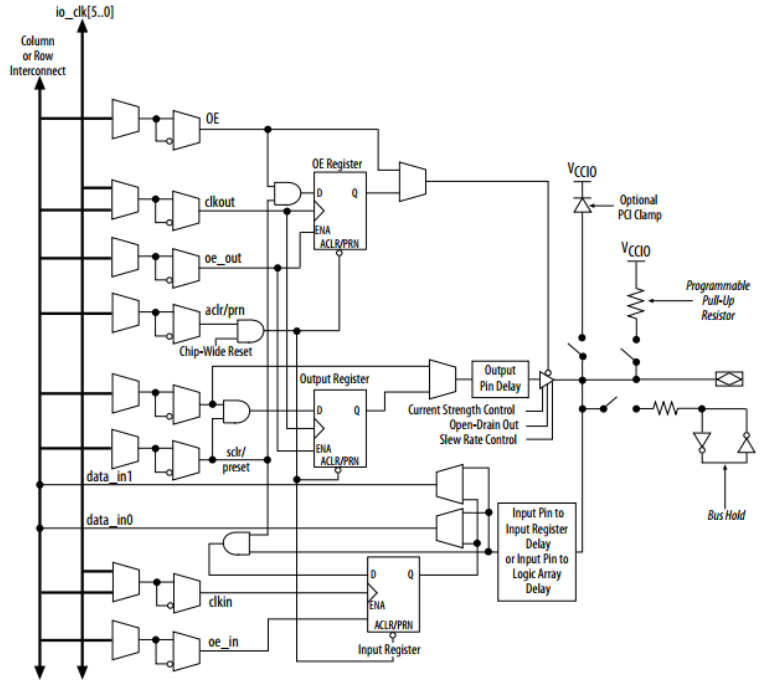


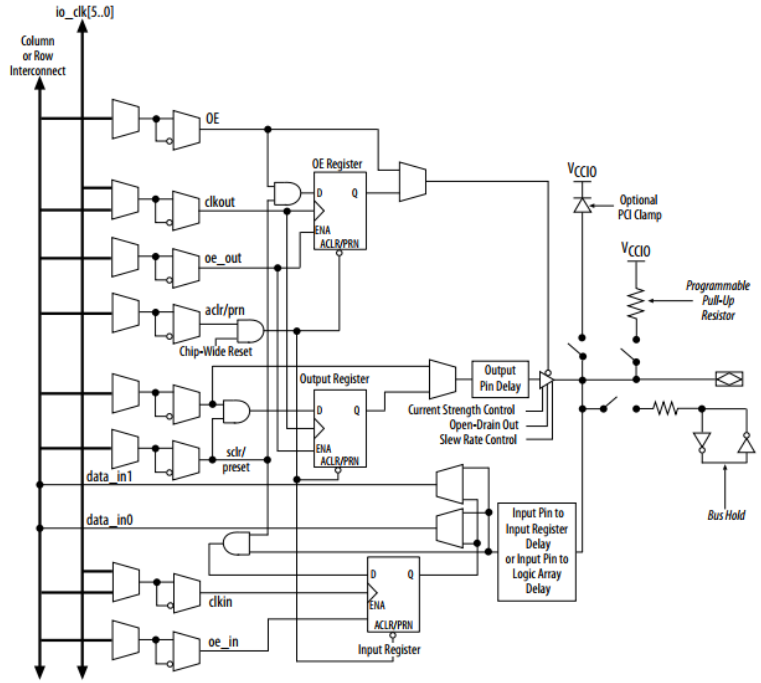
Hardwarenahe Programmierung

Fernstudiengang Elektrotechnik – SS2024

Prof. Dr. C. Jakob
University of Applied Sciences Darmstadt

fbeit
FACHBEREICH ELEKTROTECHNIK
UND INFORMATIONSTECHNIK

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES



Serielle Schnittstelle

Fernstudiengang Elektrotechnik – SS2024

Prof. Dr. C. Jakob
University of Applied Sciences Darmstadt

fbeit
FACHBEREICH ELEKTROTECHNIK
UND INFORMATIONSTECHNIK

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

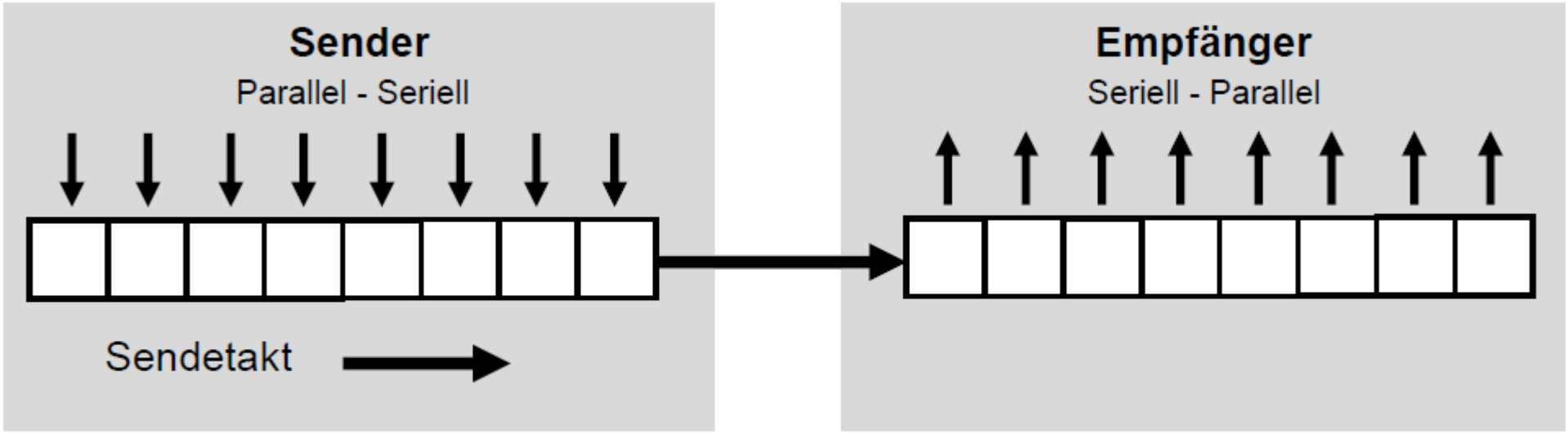


Abbildung 4-1 Serielle Datenkommunikation

© Prof. Dr. C. Jakob

Baudrate	Bit-Zeit in µs	Byte-zeit (1/8/1/N) in ms
1200	833	8.33
2400	416	4.16
4800	208	2.08
9600	104	1.04
19200	52	0.52
38400	26	0.26
115200	8.6	0.086

