
AVR1522: XMEGA-A1 Xplained Training - XMEGA USART



Prerequisites

- Required knowledge
 - AVR1512: XMEGA-A1 Xplained Training – XMEGA Basics
 - AVR1514: XMEGA-A1 Xplained Training – XMEGA Direct Memory Access Controller
- Software prerequisites
 - Atmel® AVR Studio® 5
- Hardware prerequisites
 - XMEGA-A1 Xplained evaluation board
 - JTAGICE3 (or JTAGICE mkII or AVR ONE!)
- Estimated completion time:
 - 1.5 hours

1 Introduction

The USART (Universal Synchronous Asynchronous Receiver Transmitter) is the key element in serial communications between computers, terminals and other devices.

This training covers basic setup and use of the Atmel AVR® XMEGA® USART and the three tasks will demonstrate how to use the USART In polling-mode, interrupt mode and how to use the DMAC (Direct Memory Access Controller) to transfer data without CPU interaction.

**8-bit Atmel
Microcontrollers**

Application Note

Rev. 8408A-AVR-07/11





2 Overview

This training covers some of the Atmel XMEGA USART basic features:

Task 1: Polled mode

The first task shows how to set up the USART in polling mode. Some characters will be transferred in loop-back mode.

Task 2: Interrupt mode

This task shows how to use a driver to set up the USART. The driver has a ring buffer that makes life easier for the developer. Also the hardware buffer is shown in this task.

Task 3: DMAC

Atmel XMEGA introduces Direct Memory Access Controller (DMAC) for 8-bit processors. With the USART, DMAC is very useful allowing data to flow with nearly no CPU intervention. This task will show how to set up the USART with the DMAC.

3 Task 1: USART in polling mode

Using polling mode with the USART is especially useful when for example debugging the application or when steps in the program are expected to happen synchronously. In this task, we will set up the Xplain evaluation board to send data from USART to another USART. This is a good way to test the USART.

The goals for this task are that you know how to:

- Set up the USART in polling mode
- Send some characters in loop-back mode
- Verify that the transmission was successful



Todo

1. Locate the Atmel XMEGA- USART folder, open the `xmega_usart.avrsln` solution file and set Task 1 active by selecting it as StartUp project.
2. On the XMEGA-A1 Xplained connect a jumper or cable between TX and RX on J1.

3.1 Baud rate

The Baud rate is calculated by using the peripheral frequency (f_{PER}), the BSCALE and BSEL as parameters. The target frequency and peripheral frequency on the Atmel XMEGA is set to 2 MHz default. The BSEL bits are setting the baud rate, and the BSCALE is adding even more functionality, but is ignored at this stage. By setting BSCALE to 0, the BSEL can be found by Equation 3-1.

Equation 3-1. Equation for Calculating BSEL Value

$$BSEL = \frac{f_{PER}}{2^{BSCALE} \cdot 16 f_{BAUD}} - 1$$

3. Calculate and find the BSEL value for $f_{BAUD} = 9600$
4. Verify that the calculated value is the same as in task1.c
5. Compile the project and verify that there are no errors or warnings

3.2 Debugging the Polled USART

6. Look through the code and notice the comments. Try to understand what happens
7. Build the project and start a debug session (click the Play icon)
8. Add a breakpoint on the while-loop as seen in Figure 3-1
9. Add watches to the `Rx_Buf` and the `Tx_Buf`



10. Run the code (press F5)

11. Confirm that the `Rx_Buf` and the `Tx_Buf` are equal

12. Single step (F11) and check that no transfer error occurred (LEDs light up)

Figure 3-1: Add break-point in the last while-loop



```
while(1)
{
    if(TransferError)
    {
        // No light
        LEDPORT.OUT = 0xFF;
    }
    else
    {
        // Light
        LEDPORT.OUT = 0x00;
    }
}
```

4 Task 2: USART in interrupt mode

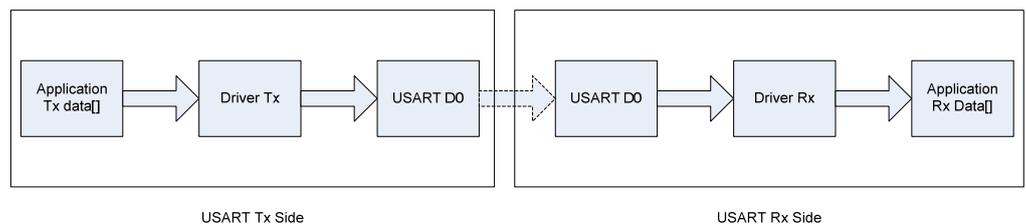
USART in interrupt mode will free CPU cycles since the microcontroller doesn't have to poll the transmit register to see if it is empty or poll the receive register to see if it contains new data. This way, the microcontroller can do other and more useful things than waiting, and increase use of its performance.

