AVR307: Half Duplex UART Using the USI Module

Features

- **Half Duplex UART Communication**
- **Communication Speed Up To 230.4 kbps at 14.75MHz**
- **Interrupt Controlled Communication**
- **Eight Bit Data, One Stop-bit, No Parity**
- **Data Buffer Handling**

Introduction

The Universal Serial Interface (USI) present in AVR devices like the ATtiny26, ATtiny2313, and ATmega169, is a communication module designed for TWI and SPI communication. The USI is however not restricted to these two serial communication standards. It can be used for UART communication as well.

By using the USI for UART communication one can avoid some of the inconveniences associated with software implemented UART communication. By using the USI, interrupts control the communication and the bits are automatically shifted out. In this way, less processing time is occupied by the communication, freeing processing time for other parts of the application while communicating.

This document describes how to use the USI of the ATtiny26 for UART communication. Source code for communication drivers for transmission and reception is provided. The drivers can be adapted to the ATtiny2313 or ATmega169 USI modules by minor changes to the code. The code is complete with both data buffer handling and combined transmitter and receiver. These features can be split or removed to optimize the use of resources.

8-bit AVR® Microcontrollers

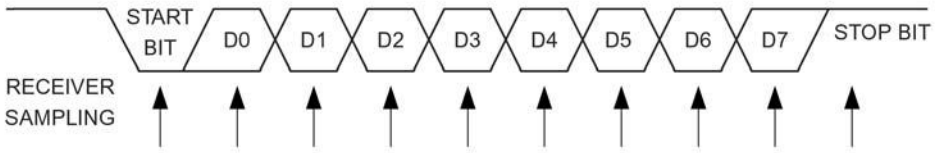
Application Note

Rev. 4300A–AVR–10/03

Theory of Operation

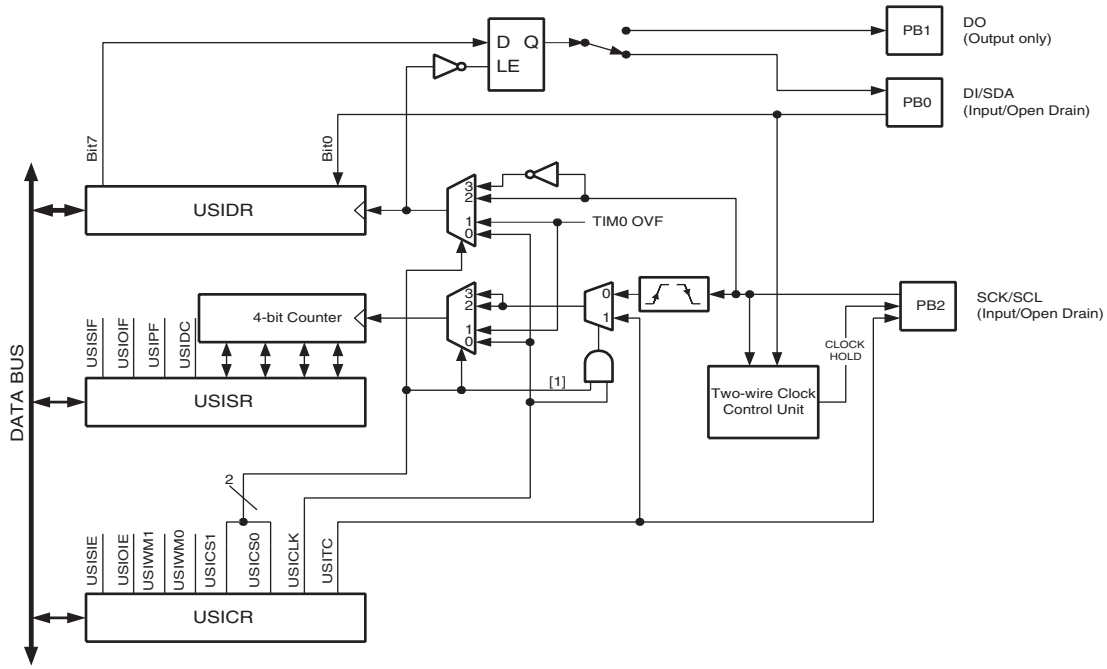
The UART protocol is an asynchronous serial communication standard. The frame format is specified as a start bit followed by 5 to 9 data bits, one parity bit (optional) and one or two stop bits. The data is sent with LSB first. The frame format for UART communication is illustrated in Figure 1.

Figure 1. Frame Format for UART Communication



The USI is basically an eight-bit shift register connected to input and output pins. When used for UART communication, a general purpose timer is control the in/out clocking of data bits, and a dedicated USI counter generates interrupts when a given number of bits have been shifted in/out. The USI block diagram can be seen in Figure 2.

Figure 2. Universal Communication Interface of the ATtiny26



Since a UART frame format is specified to be from 7 to 13 bits in total, with a typical frame size of 10 bits, the USI shift register is a couple of bits short of a typical frame. However, using the features of the USI module one can circumvent this limitation.