***Tabelle 4-1** Resultierende Bit- bzw. Bytezeiten für das (1/8/1) Format*

- Die Baudrate gibt die Anzahl der Zustandswechsel des gesendeten Signals im Zeitraum einer Sekunde an und wird mitunter auch als Bit-Takt bezeichnet. Die Angabe der Baudrate erfolgt in der Einheit Baud.
- Die Bitrate wiederrum gibt die Anzahl der gesendeten Bits pro Sekunde an und wird meist bps (*engl. Bits per second*) angegeben.

© Prof. Dr. C. Jakob

- **Gerade Parität** (*E, für engl.: even*): Das Paritätsbit wird so gesetzt, dass die Summe aus gesetzten Datenbits und dem Paritätsbit eine gerade Zahl darstellt.

Beispiel 1: 8-bit Datenwort: `0b00101001` Paritätsbit: 1
Beispiel 2: 8-bit Datenwort: `0b10101001` Paritätsbit: 0

- **Ungerade Parität** (*engl.: odd*): Das Paritätsbit wird so gesetzt, dass die Summe aus der gesetzten Datenbits und dem Paritätsbit eine ungerade Zahl darstellt.

Beispiel 1: 8-bit Datenwort: `0b11101001` Paritätsbit: 0
Beispiel 2: 8-bit Datenwort: `0b10101010` Paritätsbit: 1

- **Keine Parität** (*engl.: no parity*): Die Parität wird nicht berücksichtigt.