New to Atmel XMEGA are the three byte hardware buffers to keep data. The advantage of this buffer is to reduce occurrences of buffer overflow. Buffer overflow can occur, for instance, when interrupt service routines (ISRs) with higher priority starves out¹ other ISRs with lower priority.

A driver is used to set up a USART in interrupt mode. The driver also has a ring buffer implemented which will be explored briefly.

As seen from Figure 4-1, the application will send data to the driver which transfers it via the cable to the receiving USART. At the receiver, the driver sends the data to the application.

Figure 4-1: Data flow in task 2



The goals for this task are to:

- Know how to set up the Atmel XMEGA USART in interrupt mode
- Understand how to use a driver for the setup
- Take a quick look at the ring buffer in the driver
- Understand how the hardware buffer works



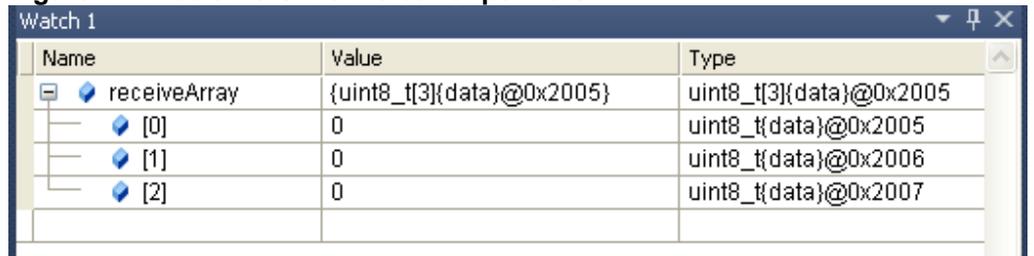
Todo

1. Locate the Atmel XMEGA- USART folder, open the `xmega_usart.avrsln` solution file and set Task 2 active by selecting it as StartUp project.
2. On the XMEGA-A1 Xplained connect a jumper or cable between TX and RX on J1.
3. Look through the code (`task2.c`) and try to understand what happens
4. Compile the code and assure that there are no warnings or errors
5. Start the debugging session

¹ The high priority interrupts are running so frequently that the lower priority interrupt routines never get time (CPU cycles) to run.

6. Step into `USART_InterruptDriver_Initialize()` (press F11) and access the driver file
7. Try to understand how the driver sets up the registers. Step out of the driver file, press Shift+F11
8. Locate the `receiveArray[]` buffer and add a watch, see Figure 4-2

Figure 4-2. Add a watch to the buffer pointers



Name	Value	Type
receiveArray	{uint8_t[3]{data}@0x2005}	uint8_t[3]{data}@0x2005
[0]	0	uint8_t{data}@0x2005
[1]	0	uint8_t{data}@0x2006
[2]	0	uint8_t{data}@0x2007

9. Now, run the code for a while by pressing F5
10. After a short while, break the execution (press Ctrl+F5)
11. Now, take a look at `receiveArray[]`. Have the characters been transferred correctly?
12. Reset the debug session (Shift+F5) and place break-points to the ISR routines. Run the program and see if it acts as expected
13. Open the `usart_driver.h` and set both `USART_RX_BUFFER_SIZE` and `USART_TX_BUFFER_SIZE` to 2
14. In `Task2.c`, set `NUM_BYTES` to 7 and recompile the project
15. In the Atmel AVR Studio menu, press Debug->Remove all Breakpoints
16. Run the program for a while (press F5) and break the execution (Ctrl+F5)



Why cannot the program run to completion? (Hint: What sizes are the software ring buffer and the hardware buffer and what is the size of the array to send.)

5 Task 3: USART using DMAC

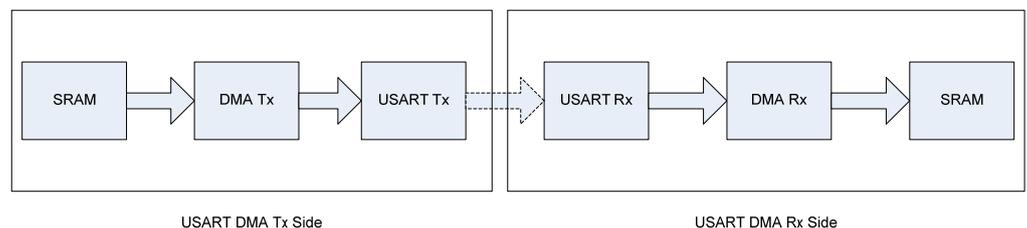
Atmel XMEGA introduces DMA controller for 8-bit microcontrollers. Using a DMA controller will offload the CPU when handling data transmission and help increase the performance of the microcontroller significantly.

