USART

ATmega128

Universal Synchronous and Asynchronous serial Receiver and Transmitter

- 2 independent units (USART0, USART1)
- full duplex
- synchronous or asynchronous
- synchronous master/slave
- 5,6,7,8,9 data bits + 1,2 stop bits
- even/odd parity, parity check in HW
- Interrupts: TX complete, TX DRE, RX complete

"115200 8N1" ?

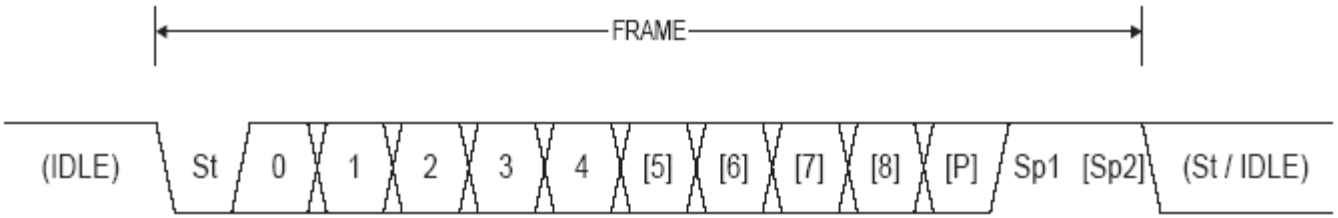
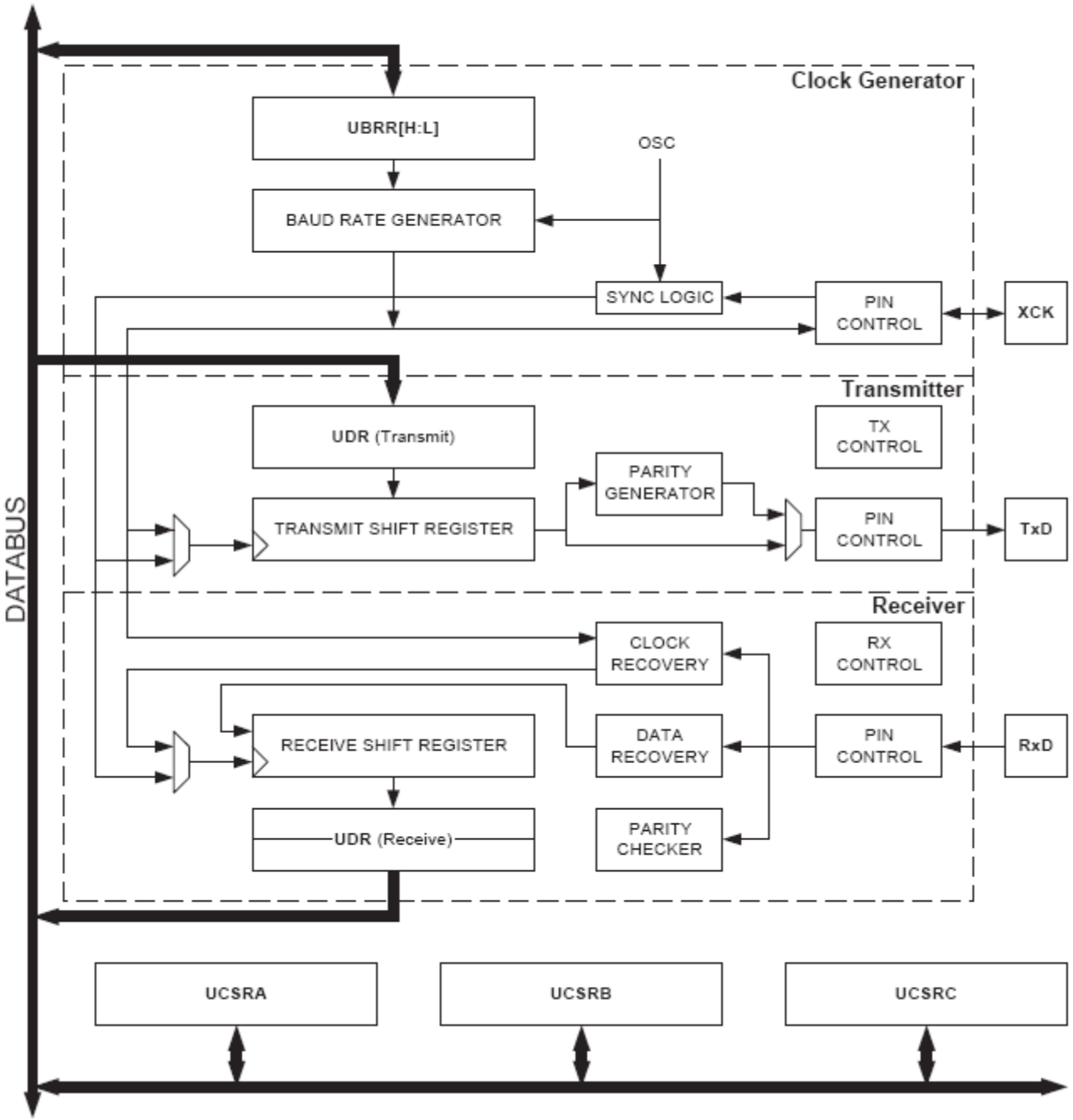
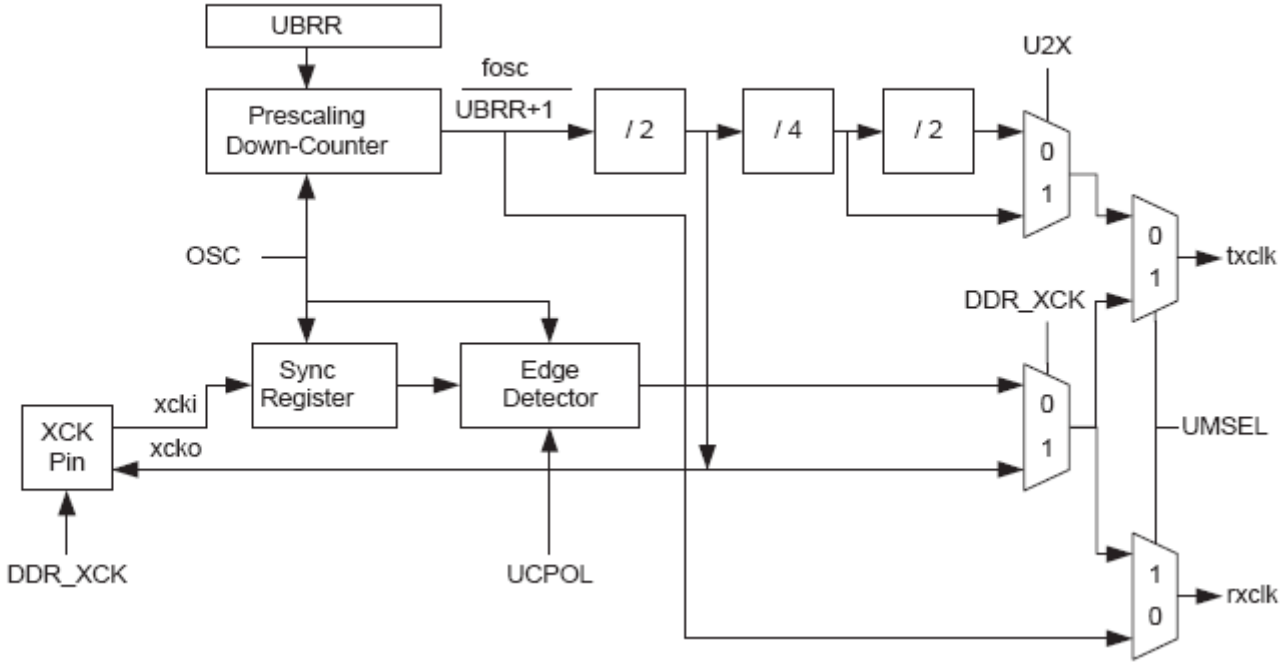