© Prof. Dr. C. Jakob

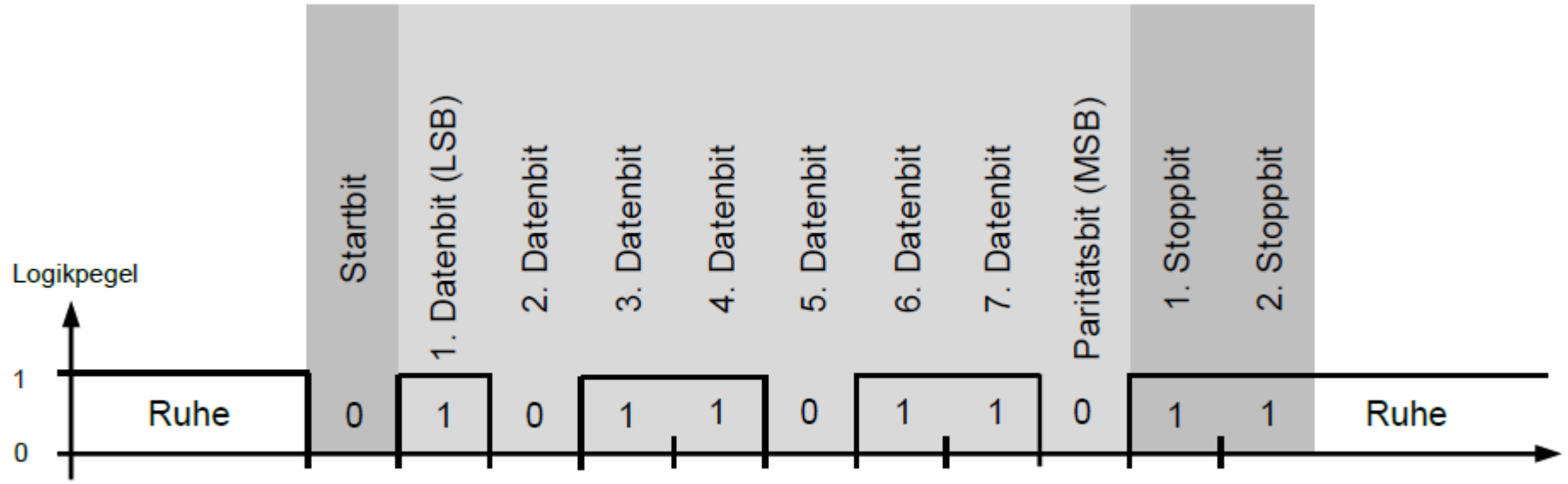


***Abbildung 4-2** USART Übertragung (1/7/1/2)*

© Prof. Dr. C. Jakob

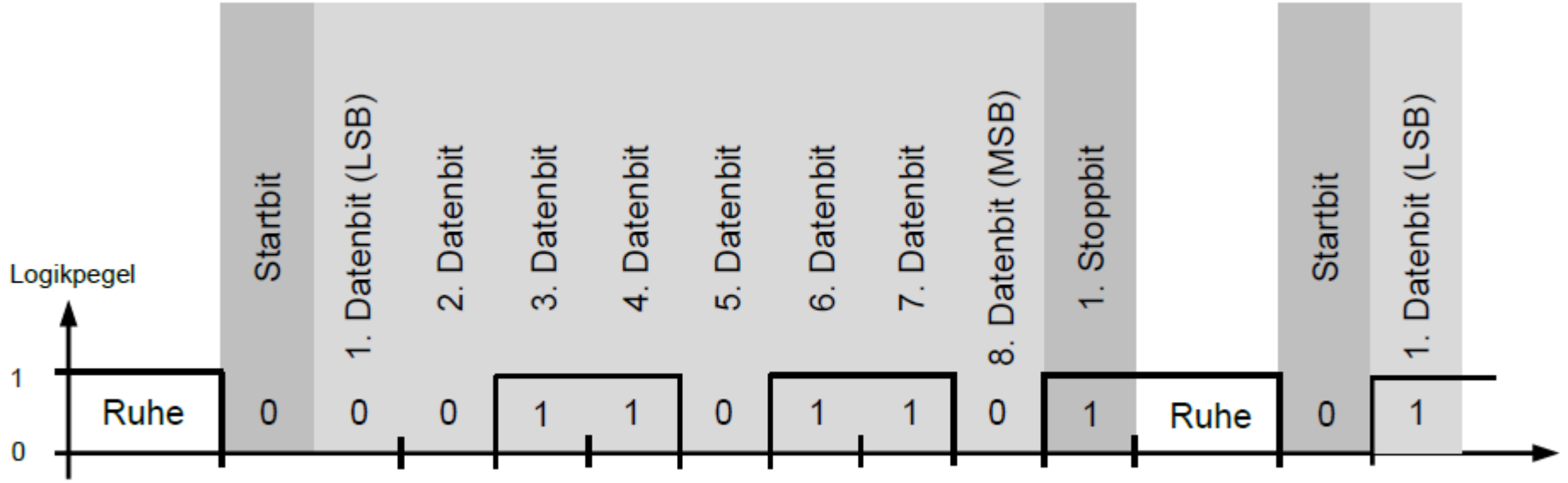
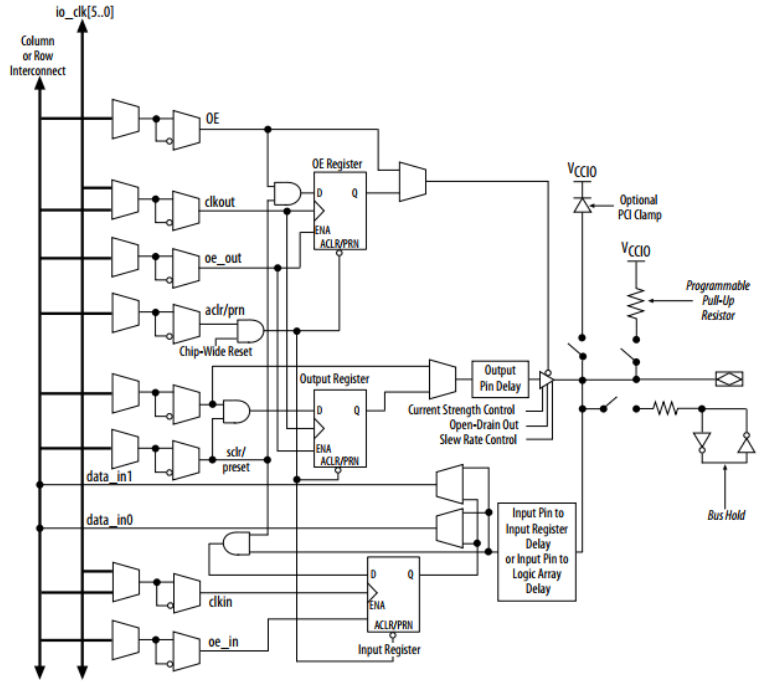


Abbildung 4-3 USART Übertragung (1/8/1)

© Prof. Dr. C. Jakob



USART – Steuer- und Konfigurationsregister

Fernstudiengang Elektrotechnik – SS2024

Prof. Dr. C. Jakob
University of Applied Sciences Darmstadt

fbeit
FACHBEREICH ELEKTROTECHNIK
UND INFORMATIONSTECHNIK

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

Die Konfiguration und Steuerung der Atmega328p internen USART Peripherie erfolgt im Wesentlichen über die Register:

- `UCSR0A` (USART Control and Status Register A)
- `UCSR0B` (USART Control and Status Register B)
- `UCSR0C` (USART Control and Status Register C)
- `UDR0` (USART Transmit and Receive Buffer)
- `UBRR0L` (USART Baud Rate Register Low)
- `UBRR0H` (USART Baud Rate Register High)

© Prof. Dr. C. Jakob

Bit	7	6	5	4	3	2	1	0	
(0xC0)	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0	UCSR0A
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

Abbildung 4-4 ATmega328p - USART – UCSR0A Steuerregister

- **Bit 7 – RXC0:** USART Receive Complete

Ist dieses Bit gesetzt, so befinden sich zu lesende Daten im Empfangspuffer. Ist es hingegen nicht gesetzt, so ist der Empfangspuffer leer oder die Empfangseinheit ist ausgeschaltet. Dieses Bit kann zudem bei gegebener Konfiguration einen Interrupt auszulösen (siehe `RXCIE0` Bit in Register `UCSR0B`).

- **Bit 6 – TXC0:** USART Transmit Complete

Ist dieses Bit nicht gesetzt, so sind noch zu übertragende Bits im Senderegister vorhanden. Es kann folglich noch kein neues Zeichen gesendet werden. Dieses Bit ist hingegen gesetzt, wenn alle zu übertragenden Bits (d.h. der gesamte Übertragungsrahmen) ausgegeben wurden und keine neuen Daten im `UDR0` Sendepuffer vorliegen.