This task will show how to set up the USART with the DMAC driver. Figure 5-1 illustrates the data flow in this task using DMA.

The goals for this task are that you know how to:

- Set up the USART using DMAC
- Set up the DMAC to read data from SRAM
- Set up the DMAC to write data to USART
- Set up the DMAC to read data from USART
- Set up the DMAC to write data to SRAM

Figure 5-1. Data flow in the DMA example



Todo

1. Locate the Atmel XMEGA-USART folder, open the `xmega_usart.avrsln` solution file and set Task 3 active by selecting it as StartUp project.

2. On the XMEGA-A1 Xplained connect a jumper or cable between TX and RX on J1.

3. Study how the Transmit channel is set up (`SetupTransmitChannel`)



- a. `Tx_Buf` is the input for the DMA transmit channel
- b. The DMA is set up to increase the address of `Tx_Buf`. Why?
- c. The USART data register is the output for the DMA transmit channel
- d. The DMA is set up to keep the address to the data register fixed during transmission. Why?
- e. Are you able to verify the correct trigger source (Data Register Empty) in the Atmel XMEGA manual?



4. Study how the Receive channel is set up (`SetupReceiveChannel`)

- a. The USART data register is the input for the DMA receive channel
- b. `Rx_Buf` is the output for the DMA receive channel
- c. The DMA is set up to have fixed receive data register address and to increase the address of `Rx_Buf`. Why?
- d. Are you able to verify that the correct trigger source (Receive complete) is used in the XMEGA manual?

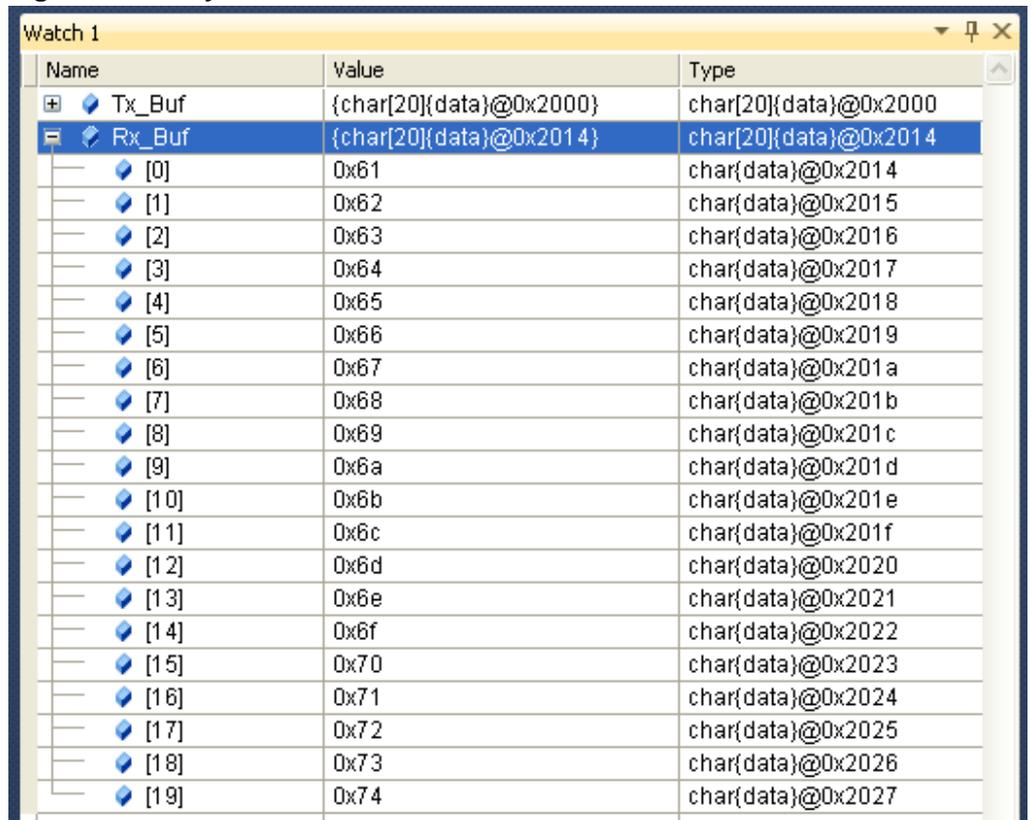
5. Build the project and start a debug session (click the Play icon)

6. Put a breakpoint on the first line after the DMA has completed (`"LEDPORT.OUT = ..."`). Run the code (press F5)



7. What is shown on the LEDs? Was the CPU able to increment the variable `i` while waiting for the DMA to complete?
8. Put a breakpoint on the line `LEDPORT.OUT = Rx_Buf[i];`
9. Run the code (press F5)
10. Check the status of the LEDs in comparison with the code; do the LEDs blink as expected?
11. Add watches to `Rx_Buf` and `Tx_Buf` and compare them, are they equal and contain characters from `a` to `t` (ascii `0x61` to `0x74`), see Error! Reference source not found..

