# AVR274: Single-wire Software UART

8-bit AVR® Microcontrollers

Application Note

## Features

- **Software implemented UART**
- **Half-duplex single-wire communication**
- **Interrupt driven**
- **Supports baud rates up to 9600 @ 1MHz System Clock**
- **Compatible with any AVR® supporting external interrupt and a 8-bit timer compare interrupt**

## 1 Introduction

UART communications are usually implemented using separate data wires for reception and transmission. A single-wire UART is using only one wire for communication, and is therefore ideal in low cost solutions where no high-speed full duplex communication is needed. This application note describes a software implementation of a single wire UART. The protocol supports half duplex communication between two devices. The only requirement is an I/O port supporting external interrupt and a timer compare interrupt.

Rev. 8065A-AVR-03/07

# 2 Theory of Operation

## 2.1 UART frame

The UART protocol is an asynchronous serial communication standard. Data is transferred sequentially, one bit at a time. This implementation uses a frame consisting of 8 data bits, one start bit and two stop bits as shown in Figure 2-1. Other implementations may use different frame formats consisting of 5 to 9 data bits, 1 parity bit for error control and 1 stop bit. The line is high when no transmission is ongoing.

**Figure 2-1.** UART frame format

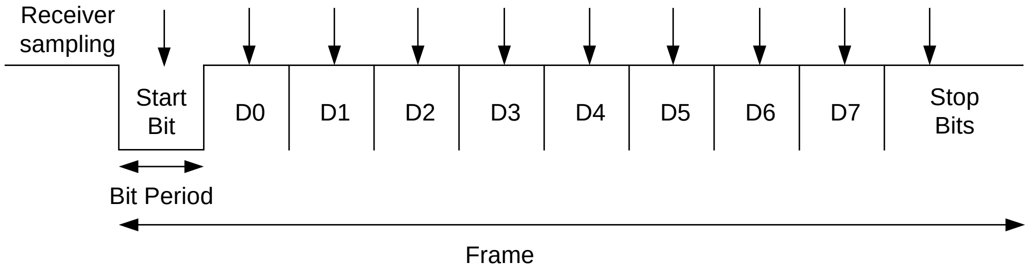
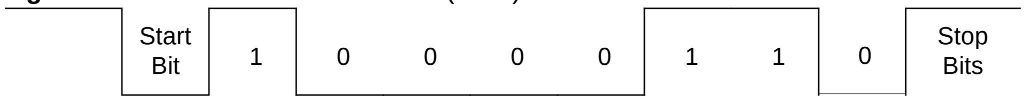


**Figure 2-2.** Serial frame for ASCII ‘a’ (0x61)



## 2.2 Transmission

Transmission is initialized by sending the start bit (pulling the line low) for one bit period. The receiver detects the falling edge and is then able to synchronize to the transmitter. The least significant bit of the data bits is sent first.

Open collector outputs are used to drive the line, but if both devices are transmitting at the same time, the transmitter sending a low bit will pull the line low even if the other transmitter is sending a high bit. To handle this situation the UART never starts transmitting while receiving. The transmitter is also sampling the line before transmitting a new bit to make sure the line has not changed since the last bit was transmitted. An error flag is set if a low bit is received when a high bit was last transmitted.

## 2.3 Reception

Reception is started when the start bit is detected. The data bits are then sampled in the middle of every period. The first data bit is then sampled one and a half bit period after the start bit is detected. This implementation is not using the three samples majority vote found in most hardware UARTs to minimize clock cycles used by the driver.

## 2.4 Baud Rate

In UART communication speed is defined by the baud rate. The baud rate is in this case equal to the number of bits transmitted per second including the start and stop bits. The receiver and transmitter have to be set up using the same baud rate, or else they will not be able to synchronize. Common baud rates are 4800, 9600, 19200, 28800 and 38400, but other rates may also be used.

## 2.5 Error Conditions

There are several error conditions that can occur. If the baud rate differs too much on the two devices, they can get unsynchronized. Since the baud rate is dependent on a clock frequency, this problem can occur if the clock differs from the intended value. If using an internal RC oscillator, it is recommended to calibrate it before using the UART. Please refer to relevant application notes on how to calibrate the internal RC oscillator.