© Prof. Dr. C. Jakob

Bit	7	6	5	4	3	2	1	0	
(0xC1)	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80	UCSR0B
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W	
Initial Value	0	0	0	0	0	0	0	0	

***Abbildung 4-5** ATmega328p - USART – UCSR0B Steuerregister*

- **Bit 7 – RXCIE0:** RX Complete Interrupt Enable

Ist dieses Bit gesetzt, so wird ein *„USART RX Complete Interrupt“* ausgelöst, wenn ein Zeichen vom USART empfangen wurde. Interrupts müssen in diesem Fall global freigeschaltet sein.

- **Bit 6 – TXCIE0:** TX Complete Interrupt Enable

Ist dieses Bit gesetzt, so wird ein *„USART TX Complete Interrupt“* ausgelöst, wenn ein Zeichen vom USART empfangen wurde. Interrupts müssen in diesem Fall global freigeschaltet sein.

- **Bit 5 – UDRIE0:** USART Data Register Empty Interrupt Enable

- **Bit 4 – RXEN0:** Receiver Enable

Über das Setzen dieses Bits kann die USART-Empfangseinheit eingeschaltet werden. In diesem Fall ist der zugehörige Pin RxD als Eingang reserviert und steht für andere Aktionen nicht zur Verfügung.

- **Bit 3 – TXEN0:** Transmitter Enable

Über das Setzen dieses Bits kann die USART-Sendeeinheit eingeschaltet werden. In diesem Fall ist der zugehörige Pin `TxD` als Eingang reserviert und steht für andere Aktionen nicht zur Verfügung.

© Prof. Dr. C. Jakob

Bit	7	6	5	4	3	2	1	0	
(0xC2)	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01	UCSZ00	UCPOL0	UCSR0C
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	1	1	0	

Abbildung 4-6 ATmega328p - USART – UCSR0C Steuerregister

- **Bit 7:6 – UMSEL01:0:** USART Mode Select

Durch die Konfiguration dieser Bits wird die Art der Kommunikation für die USART Schnittstelle definiert. Hierfür existieren die nachfolgend aufgeführten Konfigurationsmöglichkeiten:

- 00: Asynchrone USART
- 01 Synchrone USART
- 10 Reserviert (ohne Funktion)
- 11 Master SPI

- **Bit 5:4 – UPM01:0:** Parity Mode

Über dieses Bit wird ein mögliches Paritätsbit im Übertragungsrahmen definiert. Hierfür existieren folgende Konfigurationsmöglichkeiten:

- 00: Keine Parität
- 01 Reserviert (ohne Funktion)
- 10 Ungerade Parität
- 11 Gerade Parität

- **Bit 3 – USBS0:** Stop Bit Select

Über dieses Bit wird die Anzahl der Stoppbits im Übertragungsrahmen eingestellt. Ist dieses Bit gesetzt, so wird eine Übertragung mit zwei Stoppbits abgeschlossen. Andernfalls mit lediglich einem Stoppbit.

- **Bit 2:1 – UCSZ01:0:** Character Size

Über die drei Bit `UCSZ0` wird die Anzahl der zu übertragenden Datenbits festgelegt. Das Bit `UCSZ02` befindet sich im Steuerregister UCSR0B und wird lediglich für den Fall einer 9-bit Zeichengröße benötigt.

© Prof. Dr. C. Jakob

Bit	15	14	13	12	11	10	9	8	
(0xC5)	-	-	-	-	UBRR0[11:8]				**UBRR0H**
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Bit	7	6	5	4	3	2	1	0	
(0xC4)	UBRR0[7:0]								**UBRR0L**
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