UART Reception

The suggested solution for reception is to detect a start-bit and then sample the eight data bits in their appropriate time slots (while ignoring the stop bit). The first sampling of the data is done after a half-bit duration, which is the time from the falling edge of the start-bit to the middle of the start-bit. The successive samplings are done with fixed intervals of one bit duration determined by the baud rate. Since the shift register can

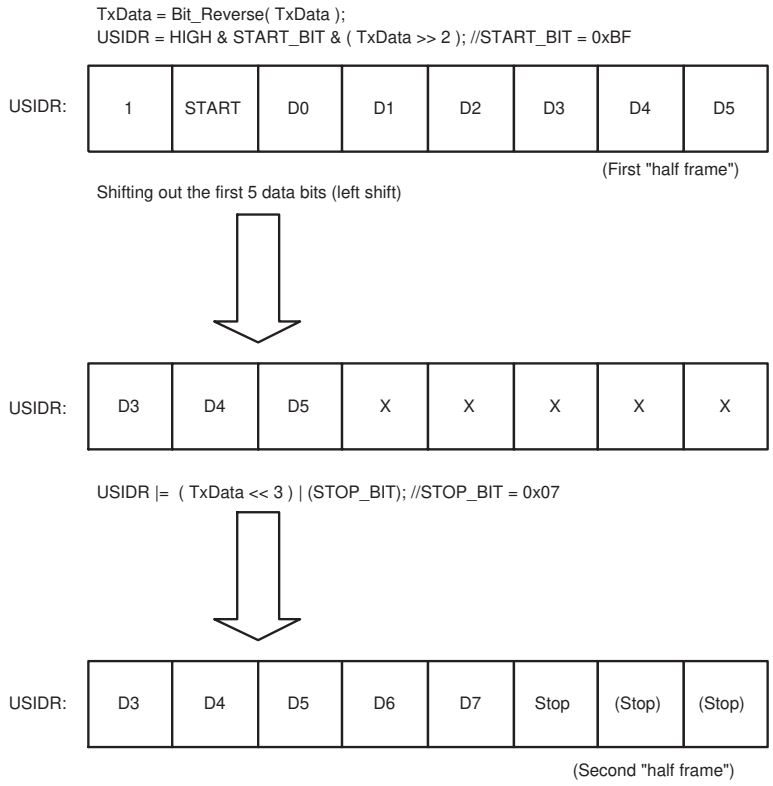
only hold 8 bits the start-bit is not kept in the shift register, but “falls” out when the last data bit is shifted in. This method is preferred next to performing the first sampling on the first data bit, since it simplifies the use of Timer/Counter0, controlling the delays and thereby the bit rates used. More details about bit rate is provided in the section “Bit Rates” on page 4” .

One should be aware that one difference between the AVR hardware UART and UART communication implemented in software or by the use of the USI module is how the input data is sampled. In the hardware UART a “majority vote” of three samples is used to determine the value of the sampled bit. The “majority vote” eliminates the likelihood of noise causing data corruption. In most full or partial software implementations only one sampling of each data bit is done. This is also the case for the USI implementation of UART communication. The possibility that noise can cause corruption of the incoming data is therefore increased. This fact could be seen as an incitement to use parity check for reception. Parity check is not added in the receive driver described in this document. An alternative solution is to use CRC checking of the complete message as described in the application note AVR350 available on the Atmel web site.

UART Transmission

UART transmission is slightly more complicated; the frame has to be divided in two. The MSB of the USI shift register is directly connected to the USI Data Output pin. To ensure correct timing the MSB must therefore hold a high state until the timer starts shifting the data correctly. The USI Shift register is loaded with a high value, the start-bit, and the first 6 LSB of the data. The transmission is then initiated. When the first 5 bits in the shift register have been shifted out the shift register is reloaded, and in this way the remainder of the frame is added to the contents of the shift register. The procedure for loading/reloading the USI data register (USIDR) for transmission is illustrated in Figure 3.

Figure 3. Loading the USIDR Buffer for Transmission



USI Modes

Different communication modes, or 'Wire Modes', can be selected for the USI, but since the USI module is designed for TWI and SPI communication, there are no dedicated wire modes for UART communication. The wire modes that can be selected are "Three-wire" (SPI) or "Two-wire" (TWI). The mode that should be used for UART communication is the SPI mode, since this is the mode that has separate data in and out lines.

In both wire modes the USI module can generate a so-called Start Condition Interrupt. The Start Condition interrupt is in SPI mode triggered by a level change on the SCK pin, but since it is desired to wake up on a change on the DI (Data Input) pin, it is better to use a pin change interrupt. A pin change interrupt can wake up the AVR from all sleep modes. The pin change interrupt can thus be used to detect the start-bit of a UART frame. The start bit in the UART reception will in this way generate a pin change interrupt and the interrupt routines can thereafter handle the reception. The SCK pin is therefore not required and can be used for other purposes.

Bit Rates

The eight-bit Timer/Counter0 is used to specify the bit rate for the serial communication. In ATtiny26 the overflow of the timer can be used to trigger the USI shift register. In ATmega169 a compare match is used to trigger the shift register. The method for generating the bit rate of the serial communication is thus to run the timer and make the timer overflow/compare match events trigger the bit shifts.

In this document the USI of the ATtiny26 and therefore the Timer/Counter0 overflow is used to trigger the bit shifting. The timer seed matching the duration of one bit can be determined from Equation 1.