The received data byte is stored in a single byte buffer. If the data is not handled before the next byte is received, the buffer is overflowed and old data overwritten. An overflow flag is set if this occur. To help this situation UART speed can be reduced, buffer increased, or the receive routine may be called more frequently if possible.

If a noise pulse forces the line low, the AVR will detect the falling edge and start receiving. If the start bit is received as a high bit the AVR will stop reception and not save any data, but if the noise pulse lasts more than a couple of cycles a corrupted byte will be saved.

Noise may also corrupt a frame under transmission. A bit is sampled only once so the frame will have an incorrect value if only one sample is corrupted by noise. To detect these errors a parity bit can be added to the UART frame.

Since an interrupt driven approach is used, application code may execute in parallel to the UART communication. Note that if other interrupt sources are active this may affect the UART timing and may cause the UART communication to fail.

It is recommended to use the driver in a master/slave configuration where the slave only sends data when requested by the master. This will prevent situations where both devices are transmitting at the same time. If the slave is in an error condition it may signal the master by pulling the UART line low for a specified time. The master's error flag will then be set, and the communication may be continued when the slave stops pulling the line low.

# 3 Implementation

The code described in this application note is written as drivers for the UART communication.

## 3.1 Baud Rate Settings

A timer compare interrupt is used to generate the bit sampling and bit transmission intervals. The timer is set in Clear Timer on Compare (CTC) mode and to generate an interrupt when the timer equals the output compare register. The time between each interrupt is dependent on the system clock, timer prescaler and value in the compare register as shown in Equation 3-1. Setting the compare value to 10 will generate 11 ticks between each interrupt. Baud rate settings are set in the UART header file (single_wire_UART.h).

**Equation 3-1.** Baud Rate Calculation

$$Baud\ Rate = \frac{System\ Clock}{(One\ Period\ Compare\ Setting + 1) \cdot Timer\ Prescaler}$$

**Equation 3-2.** One Period Compare Setting Calculation

$$One\ Period\ Compare\ Setting = \frac{System\ Clock}{Baud\ Rate \cdot Timer\ Prescaler} - 1$$

Table 3-1 shows timer settings for some common clock speeds and baud rates. The error values are calculated using Equation 3-3.

**Table 3-1.** One Period Settings for 1, 2, 4 and 8 MHz Oscillator

| Baud Rate (bps) | 1 MHz Oscillator | | | 2 MHz Oscillator | | |
|---|---|---|---|---|---|---|
| | One Period Setting | Prescaler Setting | Error | One Period Setting | Prescaler Setting | Error |
| 4800 | 207 | 1 | -0.16% | 51 | 8 | -0.16% |
| 9600 | 103 | 1 | -0.16% | 207 | 1 | -0.16% |
| 19200 | NA | | | 103 | 1 | -0.16% |
| | 4 MHz Oscillator | | | 8 MHz Oscillator | | |
| 4800 | 103 | 8 | -0.16% | 207 | 8 | -0.16% |
| 9600 | 51 | 8 | -0.16% | 103 | 8 | -0.16% |
| 19200 | 207 | 1 | -0.16% | 51 | 8 | -0.16% |
| 28800 | 138 | 1 | 0.08% | 34 | 8 | 0.82% |
| 38400 | 103 | 1 | -0.16% | 207 | 1 | -0.16% |

**Equation 3-3.** Error calculation

$$Error\ [\%] = (\frac{Baud\ Rate_{Closest\ Match}}{Baud\ Rate} - 1) \cdot 100\%$$

Please note that there is a maximum setting for the baud rate. This is because the timer interrupt has to be finished before a new interrupt is generated. Equation 3-4 gives the maximum baud rate possible. The interrupt must finish before a new compare interrupt is triggered. The maximum cycles in a compare interrupt are about 100-110 depending on the compiler settings. Using a 1MHz system clock the maximum baud rate is about 10,000 bit/s. The UART will consume about all CPU resources when communicating at this baud rate. The application using the UART must also have time to read the received data before a new frame is received, or else an overflow error will occur. It is recommended to set the baud rate well below the maximum setting depending on the application.