***Abbildung 4-7** ATmega328p - USART – UBRR0L/UBRR0H Steuerregister*

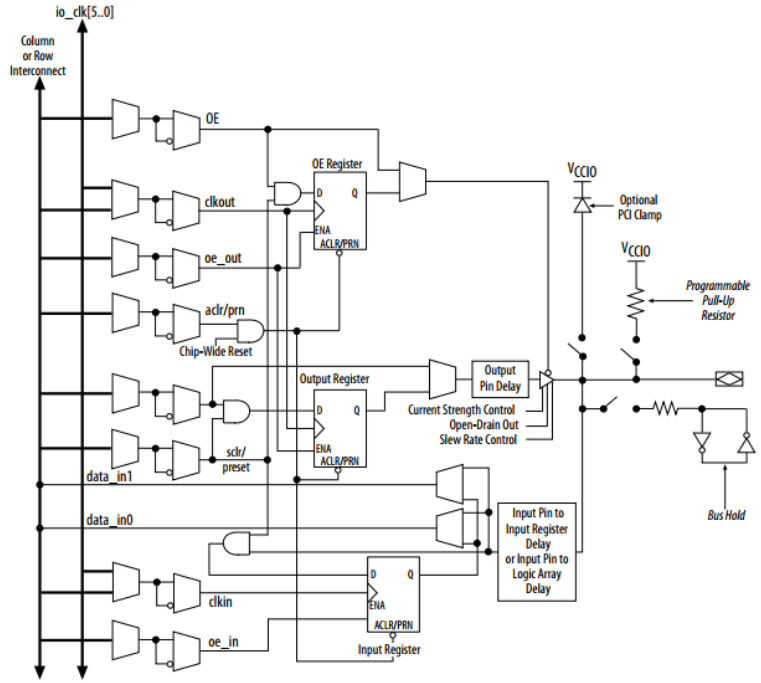
$$\text{Baudrate} = \frac{Systemtakt}{16(Teiler + 1)} \qquad \text{Teiler} = \frac{Systemtakt}{16\,Baudrate} - 1$$

© Prof. Dr. C. Jakob

Bit	7	6	5	4	3	2	1	0	
(0xC6)	RXB[7:0]								UDR0 (Read)
(0xC6)	TXB[7:0]								UDR0 (Write)
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

***Abbildung 4-8** ATmega328p - USART – UDR0 Register*

© Prof. Dr. C. Jakob



USART – Implementierung von Basistreibern

Fernstudiengang Elektrotechnik – SS2024

Prof. Dr. C. Jakob
University of Applied Sciences Darmstadt

fbeit
FACHBEREICH ELEKTROTECHNIK
UND INFORMATIONSTECHNIK

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

```
1. // USART Initialisierungsroutine, kein Rückgabewert, keine
2. // Übergabeparameter, Baudrate: wird über globales 'define' gesetzt
3. // 8-Datenbits, 1-Stoppbit, Gerade Parität, Asynchroner Modus
4. // Zielplattform: ATMEL ATmega328p, Arduino Uno R3 SMD Edition
5. // by Dr. C. Jakob, fbeit, h_da, Januar 2015, christian.jakob@h-da.de
6.
7. void usart_init(void){
8.      // Baudrate einstellen
9.      UBRR0H = (unsigned char)(BAUD_PRESCALLER >> 8);
10.     UBRR0L = (unsigned char)(BAUD_PRESCALLER);
11.     // asynchroner USART Modus
12.     UCSR0C &= ~((1 << UMSEL01)|(1 << UMSEL00));
13.     // 8-Datenbits
14.     UCSR0C |= (1 << UCSZ00)| (1 << UCSZ01);
15.     // 1-Stoppbit
16.     UCSR0C &= ~(1 << USBS0);
17.     // USART Sende- und Empfangseinheiten aktivieren
18.     UCSR0B = (1 << RXEN0)|(1 << TXEN0);
19. }
```

***Quellcode 4-1** ATMEL ATmega328p USART Initialisierungsroutine*

© Prof. Dr. C. Jakob

```
1. // USART Empfangsroutine - 8-bit Rückgabewert, keine Übergabeparameter
2. // Zielplattform: ATMEL ATmega328p, Arduino Uno R3 SMD Edition
3. // by Dr. C. Jakob, fbeit, h_da, Januar 2015, christian.jakob@h-da.de
4.
5. unsigned char usart_receive(void) {
6.      while((UCSR0A & (1 << RXC0)) == 0);   // Daten verfügbar?
7.      return UDR0;
8. }
```

***Quellcode 4-2** ATMEL ATmega328p USART 8-bit Empfangsroutine*

© Prof. Dr. C. Jakob

```
1. // USART Senderoutine - kein Rückgabewert, 8-bit Übergabeparameter
2. // Zielplattform: ATMEL ATmega328p, Arduino Uno R3 SMD Edition
3. // by Dr. C. Jakob, fbeit, h_da, Januar 2015, christian.jakob@h-da.de
4.
5. void usart_put_byte(unsigned char data){
6.      while((UCSR0A & (1 << UDRE0)) == 0);  // Senderegister frei?
7.      UDR0 = data;
8. }
```