Figure 4. Equation 1

$$Cycles\ per\ bit = \frac{System\ Clock}{Baudrate \cdot Timer\ prescaler}$$

$$Timer0\ seed = 256 - (Cycles\ per\ bit)$$

If the timer seed approaches the top value of the timer one must be aware that the rounding/ truncation of the timer seed can introduce a significant timing error. The rounding error relative to the number of cycles per bit is recommended as low as possible. As a rule of thumb the total timing error should be less than 2% to ensure error-less communication – the total error is the sum of both the accuracy of the system clock and the UART bit timing.

Further, if the timer seed is close to the top value of the timer, which will occur when using high-speed communication, there will be rather few cycles between each timer interrupt. This can potentially cause problems in serving the interrupts fast/often enough. It is recommended to carefully consider the number of cycles required to serve the timer interrupt, especially if other interrupts can occur and thereby block the timer overflow interrupt. In addition one should also consider the interrupt latency as an important factor and correct the timer seed according to this. The interrupt latency should be added to the timer seed value together with the number of cycles it takes to replant the timer seed. This is described in the implementation section.

A list of timer seeds is found in Table 1. The timer seeds are calculated using Equation 1 and are not corrected with regard to interrupt latency.

Table 1. Timer0 Seed for Different System Clocks and Communication Speeds

System Clock (MHz)	Communication Speeds Possible						
	9600	14.4k	19.2k	28.8k	57.6k	115.2k	230.4k
1	152	187(3)	Na	Na	Na	Na	Na
1.84	64	128	160	192(3)	Na	Na	Na
2	48	117	152	187(3)	Na	Na	Na
3.69	208(2)	(1)	64	128	192(3)	Na	Na
4	204(2)	221(2)	48	117	186(3)	Na	Na
7.37	160(2)	192(2)	207(2)	(1)	128	192(3)	Na
8	152(2)	187(2)	204(2)	221(2)	117	187(3)	Na
11.06	112(2)	160(2)	184(2)	208(2)	64	160	Na
14.75	64(2)	128(2)	160(2)	192(2)	(1)	128	192(3)
16	48(2)	117(2)	152(2)	187(2)	221(2)	117	187(3)

Notes: 1. Does not require timer0 timer seed.
2. Timer prescaler is CK/8.
3. Indicates critical timing.

At some communication speeds the timer seed is “0”. In these cases it is not required to replant the timer seed, since the seed is “replanting” itself by wrapping once the overflow occurs. The Timer/Counter overflow interrupt can thus be disabled in these cases.

Note that the initial timer seed for reception is different from the timer seeds specified in Table 1; the initial timer seed used for the reception should generate a delay of a ½ or 1½ bit duration, depending on whether your first sample is going to be the start bit or the first data bit. They can be calculated as shown by Equation 2 and Equation 3.

Figure 5. Equation 2

$$Timer0\ seed = 256 - \left(Cycles\ per\ bit \cdot 1/2\right)$$

Figure 6. Equation 3

$$Timer0\ seed = 256 - \left(Cycles\ per\ bit \cdot 3/2\right)$$

Each strategy has its limitations that you must be aware of. If you have a very low Cycles per bit value, dividing it by 2 may give you an initial timer seed that is shorter then the time it takes to set up the USI for reception. On the other hand, if the Cycles per bit value is very high, then multiplying it by 1.5 may give an initial timer seed value that is higher then 256.

A recommended approach could be to first check if ((Cycles per bit * 3/2) > 256) then use (Cycles per bit * 1/2).

For the ATmega169 the considerations related to the reloading of the timer seed are minimized since the compare match is used; the compare match is equivalent to auto-loading of the timer seed, similar to the situation mentioned above where the timer seed is “0”.

USI Counter

The Timer/Counter0 determines the bit rate, whereas the USI counter is handling the transmission of the data on byte/frame level. The USI counter is a four-bit counter used to count the number of bits received/transmitted from USIDR. The counter top value is 16. The clock for the USI counter is depending on the USI Clock Source Select bits (USICS1:0) and the USI Clock Strobe bit (USICLK). When the USI is used for UART communication these bit should be USICS1:0 = 01 and USICLK = X (don’t care). With these settings the USI counter and the USIDR shift register are clocked by the Timer/Counter0 overflow. The USI counter thus counts the number of bits transmitted/received.

Reception

The USI timer/counter can generate an overflow interrupt, which can be used to indicate that a number of bits have been received. To accomplish this the USI counter is pre-loaded with the top value minus the number of data bits (typically eight bit). See Figure 7 and Figure 8.

Figure 7. Equation 4

$$USI\ Counter\ seed = Counter\ top\ value - (Start\ bit + Data\ bits)$$

Figure 8. Equation 5

$$USI\ Counter\ seed = Counter\ top\ value - (Data\ bits)$$

Counter top value is 16, *Start bit* is 1 and *Data bits* is typically 8 (size of one byte). The choice is dependent on whether your initial timer seed will make you sample the start bit or not. See initial timer seed discussion in the “Bit Rates” on page 4 for more information.

Transmission

In a transmission the USI counter overflow is used to determine that the USIDR needs reloading (to be able to send an entire 10-bit frame). The USI counter is in this case pre-loaded with the Counter top value minus the *Half-frame size*, which is the number of bits that should be transmitted before reloading the counter (see Figure 3). The initial USI Counter seed for transmission is determined as shown by Figure 9.