**Equation 3-4.** Maximum Baud Rate Setting

$$Baud\ Rate < \frac{System\ Clock}{Maximum\ Cycles\ in\ Compare\ Interrupt}$$

## 3.2 Hardware

This implementation is designed using an external pull-up circuit. The I/O pins must therefore use open-collector outputs. In addition they need to generate an external interrupt to detect an incoming frame.

A typical value for a pull-up resistor on an AVR microcontroller is 15-40kΩ. When sending a high bit or receiving the AVR port is tri-stated. A low bit is sent by configuring the port to output low.

If communicating with a device supporting the RS-232 standard, voltage levels from about -15 to 15V are needed. A circuit is then needed to convert the signals to these voltages. An example of a single chip interface circuit is Maxims MAX232. It operates from a single 5V power supply, and has an onboard DC/DC converter to generate the RS-232 levels.

## 3.3 Status register

The single-wire UART has a status register containing the following four flags:

`SW_UART_TX_BUFFER_FULL`

Set if TX data is ready to be transmitted. This flag must be zero when calling the SW_UART_transmit function.

`SW_UART_RX_BUFFER_FULL`

Set if data is available in the receive buffer. This flag must be one when calling the SW_UART_Receive function.

`SW_UART_RX_BUFFER_OVERFLOW`

Set if incoming data is lost due to overflow in the receive buffer.

`SW_UART_FRAME_ERROR`

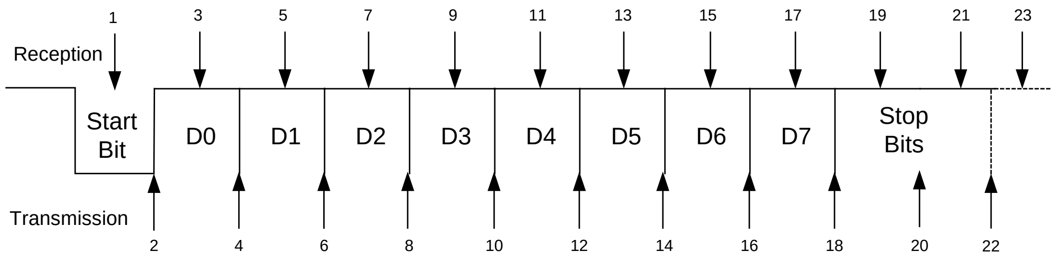
Set when receiving if a high start bit or a low stop bit is sampled. Also set when transmitting if a different bit then the last transmitted is sampled.

To decrease code size and increase speed the status register may be put in a GPIO register if available. (Not available on the ATmega32).

## 3.4 UART counter

A counter variable is used in the UART driver to control the state and which bit to be transmitted/received by the UART. The UART is idle when the counter value is 0. The counter value is even when transmitting and odd when receiving as shown in Figure 3-1.

**Figure 3-1.** UART Counter Values



## 3.5 UART functions

The driver consists of three global functions:

```
void      SW_UART_Enable(void)
void      SW_UART_Transmit(uint8_t)
uint8_t   SW_UART_Receive(void)
```

The SW_UART_status is a global variable holding the UART status flags. The SET_FLAG, CLEAR_FLAG and READ_FLAG macros can be used to access the status flags defined in single_wire_UART.h.

The following interrupts routines are used in the UART implementation:

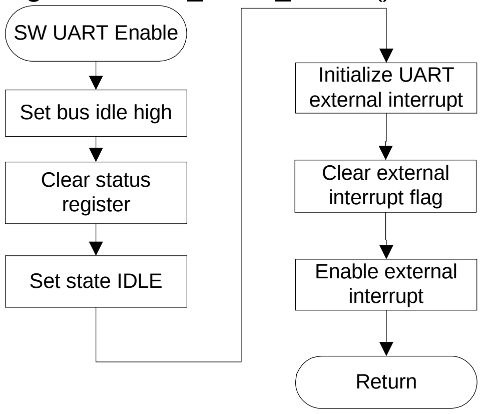
```
__interrupt void External_SW_UART_ISR()
__interrupt void Timer_Compare_SW_UART_ISR()
```

The next section contains brief descriptions and flowcharts for the UART functions:

### 3.5.1 SW_UART_Enable

Before data can be received or transmitted the UART needs to be enabled by calling the SW_UART_Enable function. It tri-states the UART pin so the line is idle high. The status register and counter is cleared so all ongoing transmissions are stopped. Disabling the external and timer interrupt will stop the UART.