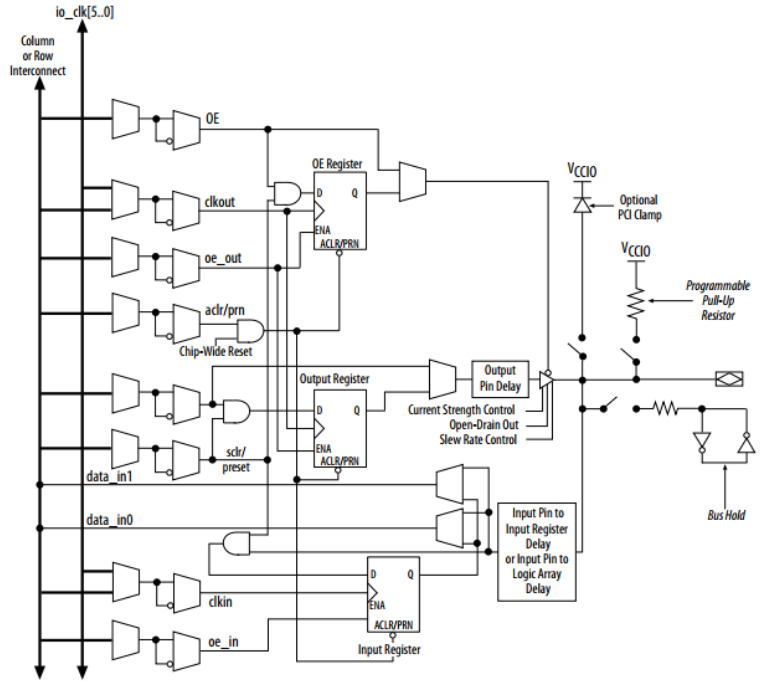
***Quellcode 4-3** ATMEL ATmeag328p USART ‚put_byte' Senderoutine*

© Prof. Dr. C. Jakob

```
1. // USART Senderoutine - kein Rückgabewert, Pointer auf eine Variable
2. // vom Typ ‚character' - Nullterminierung erforderlich
2. // Zielplattform: ATMEL ATmega328p, Arduino Uno R3 SMD Edition
3. // by Dr. C. Jakob, fbeit, h_da, Januar 2015, christian.jakob@h-da.de
4.
5. void usart_put_string(char* string_ptr){
6.      while(*string_ptr != 0x00){
7.         usart_put_byte(*string_ptr);
8.         string_ptr++;
9.      }
10. }
```

***Quellcode 4-4** ATMEL ATmega328p USART 'put_string' Senderoutine*

© Prof. Dr. C. Jakob



Polling basierte USART Konfiguration

Fernstudiengang Elektrotechnik – SS2024

Prof. Dr. C. Jakob
University of Applied Sciences Darmstadt

fbeit
FACHBEREICH ELEKTROTECHNIK
UND INFORMATIONSTECHNIK

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

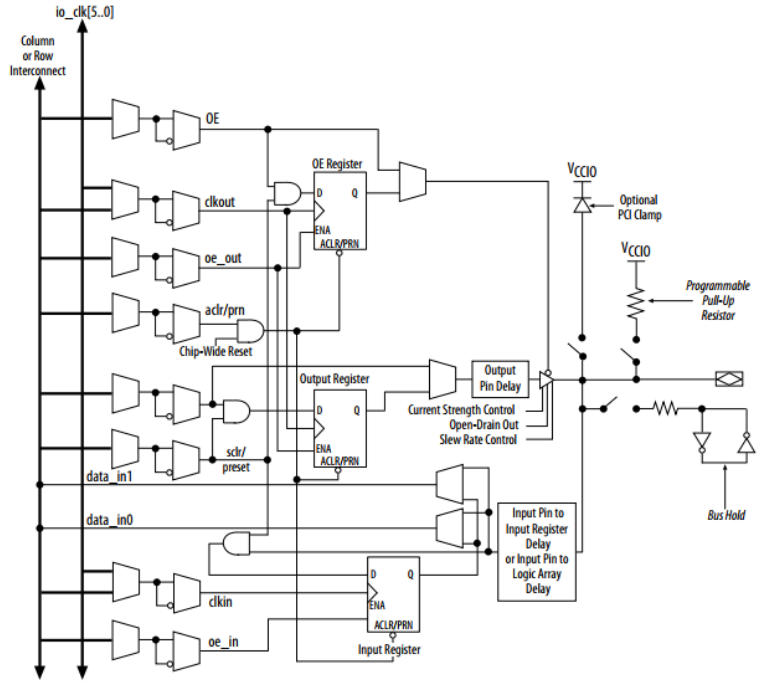
```
1. // ATmega328p USART Demo - Polling Modus
2. // Zielplattform: ATMEL ATmega328p, Arduino Uno R3 SMD Edition
3. // by Dr. C. Jakob, fbeit, h_da, Januar 2015, christian.jakob@h-da.de
4.
5.  #define F_CPU 16000000UL
6.  #define BAUDRATE 9600
7.  #define BAUD_PRESCALLER (((F_CPU / (BAUDRATE * 16UL))) - 1)
8.  #define LINE_FEED  10
9.
10. #include <avr/io.h>
11. #include <util/delay.h>
12. #include 'usart.h'
13.
14. char my_string[] = "Command received!";
15.
16. int main(void){
17.     unsigned char rec_byte;
18.     // -- Konfiguration Status-LED --
19.     DDRB  |=  (1 << DDB5);         // PIN PB5 als Ausgang konfigurieren
20.     PORTB &= ~(1 << PORTB5);       // LED ausschalten
21.     // -- Initialisierung USART Schnittstelle --
22.     usart_init();
23.
```