Figure 9. Equation 6

$$USI\ Counter\ seed = Counter\ top\ value - (Half\ \ frame\ size)$$

After the second “half-frame” is loaded into USIDR, the USI counter is preloaded again to generate an interrupt once the entire frame has been transmitted. The secondary USI Counter seed for transmission is determined by Equation 7.

Figure 10. Equation 7

$$USI\ Counter\ seed = Counter\ top\ value - ((Full\ \ frame\ size) - (Half\ \ frame\ size))$$

Summary

The Timer/Counter0 is used to control the bit rate and the four-bit USI counter overflow is used for the frame handling. Since both reception and transmission require Timer/Counter0 and the USI counter, it is only possible to use the USI for half duplex communication.

The pin change interrupt on the USI Data Input pin is in the reception used for detecting the frame start.

In transmission, the USI Data Register must be loaded twice for each transmitted frame.

Implementation

The code described this application note is written as drivers for UART communication. The code includes the control elements allowing interlaced receptions and transmissions giving true half duplex communication. To simplify the use of the drivers, communication buffers have been implemented. This enables your application to execute in parallel with the UART communication. If there is no need for both transmission and reception, or communication buffers, then this code could be removed to minimize the drivers in size and optimize the performance.

The code has been written and compiled using the IAR EWAVR 2.28A compiler with all optimizations enabled. Testing and debugging has been done using AVR Studio 4.07 and ICE50.

Communication Buffers

The receive and transmit routines use 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. Note that these variables must be a power of 2. If not, a compiler error message will be flagged.

By inspecting the buffer head and tail pointers it can be determined if new data is available; that is, if the write pointer is different from the read pointer.

It is recommended to increase the receive buffer size if high-speed communication with critical timing is required. Otherwise, data may be lost since the receive driver does not verify that data have been read from the receive buffer.

An extra function is added to the example code. The DataInReceiveBuffer returns zero if the receive buffer does not contain any data. This function does not, in contrast to the ReceiveByte function, wait for incoming data, but returns immediately the status of the buffer.

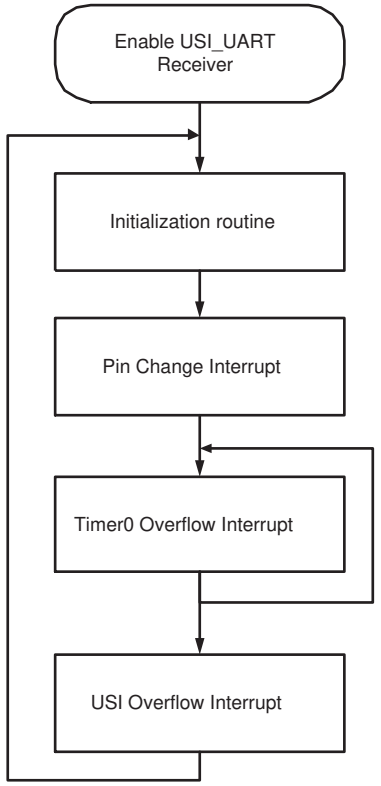
Note: This routine does not return the number of bytes in the buffer.

USI UART Receiver

The driver consists of four parts:

- USI UART initialization
- Pin Change interrupt service routine
- Timer/Counter0 overflow interrupt service routine
- USI Counter overflow interrupt service routine

Figure 11. Execution Sequence of the Receive Driver



Each of these parts are described below by flow-charts and comments to these flow charts.

USI UART Initialization - Receiver

This function prepares the device for a reception.

The USI has a dedicated USI Start Condition interrupt, but this is associated with the SCK line. For UART communication a separate clock line is not used, a reception is detected as activity on the Data Input (DI) line. The initial activity is always a high-to-low transition on the bus, due to the start-bit of the UART frame. A Pin Change interrupt is used to detect the initiated transmission. Therefore, the USI_UART initialization routine sets up the Pin Change interrupt to wait for the incoming data.

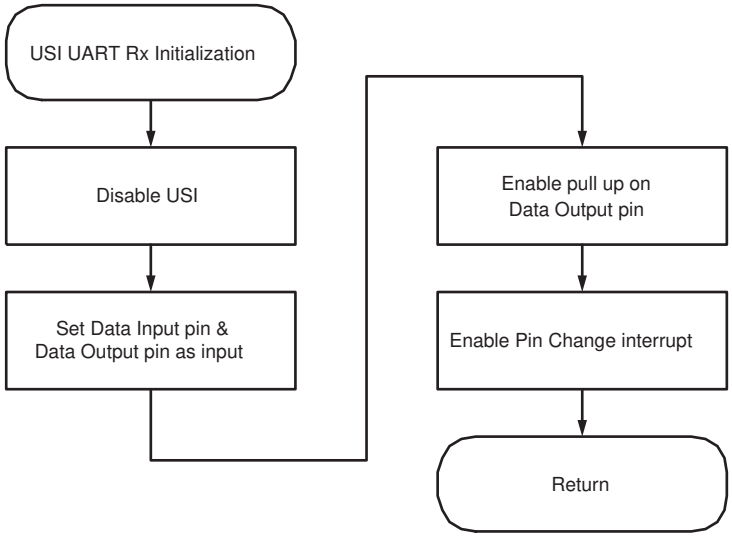
The USI functionality will override the Pin Change feature; the USI module is therefore disabled. Then both the DI and DO pins are defined as inputs. DI because data is expected on this pin, and DO to ensure that no data is shifted out unintentionally.