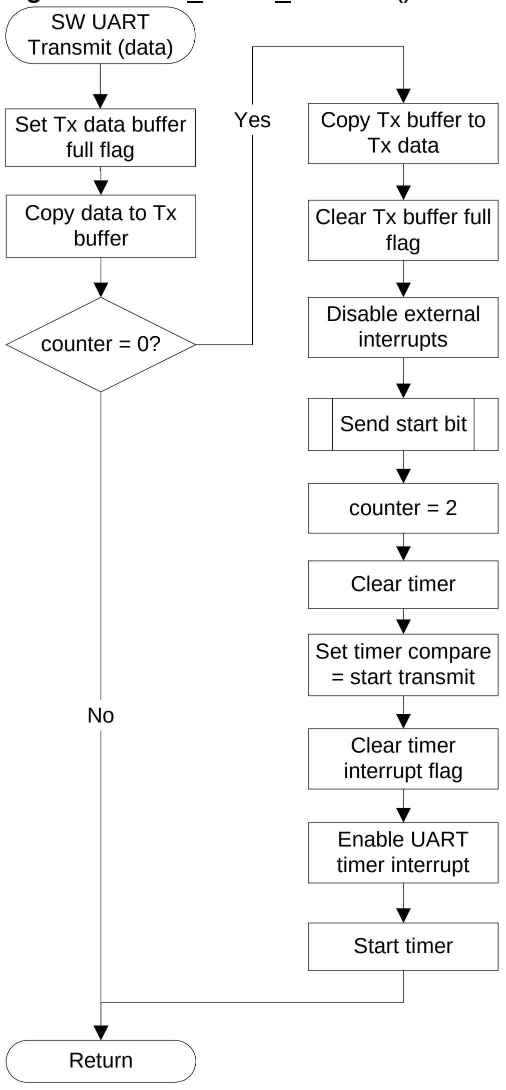
**Figure 3-2.** SW_UART_Enable() function


SW UART Enable
Set bus idle high
Clear status register
Set state IDLE
Initialize UART external interrupt
Clear external interrupt flag
Enable external interrupt
Return

### 3.5.2 SW_UART_Transmit

The SW_UART_Transmit() function takes one byte as parameter and adds this byte to the transmit buffer. The SW_UART_TX_BUFFER_FULL flag must be zero when calling this function, or else data will be lost. If a data transfer is not in progress when this function is called, a new transmission will be started by sending a start bit and enabling the timer interrupt.

**Figure 3-3.** SW_UART_Transmit() function


SW UART Transmit (data)
Set Tx data buffer full flag
Copy data to Tx buffer
counter = 0?
Yes
No
Copy Tx buffer to Tx data
Clear Tx buffer full flag
Disable external interrupts
Send start bit
counter = 2
Clear timer
Set timer compare = start transmit
Clear timer interrupt flag
Enable UART timer interrupt
Start timer
Return

### 3.5.3 SW_UART_Receive

This function returns the byte received in Rx_data. The UART_RX_BUFFER_FULL flag must be checked before calling this function to make sure there is valid data in Rx_data. When receiving multiple bytes it is important to call this function before next byte is received, or else the receive buffer will be overflowed and data lost.

