AVR1510: Xplain training - XMEGA USART

Prerequisites

- **Required knowledge**
  AVR1500: Xplain training – XMEGA™ Basics
  AVR1502: Xplain training – XMEGA Direct Memory Access Controller
- **Software prerequisites**
  Atmel® AVR® Studio® 4.18 or later
  WinAVR/GCC 20100110 or later
- **Hardware prerequisites**
  Xplain evaluation board
  JTAGICE mkII
- **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 XMEGA USART and the three tasks will demonstrate how to use the USART I polling-mode, interrupt mode and how to use the DMAC (Direct Memory Access Controller) to transfer data without CPU interaction.

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 goal for this task is 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

1. Start Atmel AVR Studio and open the project file Polled_Usart.aps in the XMEGA-USART folder

2. On the Xplain evaluation board, connect a jumper or cable between pins PD2 and PD3

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

www.BDTIC.com/ATMEL

3.2 Debugging the Polled USART

6. Look through the code and see 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
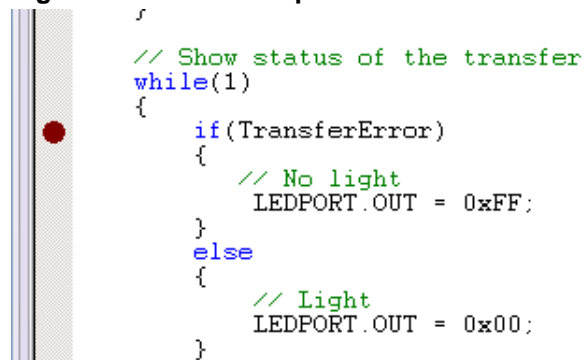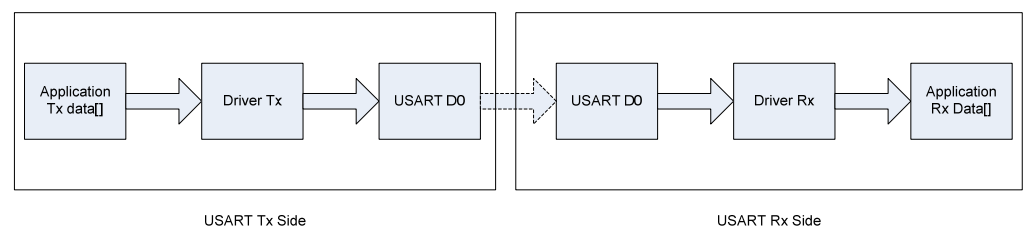
4 Task 2: USART in interrupt mode

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 are starving out[1] 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

| Application Tx data[] | Driver Tx | USART D0 | USART D0 | Driver Rx | Application Rx Data[] |

USART Tx Side   USART Rx Side

The goal for this task is 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
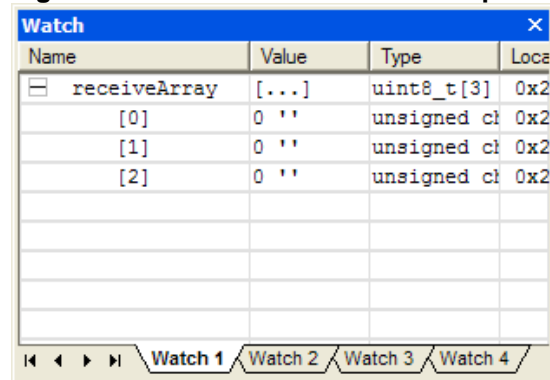• Understand how the hardware buffer works

1. Locate and open the project `InterruptControlled.aps`

2. Connect a jumper between Tx and Rx on PORTD, that is PD2 and PD3

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	Loca
⊟ receiveArray	[...]	uint8_t[3]	0x2
[0]	0 ''	unsigned ch	0x2
[1]	0 ''	unsigned ch	0x2
[2]	0 ''	unsigned ch	0x2

Watch 1 / Watch 2 / Watch 3 / Watch 4

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 goal for this task is 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
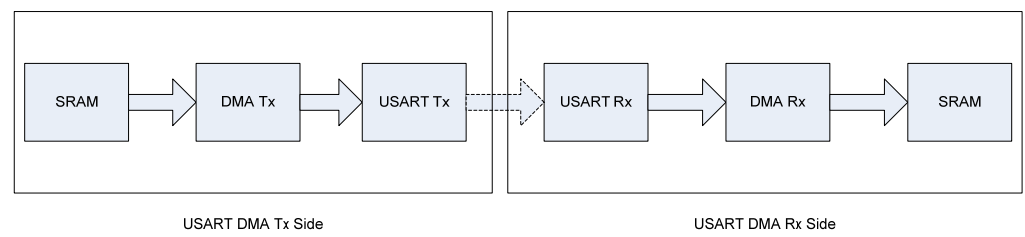
1. Open the project file `USART_DMA.aps` in Atmel AVR Studio and open task3.c

2. Connect a jumper between Tx and Rx on PORTD, that is PD2 and PD3

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? (0x6C)

