mbps	

Note: 1. UBRR = 0, Error = 0.0%

Table 3-5. Examples of UBRR Settings for Commonly Used Oscillator Frequencies (Continued)

Baud Rate [bps]	f_{osc} = 16.0000MHz				f_{osc} = 18.4320MHz				f_{osc} = 20.0000MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	416	-0.1%	832	0.0%	479	0.0%	959	0.0%	520	0.0%	1041	0.0%
4800	207	0.2%	416	-0.1%	239	0.0%	479	0.0%	259	0.2%	520	0.0%
9600	103	0.2%	207	0.2%	119	0.0%	239	0.0%	129	0.2%	259	0.2%
14.4k	68	0.6%	138	-0.1%	79	0.0%	159	0.0%	86	-0.2%	173	-0.2%
19.2k	51	0.2%	103	0.2%	59	0.0%	119	0.0%	64	0.2%	129	0.2%
28.8k	34	-0.8%	68	0.6%	39	0.0%	79	0.0%	42	0.9%	86	-0.2%
38.4k	25	0.2%	51	0.2%	29	0.0%	59	0.0%	32	-1.4%	64	0.2%
57.6k	16	2.1%	34	-0.8%	19	0.0%	39	0.0%	21	-1.4%	42	0.9%
76.8k	12	0.2%	25	0.2%	14	0.0%	29	0.0%	15	1.7%	32	-1.4%

Baud Rate [bps]	f_{osc} = 16.0000MHz				f_{osc} = 18.4320MHz				f_{osc} = 20.0000MHz			
	U2X = 0		U2X = 1		U2X = 0		U2X = 1		U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
115.2k	8	-3.5%	16	2.1%	9	0.0%	19	0.0%	10	-1.4%	21	-1.4%
230.4k	3	8.5%	8	-3.5%	4	0.0%	9	0.0%	4	8.5%	10	-1.4%
250k	3	0.0%	7	0.0%	4	-7.8%	8	2.4%	4	0.0%	9	0.0%
0.5M	1	0.0%	3	0.0%	–	–	4	-7.8%	–	–	4	0.0%
1M	0	0.0%	1	0.0%	–	–	–	–	–	–	–	–
Max.(1)	1Mbps		2Mbps		1.152Mbps		2.304Mbps		1.25Mbps		2.5Mbps	

Note: 1. UBRR = 0, Error = 0.0%

4. About Code Examples

There are three projects available in the zip file which comes with this application note, one for UART and two for USART. Each of these projects contains two source files (polling method and interrupt method). The projects contain the code example for polling method. To test interrupt method, the corresponding file can be added to the project after removing the existing source file.

UART based examples are tested for ATmega8515 device and USART based ones are tested on ATmega324PB and ATtiny104 . All Six files can be rebuild using avr-gcc or IAR Toolchain and can be used on any AVR ATtiny® and AVR ATmega® devices with minor changes if any such as register name or register bit name. Baudrate calculations in these examples are based on 8MHz and 1MHz (ATtiny104) CPU clock frequency. The main application stays in an infinite loop and sends back the data byte received.

The interrupt handling routines uses modulo 2n addressing of circular buffers for buffering incoming and outgoing data. The buffer sizes must be defined before using the routines. Set the UART_RX_BUFFER_SIZE and UART_TX_BUFFER_SIZE variables to the buffer size in bytes. These variables must be a power of 2 for the given example, if not, a compiler error message will be flagged.

Note: The interrupt handling for ATtiny104 does not use circular buffering incoming and outgoing data to reduce RAM usage.

5. Testing the Examples

UART examples based on ATmega8515 can be tested on STK600. Use the appropriate socket card and routing card to place the device on the STK600. The UART pins of this device are as follows:

Table 5-1. UART Pins Used in the Example

Signal	Pin Name on ATmega8515
RXD	PD0
TXD	PD1

1. Open the Atmel Studio project in Atmel Studio 7.
2. Rebuild the project and load the hex file into the device using an ISP programmer.
3. Use wired jumpers to connect PD0 to RXD pin of the RS232 SPARE connector on the STK600 and PD1 to TXD pin of the connector.
4. Connect an RS232 cable from STK600 to the PC and open a terminal application (9600 Baud, 8 data bits, 1 stop bit) to transfer characters.
5. Type characters on the terminal window. These will be echoed by the application running on the ATmega8515 device.

USART examples based on ATmega324PB can be tested on the ATmega324PB Xplained Pro. This board has an embedded debugger (EDBG) which provides a virtual COM port. Pins of USART1 are connected to this virtual COM port. To test the code,

1. Connect the ATmega324PB Xplained Pro to the PC.
2. Build the project and load the hex file into the device using EDBG as programmer.
3. Open a terminal application for the EDBG Virtual COM port (9600 Baud, 8 data bits, 1 stop bit).
4. Type characters on the terminal window. These will be echoed by the application running on the ATmega324PB device.

USART examples based on ATtiny104 can be tested on the ATTiny104 Xplained NANO. This board has an embedded debugger (mEDBG) which provides a virtual COM port. Pins of USART are connected to this virtual COM port. To test the code,

1. Connect the ATTiny104 Xplained NANO to the PC.
2. Build the project and load the hex file into the device using mEDBG as programmer.
3. Open a terminal application for the mEDBG Virtual COM port(2400 Baud, 8 data bits, 1 stop bit).
4. Type characters on the terminal window. These will be echoed by the application running on the ATtiny104 device.

Note:

1. Link which shows details on programming an Atmel microcontroller can be found in References.
2. Turn OFF local echo in the terminal application while testing these examples, the application will echo the received character.

6. References

1. AVR317: Using the USART on the ATmega48/88/168 as a SPI master
2. AVR054: Run-time calibration of the internal RC oscillator via the UART
3. STK600 Routing Cards and Socket Cards
4. ATmega324PB Xplained Pro User Guide
5. Atmel Studio - Programming Dialogue
6. ATTINY104-XNANO User Guide

7. Revision History

Doc Rev.	Date	Comments
1451C	04/2016	• Updated content to SDL format • Added new sections on BAUD rate settings and testing the examples • Included examples based on ATmega324PB Xplained Pro and ATtiny104 Xplained NANO
1451B	06/2002	Updated with new device
1451A	08/1999	Initial document release

Atmel Corporation 1600 Technology Drive, San Jose, CA 95110 USA **T:** (+1)(408) 441.0311 **F:** (+1)(408) 436.4200 | **www.atmel.com**