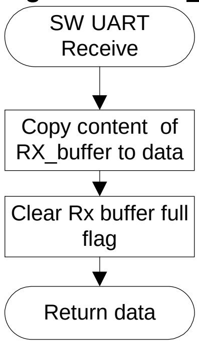
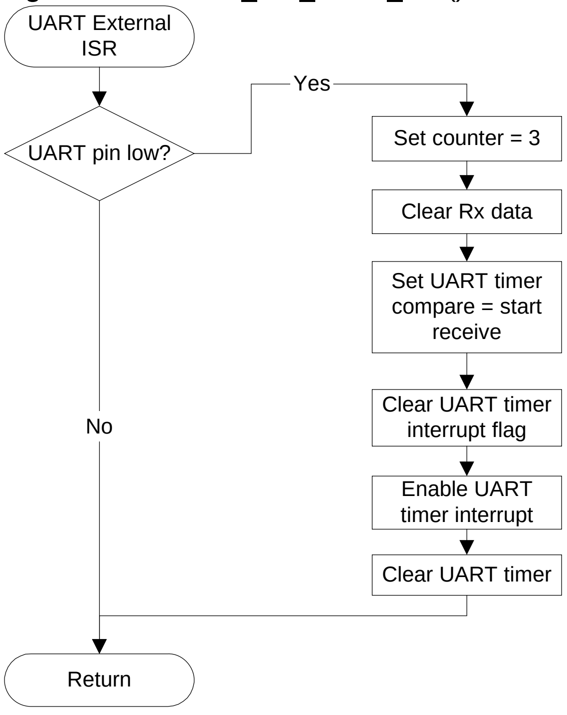


**Figure 3-4.** SW_UART_Receive() function

### 3.5.4 SW_External_UART_ISR

The External_SW_UART_ISR() is triggered when a falling edge is detected on the line when there is no ongoing transmission or reception. The routine checks if the UART pin is low. If not, no reception is started. It also disables further external interrupts so no falling edges on the data bits will trigger a new interrupt, as it is only used to detect the start bit.

**Figure 3-5.** External_SW_UART_ISR()


UART External ISR
UART pin low?
Yes
No
Set counter = 3
Clear Rx data
Set UART timer compare = start receive
Clear UART timer interrupt flag
Enable UART timer interrupt
Clear UART timer
Return

### 3.5.5 Timer_Compare_SW_UART_ISR

The Timer_Compare_SW_UART_ISR() (Figure 3-1) controls the handling for transmitting and receiving frames. It is called automatically when the output compare register equals the timer and the UART timer interrupt is set. Data is transmitted when the UART counter is even, and received when the counter is odd. The receive - (Figure 3-8) and transmit handler (Figure 3-7) is implemented directly into the timer interrupt. Please refer to Figure 3-1 for details on the different counter values.