Figure 5-2 Verify DMA transmission



Name	Value	Type
Tx_Buf	{char[20]{data}@0x2000}	char[20]{data}@0x2000
Rx_Buf	{char[20]{data}@0x2014}	char[20]{data}@0x2014
[0]	0x61	char{data}@0x2014
[1]	0x62	char{data}@0x2015
[2]	0x63	char{data}@0x2016
[3]	0x64	char{data}@0x2017
[4]	0x65	char{data}@0x2018
[5]	0x66	char{data}@0x2019
[6]	0x67	char{data}@0x201a
[7]	0x68	char{data}@0x201b
[8]	0x69	char{data}@0x201c
[9]	0x6a	char{data}@0x201d
[10]	0x6b	char{data}@0x201e
[11]	0x6c	char{data}@0x201f
[12]	0x6d	char{data}@0x2020
[13]	0x6e	char{data}@0x2021
[14]	0x6f	char{data}@0x2022
[15]	0x70	char{data}@0x2023
[16]	0x71	char{data}@0x2024
[17]	0x72	char{data}@0x2025
[18]	0x73	char{data}@0x2026
[19]	0x74	char{data}@0x2027

6 Summary

Here are some of the high-lights from this training:

- USART in Polling mode
- USART in interrupt mode
- USART driver

- USART software ring buffer
- USART hardware buffer
- USART DMA

7 Resources

- Atmel XMEGA Manual and Datasheets
 - <http://www.atmel.com/xmega>
- Atmel AVR Studio with help files
 - <http://www.atmel.com/avrstudio>
- Atmel IAR Embedded Workbench® compiler
 - <http://www.iar.com/>

8 Atmel Technical Support Center

Atmel has several support channels available:

- | | | |
|---------------|---|----------------------------|
| • Web portal: | http://support.atmel.no/ | All Atmel microcontrollers |
| • Email: | avr@atmel.com | All Atmel AVR products |
| • Email: | avr32@atmel.com | All AVR32 products |

Please register on the web portal to gain access to the following services:

- Access to a rich FAQ database
- Easy submission of technical support requests
- History of all your past support requests
- Register to receive Atmel microcontrollers' newsletters
- Get information about available trainings and training material



Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: (+1)(408) 441-0311
Fax: (+1)(408) 487-2600
www.atmel.com

Atmel Asia Limited
Unit 01-5 & 16, 19F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
HONG KONG
Tel: (+852) 2245-6100
Fax: (+852) 2722-1369

Atmel Munich GmbH
Business Campus
Parkring 4
D-85748 Garching b. Munich
GERMANY
Tel: (+49) 89-31970-0
Fax: (+49) 89-3194621

Atmel Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chou-ku, Tokyo 104-0033
JAPAN
Tel: (+81) 3523-3551
Fax: (+81) 3523-7581

© 2011 Atmel Corporation. All rights reserved.

Atmel®, Atmel logo and combinations thereof, XMEGA®, AVR Studio®, AVR®, AVR® logo and others are registered trademarks or trademarks of Atmel Corporation or its subsidiaries. Other terms and product names may be trademarks of others.

Disclaimer: The information in this document is provided in connection with Atmel products. No license, express or implied, by estoppel or otherwise, to any intellectual property right is granted by this document or in connection with the sale of Atmel products. **EXCEPT AS SET FORTH IN THE ATMEL TERMS AND CONDITIONS OF SALES LOCATED ON THE ATMEL WEBSITE, ATMEL ASSUMES NO LIABILITY WHATSOEVER AND DISCLAIMS ANY EXPRESS, IMPLIED OR STATUTORY WARRANTY RELATING TO ITS PRODUCTS INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. IN NO EVENT SHALL ATMEL BE LIABLE FOR ANY DIRECT, INDIRECT, CONSEQUENTIAL, PUNITIVE, SPECIAL OR INCIDENTAL DAMAGES (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS AND PROFITS, BUSINESS INTERRUPTION, OR LOSS OF INFORMATION) ARISING OUT OF THE USE OR INABILITY TO USE THIS DOCUMENT, EVEN IF ATMEL HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.** Atmel makes no representations or warranties with respect to the accuracy or completeness of the contents of this document and reserves the right to make changes to specifications and product descriptions at any time without notice. Atmel does not make any commitment to update the information contained herein. Unless specifically provided otherwise, Atmel products are not suitable for, and shall not be used in, automotive applications. Atmel products are not intended, authorized, or warranted for use as components in applications intended to support or sustain life.