The idle level for a UART communication pin is high, as it is an open collector bus, so the pull up on the DO pin is enabled.

Finally the Pin Change interrupt is enabled.

Note that global interrupts are not enabled in the initialization function. This should therefore be done in the main routine. The flow chart is shown in Figure 12.

Figure 12. Initialization of the Receive Driver



Pin Change Interrupt – Receiver

The purpose of the Pin Change interrupt is to start Timer/Counter0 once the falling edge of the start bit is detected. Once the start bit is detected the Pin Change interrupt should be disabled, since only the falling edge of the start bit should generate a Pin Change interrupt. This is done automatically by enabling the USI, since it will overrun the pin change feature on the same pins. The pin change interrupt is enabled again in the USI Counter Overflow interrupt service routine once a complete frame has been received. A flow chart of the Pin Change interrupt service routine is presented in Pin Change Interrupt Service Routine.

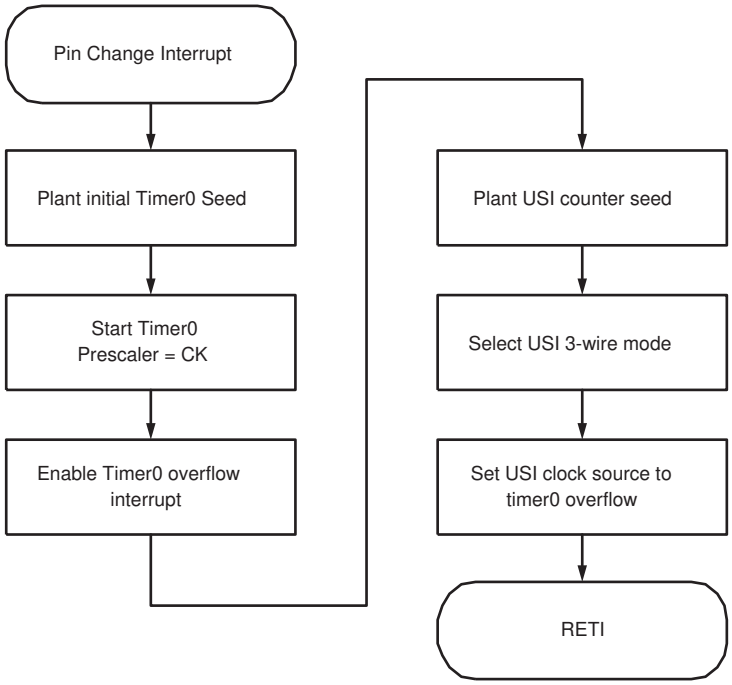
To use the USI for reception the wire-mode is set to “three-wire” (SPI). The USI clock source is selected to be the Timer0 overflow.

The USI counter is initialized so that it will generate overflow when the required number of bits have been received.

Timer/Counter0 is initialized by loading it with the timer seed required to generate the initial delay; the delay from the falling edge of the start-bit to the middle of the sampled bit. See Figure 1 regarding the frame format. For exact timing one could also include the delay from when the DI pin changes value to when the timer0 starts running. This value depends on how fast you are able to preload and start the timer0 in the interrupt service routine. Place this operation first in the ISR and simulate execution of the final code to calculate the delay.

Finally all the required interrupts are enabled. If the Timer/Counter0 timer seed is 0, the Timer/Counter0 overflow does not need to be enabled. See comments on this in the section “Timer/Counter0 Overflow Interrupt - Transmitter” on page 15.

Figure 13. Pin Change Interrupt Service Routine

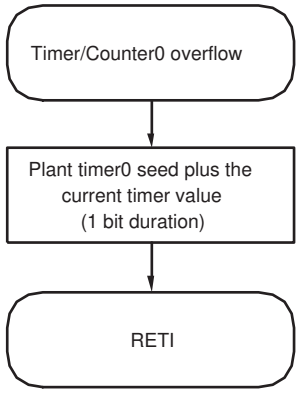


Timer/Counter0 Overflow Interrupt - Receiver

The purpose of the Timer/Counter0 Overflow interrupt service routine is to replant the timer seed that controls the sampling of the incoming data bits. The interrupt routine is described by the flow chart in Figure 14. The Timer/Counter0 overflow interrupt service routine is similar for both the receive driver and the transmit driver.

If the timer seed is 0 (see Table 1) the overflow interrupt service routine is not required since there is no timer seed to replant. If the timer overflow interrupt can be avoided the timing of the UART communication will be less influenced by the presence of other interrupts sources. Further, less processing resources are used on the UART reception (and transmission).