**Figure 3-6.** Timer_Compare_SW_UART_ISR()

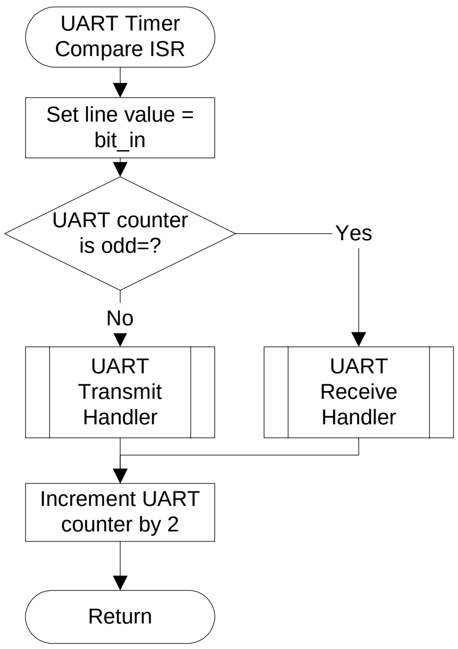


**Figure 3-7.** UART Transmit Handler()

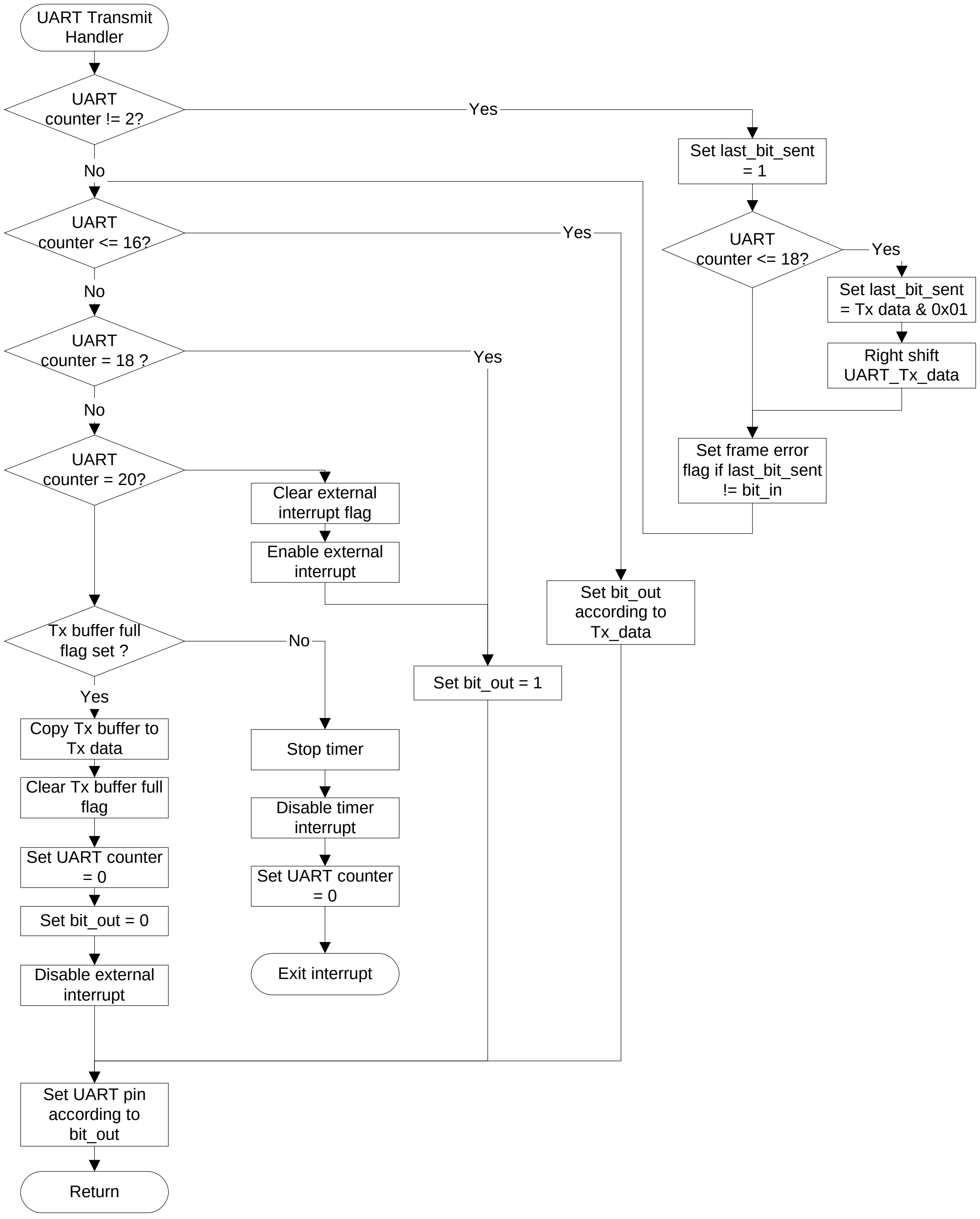
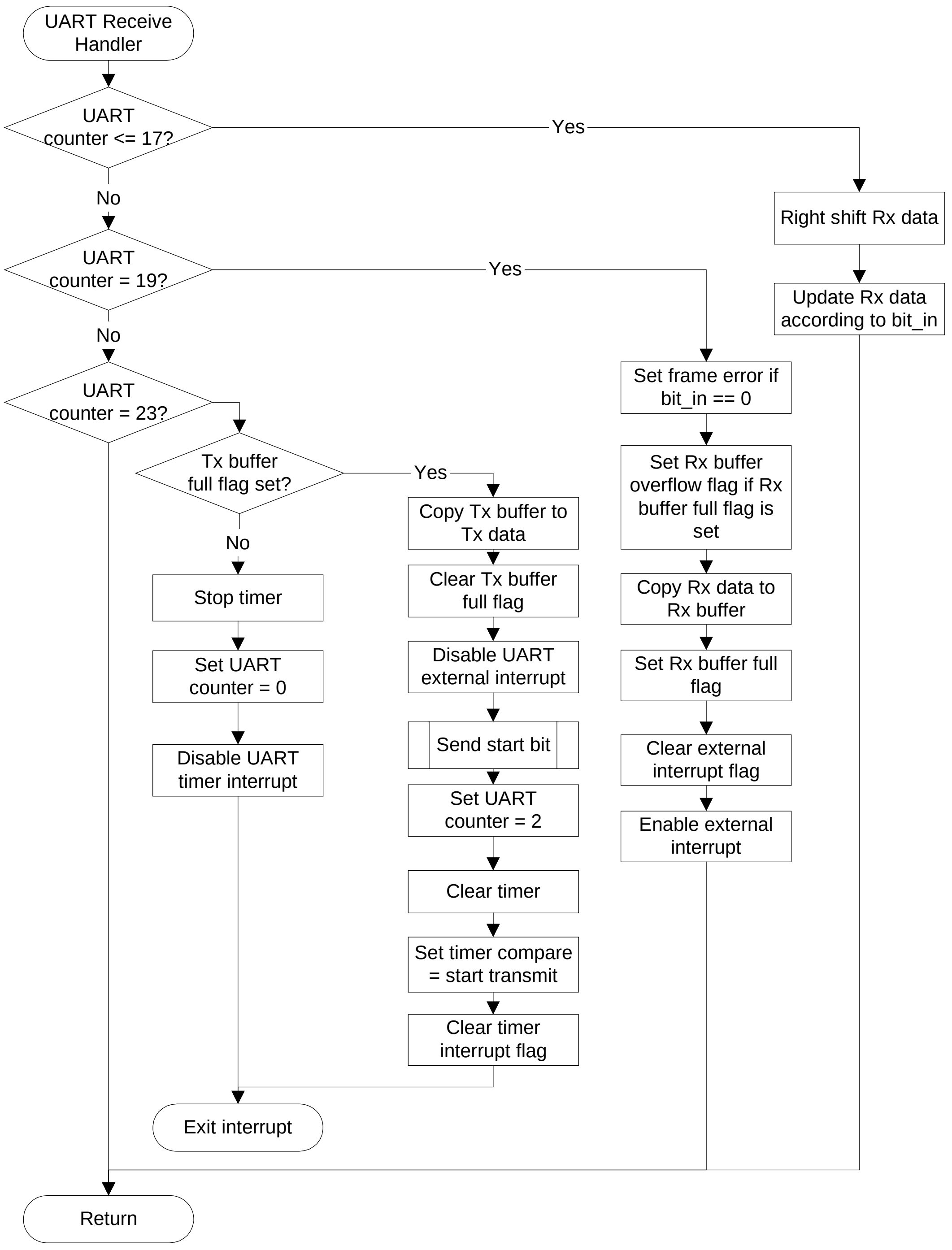