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?
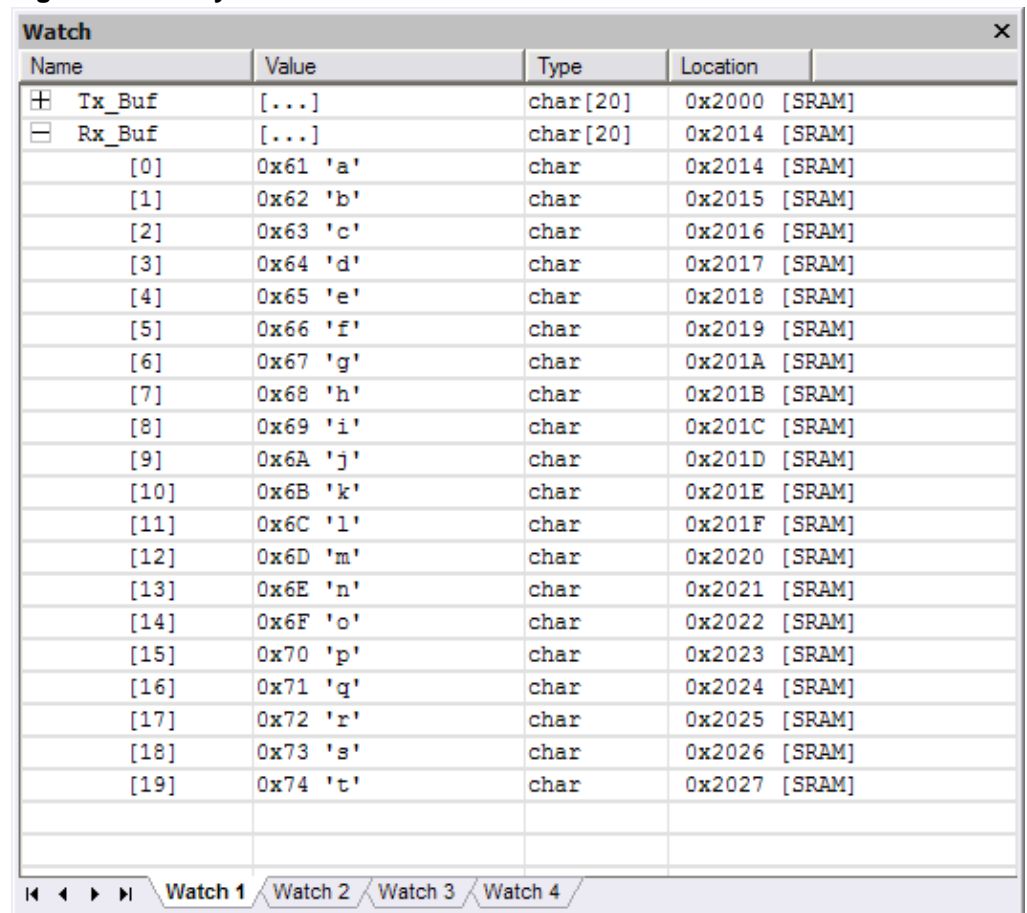
7

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, see Figure 5-2.

Figure 5-2. Verify DMA transmission

Watch				×
Name	Value	Type	Location	
⊞ Tx_Buf	[...]	char[20]	0x2000 [SRAM]	
⊟ Rx_Buf	[...]	char[20]	0x2014 [SRAM]	
[0]	0x61 'a'	char	0x2014 [SRAM]	
[1]	0x62 'b'	char	0x2015 [SRAM]	
[2]	0x63 'c'	char	0x2016 [SRAM]	
[3]	0x64 'd'	char	0x2017 [SRAM]	
[4]	0x65 'e'	char	0x2018 [SRAM]	
[5]	0x66 'f'	char	0x2019 [SRAM]	
[6]	0x67 'g'	char	0x201A [SRAM]	
[7]	0x68 'h'	char	0x201B [SRAM]	
[8]	0x69 'i'	char	0x201C [SRAM]	
[9]	0x6A 'j'	char	0x201D [SRAM]	
[10]	0x6B 'k'	char	0x201E [SRAM]	
[11]	0x6C 'l'	char	0x201F [SRAM]	
[12]	0x6D 'm'	char	0x2020 [SRAM]	
[13]	0x6E 'n'	char	0x2021 [SRAM]	
[14]	0x6F 'o'	char	0x2022 [SRAM]	
[15]	0x70 'p'	char	0x2023 [SRAM]	
[16]	0x71 'q'	char	0x2024 [SRAM]	
[17]	0x72 'r'	char	0x2025 [SRAM]	
[18]	0x73 's'	char	0x2026 [SRAM]	
[19]	0x74 't'	char	0x2027 [SRAM]	
⏮ ◀ ▶ ⏭ \ Watch 1 \ Watch 2 \ Watch 3 \ Watch 4 /				

AVR1510

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/products/AVR/
- WINAVR GCC compiler
  - http://winavr.sourceforge.net/
- 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

Headquarters

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

International

Atmel Asia
Unit 1-5 & 16, 19/F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
Hong Kong
Tel: (852) 2245-6100
Fax: (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-
Yvelines Cedex
France
Tel: (33) 1-30-60-70-00
Fax: (33) 1-30-60-71-11

Atmel 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

Product Contact

Web Site
www.atmel.com

Technical Support
avr@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Request
www.atmel.com/literature