Figure 14. Timer/Counter0 Overflow Interrupt Service Routine



USI Counter Overflow Interrupt - Receiver

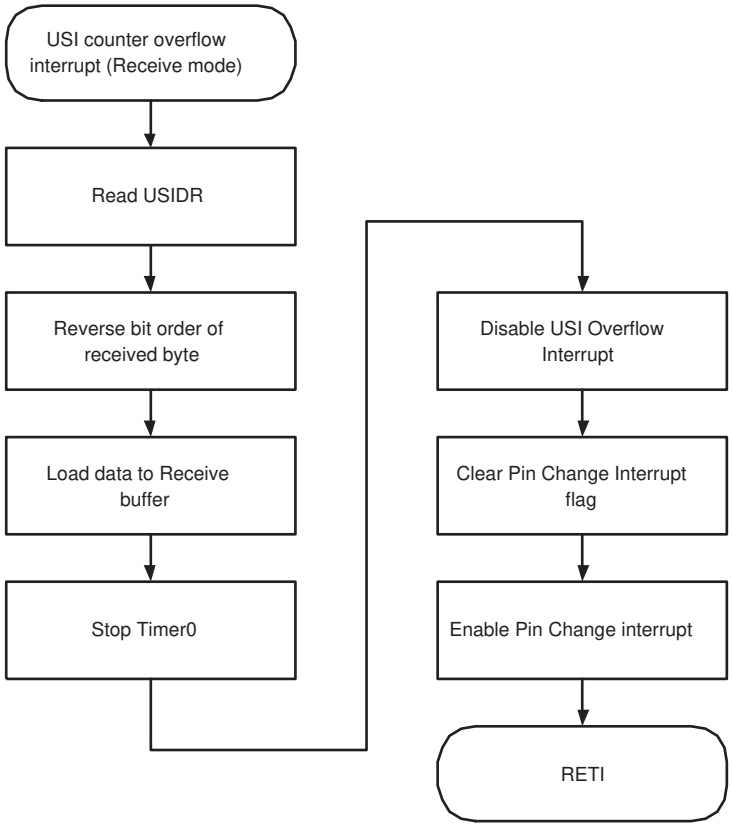
In the receive driver, the USI Counter overflow is triggered when an entire byte has been received. It thereby indicates the completion of a reception and handles the moving of the received data to the receive buffer and the re-initialization of the USI to make it ready for receiving a new frame. The interrupt routine is visualized in Figure 15.

First, the Timer/Counter0 is stopped. This ensures that the no further in-shifting of data is made.

The received data is located in USIDR and needs to be moved before new data can be received. The data is moved to a circular data buffer. The data needs to be bit reversed before used. This could either be done here or when it is read out from the receive buffer.

The re-initialization of the USI as receiver involves disabling the USI and enabling the pin change interrupt.

Figure 15. USI Counter Overflow Interrupt Service Routine



Used Resources

The receiver uses Timer0, USI module/pins, and the pin change interrupt.

The memory requirements for the receive driver are displayed in Memory Requirements of the Receive Driver. The majority of the code memory, which is 136 bytes of the total 236, is required by the interrupts. Reversing the bit order of the data is among the routines that demand much memory. Note that many of the sub sections of the code are equal with and therefore shared with the transmitter, making the total code size smaller then the sum.

Table 2. Memory Requirements of the Receive Driver

Receiver		Code [bytes]
Initialisation and data/buffer handling		
	USI_UART_Initialise_Receiver()	30
	USI_UART_Receive_Byte()	28
	Bit_Reverse()	42
	Sum	100
Sampling & storing data		
	IO_Pin_Change_ISR()	68
	USI_Counter_Overflow_ISR()	68
	Sum	136