**Figure 3-8.** UART Receive Handler()



UART Receive Handler
UART counter <= 17?
Yes
No
Right shift Rx data
Update Rx data according to bit_in
UART counter = 19?
Yes
No
Set frame error if bit_in == 0
Set Rx buffer overflow flag if Rx buffer full flag is set
Copy Rx data to Rx buffer
Set Rx buffer full flag
Clear external interrupt flag
Enable external interrupt
UART counter = 23?
Tx buffer full flag set?
Yes
No
Copy Tx buffer to Tx data
Clear Tx buffer full flag
Disable UART external interrupt
Send start bit
Set UART counter = 2
Clear timer
Set timer compare = start transmit
Clear timer interrupt flag
Stop timer
Set UART counter = 0
Disable UART timer interrupt
Exit interrupt
Return

## 3.6 Example Program

The main.c contains an example program for testing the UART. It receives data to a byte array and transmits the data back when the array is full or when a return character is received.

## 3.7 Code Size

When complied with IAR® EWAVR 4.21A and maximum speed optimization turned on the code size for the UART driver is 500 bytes.

## 4 Getting Started

The source code can be downloaded as a zip-file from www.atmel.com/avr. The code is written for the ATmega32 and complied using the IAR EWAVR 4.20A complier. To compile the source without any changes the IAR EWAVR complier is needed. Doxygen documentation is available in readme.html.

## Headquarters

*Atmel Corporation*

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

## International

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

## 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*
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex
France
Tel: (33) 4-76-58-47-50
Fax: (33) 4-76-58-47-60

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