© Prof. Dr. C. Jakob

```
24.     while(1){
25.         rec_byte = usart_receive();
26.         if(rec_byte == 'a'){
27.             PORTB ^= (1 << PORTB5);
28.             usart_put_string(my_string);
29.             usart_put_byte(LINE_FEED);
30.         }
31.         else {
32.             PORTB &= ~(1 << PORTB5);
33.         }
34.         _delay_ms(500);
35.     }
36.     return 0;
37. }
```

Quellcode 4-5 ATmega328p USART Demo – Polling Modus

© Prof. Dr. C. Jakob



IRQ basierte USART Konfiguration

Fernstudiengang Elektrotechnik – SS2024

Prof. Dr. C. Jakob
University of Applied Sciences Darmstadt

fbeit
FACHBEREICH ELEKTROTECHNIK
UND INFORMATIONSTECHNIK

h_da
HOCHSCHULE DARMSTADT
UNIVERSITY OF APPLIED SCIENCES

```
1. // ATmega328p USART Demo - Interrupts Modus - Empfangsinterrupt
2. // Zielplattform: ATMEL ATmega328p, Arduino Uno R3 SMD Edition
3. // by Dr. C. Jakob, fbeit, h_da, Januar 2015, christian.jakob@h-da.de
4.
5.  #define F_CPU 16000000UL
6.  #define BAUDRATE 9600
7.  #define BAUD_PRESCALLER (((F_CPU / (BAUDRATE * 16UL))) - 1)
8.  #define LINE_FEED  10
9.
10. #include <avr/io.h>
11. #include <util/delay.h>
12. #include 'usart.h'
13.
14. volatile unsigned char rec_byte;  // ISR Variable
15.
16. char my_string[] = "Command received!";
17.
18. int main(void){
19.     // -- Konfiguration Status-LED --
20.     DDRB  |=  (1 << DDB5);         // PIN PB5 als Ausgang konfigurieren
21.     PORTB &= ~(1 << PORTB5);       // LED ausschalten
22.     // -- Initialisierung USART Schnittstelle --
23.     usart_init();
24.     // -- Aktivierung des ‚USART Receive' Interrupts -
25.     UCSR0B |= (1 << RXCIE0);
```


© Prof. Dr. C. Jakob

```
27.     sei();                          // globale Interrupts aktivieren
28.
29.     while(1){
30.        if(rec_byte == 'a'){
31.             PORTB ^= (1 << PORTB5);
32.             usart_put_string(my_string);
33.             usart_put_byte(LINE_FEED);
34.             rec_byte = 0;
35.        }
36.        else {
37.             PORTB &= ~(1 << PORTB5);
38.        }
39.        _delay_ms(500);
40.     }
41.     return 0;
42. }
43.
44. ISR (USART_RXC_vect){              // Empfangenes Byte auslesen ...
45.     rec_byte = UDR0;
46. }
```

***Quellcode 4-6** ATMEL ATmega328p USART Demo – Interrupt Modus – Empfangsinterrupt*

© Prof. Dr. C. Jakob