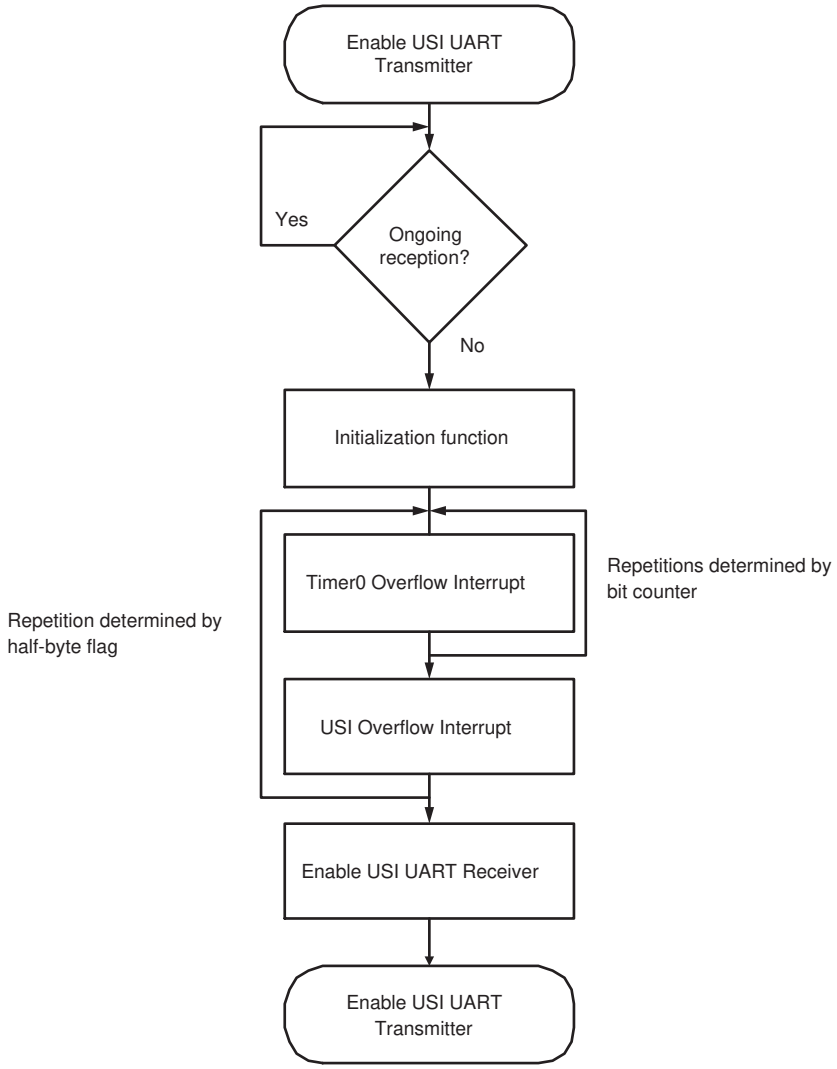
USI UART Transmitter

The transmit driver is implemented as a function call, where the byte to transmit is passed as an argument in the function call.

The transmit driver consists of four parts:

- Transmit function
- USI UART initialization
- Timer/Counter0 overflow interrupt service routine
- USI Counter Overflow interrupt service routine

Figure 16. Execution Sequence of the Transmission Driver



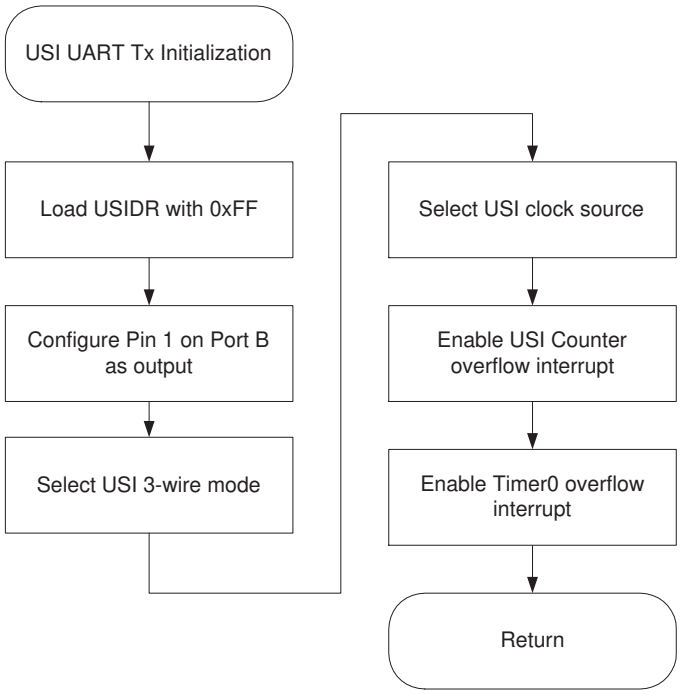
Transmit Function

To initiate a transmission the transmit function is called. The function is the part of the driver that the application interfaces. The transmit function is declared as:

void USI_UART_TransmitByte(**unsigned char**);

The data is prepared for transmission by reversing the bit order of the byte that is about to be transmitted (TxByte). Then it is stored in the transmit buffer. If a transmission is already in operation then no other actions are needed since the interrupt routines will pick up all data from the buffer automatically. If there are no transmissions ongoing then a test will be done of whether there is an ongoing reception before initializing the USI for transmission. The USI is configured for operation, but initial seeds are set to minimize the time until the interrupt routines are entered. It is in the interrupt routines that the data registers and seeds are loaded with correct content.

Figure 17. USI UART Transmit and Initialization Functions



USI UART Initialization – Transmitter

When three-wire mode is selected, the direction of pin 1 on port B is set to output, which in Three-wire mode is an open-drain output. An external pull-up is therefore required.

The interrupts required for the transmission are enabled.

Timer/Counter0 Overflow Interrupt - Transmitter

Timer/counter0 overflow interrupt service routine is similar to the one used by the receive driver. Refer to the section ““Timer/Counter0 Overflow Interrupt - Receiver” on page 11” for details on this routine.

USI Counter Overflow Interrupt – Transmitter

Note that the USI counter overflow interrupt is used for both transmit and receive. The first action taken is therefore to determine the current mode.

The data needs to be reversed before transmission. This is already prepared in the TransmitByte function. This is to transmit the data with LSB first according to the UART frame format (see Figure 1). The TxByte is then merged with a high value and a start bit. The merged data is copied to the USIDR (see Figure 3). A status bit is set to indicate that the USIDR has been loaded with the first “half-frame”, the start bit and the 6 LSB of the TxByte. This status bit is used in the USI Counter interrupt service routine.

The timer seeds for the Timer/Counter0 and the USI Counter are planted and the Timer/Counter0 is started. The Timer/Counter overflow will generate the clock for shifting out the bits in USIDR.

The next overflow interrupts from the USI Counter are generated when the first 5 bits of a frame (a “half frame”) have been transmitted and when a frame transmission has been completed. The USI Counter Overflow interrupt service routine is illustrated in Figure 15. Note that the figure also shows the receiver since it too uses the same USI Overflow interrupt.

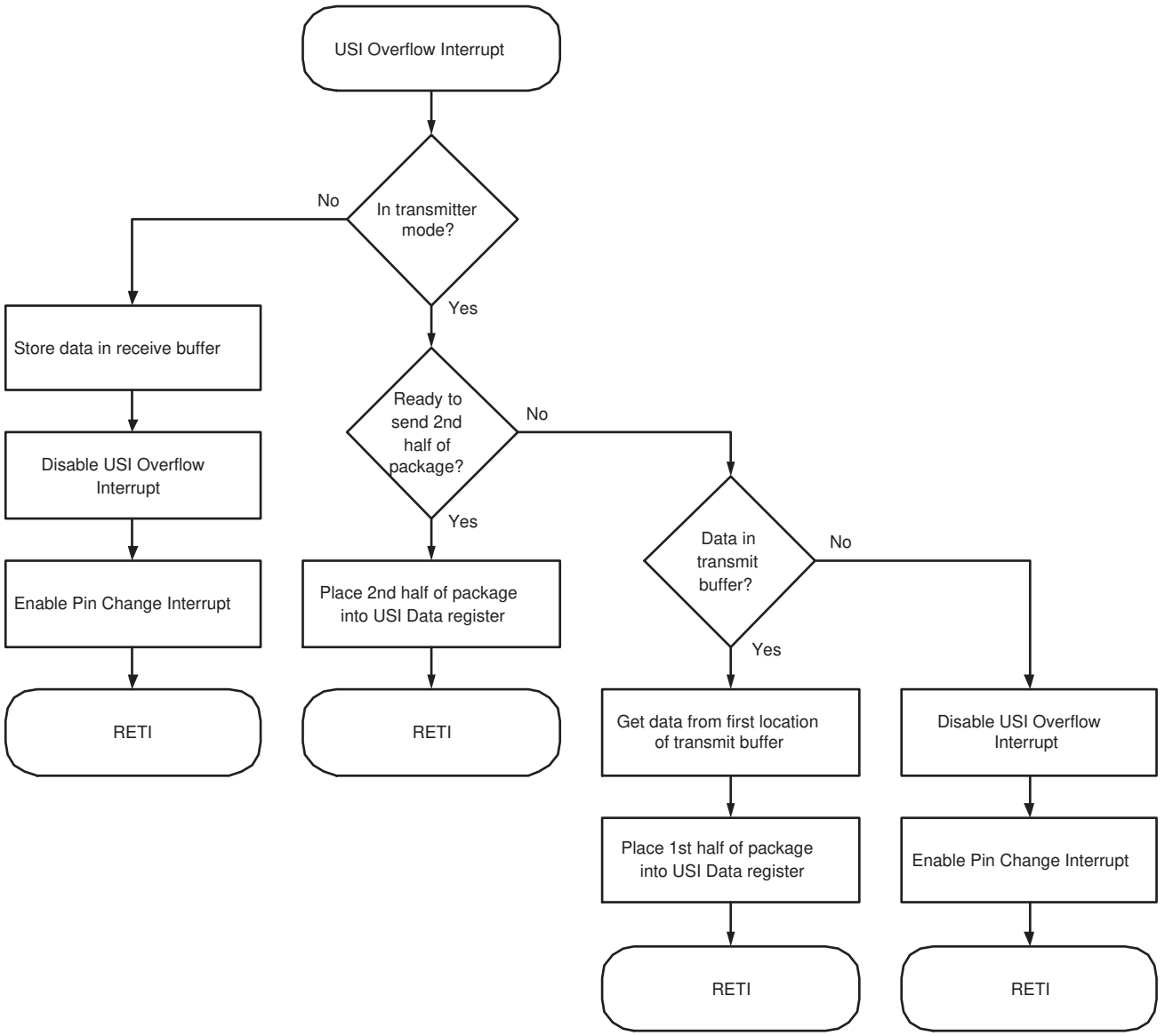
If the interrupt is due to the transmission of the first half frame, the rest of the data and a stop-bit are merged into USIDR (see Figure 3). The status bit that indicates that USIDR is loaded with the first half frame is then cleared.

If the interrupt is triggered by the transmission of a complete frame, the Timer/Counter0 is stopped and the TxOngoing flag is cleared.

Finally the USI Counter Overflow flag is cleared (all flags are cleared), since this is not done in hardware by triggering the interrupt vector.

The overflow interrupt routine will work through the complete transmit buffer before it disables the USI and reenters the USI UART Receive mode.

Figure 18. USI Counter Overflow Interrupt Service Routine



Used Resources

The receiver uses Timer0 and USI module/pins.

The memory requirements for the receive driver are displayed in Table 3. The majority of the code memory, which is 116 bytes of the total 252, is required by the USI counter overflow interrupt. Reversing the bit order of the data is among the routines that demand much memory. Note that many of the sub sections of the codes are equal with and therefore shared with the receiver, making the total code size smaller then the sum. The total code size is ~450 bytes.

Table 3. Memory Requirements of the Transmit Driver

Transmitter		**Code [bytes]**
Initialisation and data/buffer handling		
	USI_UART_Initialise_Transmitter()	46
	USI_UART_Transmit_Byte()	48
	Bit_Reverse()	42
	Sum	136
Sampling & storing data		
	USI_Counter_Overflow_ISR()	116

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

4300A–AVR–10/03