Figure 79. USART Block Diagram



Clock generation

Figure 80. Clock Generation Logic, Block Diagram



Clock Mode

- normal asynchronous
- double asynchronous
- master synchronous
- slave synchronous

- Control registers:
  - USCRC:UMSEL … synchronous / asynchronous
  - UCSRA:U2X … normal / double
  - DDR_XCK … synchronous master / slave
    - USART0: DDRE:2, USART1: DDRD:5

Baud Rate control

- USART Baud Rate Register (UBRR)

Table 52. Equations for Calculating Baud Rate Register Setting

Operating Mode	Equation for Calculating Baud Rate[1]	Equation for Calculating UBRR Value
Asynchronous Normal mode (U2X = 0)	$BAUD = \frac{f_{OSC}}{16(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{16BAUD} - 1$
Asynchronous Double Speed Mode (U2X = 1)	$BAUD = \frac{f_{OSC}}{8(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{8BAUD} - 1$
Synchronous Master Mode	$BAUD = \frac{f_{OSC}}{2(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{2BAUD} - 1$

(Slave – clock is external from master)

Baud Rate register value example

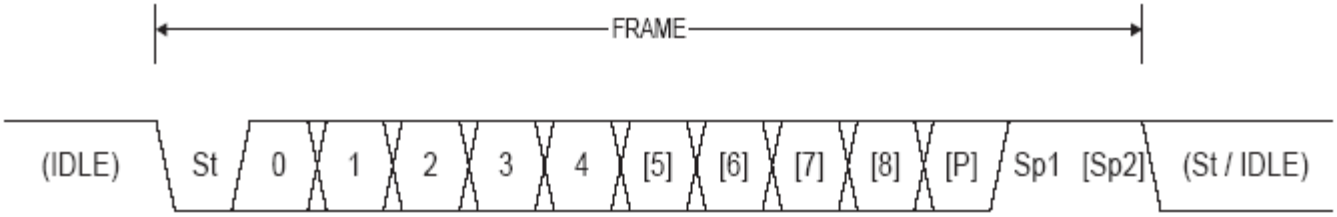
f_{osc} = 16 MHz	U2X = 0		U2X = 1	
Baud Rate (bps)	UBRR	Error	UBRR	Error
2 400	416	-0.1%	832	0.0%
4 800	207	0.2%	416	-0.1%
9 600	103	0.2%	207	0.2%
14 400	68	0.6%	138	-0.1%
19 200	51	0.2%	103	0.2%
28 800	34	-0.8%	68	0.6%
38 400	25	0.2%	51	0.2%
57 600	16	2.1%	34	-0.8%
76 800	12	0.2%	25	0.2%
115 200	8	-3.5%	16	2.1%
230 400	3	8.5%	8	-3.5%
250 000	3	0.0%	7	0.0%
500 000	1	0.0%	3	0.0%
1 000 000	0	0.0%	1	0.0%

Baud Rate register value example

f_{osc} = 1 MHz	U2X = 0		U2X = 1	
Baud Rate (bps)	UBRR	Error	UBRR	Error
2 400	25	0.2%	51	0.2%
4 800	12	0.2%	25	0.2%
9 600	6	-7.0%	12	0.2%
14 400	3	8.5%	8	-3.5%
19 200	2	8.5%	6	-7.0%
28 800	1	8.5%	3	8.5%
38 400	1	-18.6%	2	8.5%
57 600	0	8.5%	1	8.5%
76 800	-	-	1	-18.6%
115 200	-	-	0	8.5%
230 400	-	-	-	-
250 000	-	-	-	-
500 000	-	-	-	-
1 000 000	-	-	-	-

Frames

- 1 start bit
- 5-9 data bits (UCSRC:UCSZ2:0)
- no/odd/even parity (UCSRC:UPM1:0)
- 1-2 stop bits (UCSRB:USBS)



USARTn
Control and Status Register A

UCSRnA

7	6	5	4	3	2	1	0
RXCn	TXCn	UDREn	FEn	DORn	PEn	U2Xn	MPCMn
R	R/W	R	R	R	R	R/W	R/W
0	0	1	0	0	0	0	0

RXCn USART Receive Complete

TXCn USART Transmit Complete

UDREn USART Data Register Empty

FEn Frame Error

DORn Data OverRun

PEn Parity Error

U2Xn USART Double Transmission Speed

MPCMn Multi-Processor Communication Mode

USARTn
Control and Status Register B

UCSRnB

7	6	5	4	3	2	1	0
RXCIEn	TXCIEn	UDRIEn	RXENn	TXENn	UCSZn2	RXB8n	TXB8n
R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
0	0	0	0	0	0	0	0

RXCIEn RX Complete Interrupt Enable

TXCIEn TX Complete Interrupt Enable

UDRIEn Data Register Empty Interrupt Enable

RXENn Receiver Enable

TXENn Transmitter Enable

UCSZn2 Character Size

RXB8n Receive Data Bit 8

TXB8n Transmit Data Bit 8

USARTn
Control and Status Register C

UCSRnC	7	6	5	4	3	2	1	0
	-	UMSELn	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
	0	0	0	0	0	1	1	0

UMSELn	USART Mode Select
UPMn1:0	Parity Mode
USBSn	Stop Bit Select
UCSZn1:0	Character Size
UCPOLn	Clock Parity

UBRRnH, UBRRnL

UBRRnH

7	6	5	4	3	2	1	0
-	-	-	-	UBRRn11:8			
R	R	R	R	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

UBRRnL

7	6	5	4	3	2	1	0
UBRRnL7:0							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

UBBR11:0 USART Baud Rate Register

USARTn
I/O Data Register

UDRn

7	6	5	4	3	2	1	0
RXBn7:0 / TXBn7:0							
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

Inicialization

- mode
- baud rate
- frame format
- interrupt?
- enable

Inicialization example

```
USART_Init:
    ; Set baud rate
    out UBRRH, r17
    out UBRRL, r16
    ; Enable receiver and transmitter
    ldi r16, (1<<RXEN)|(1<<TXEN)
    out UCSRB,r16
    ; Set frame format: 8data, 2stop bit
    ldi r16, (1<<USBS)|(3<<UCSZ0)
    out UCSRC,r16
    ret
```

Personal note:
This and the following examples were copied from official ATmega128 datasheet. I would prefer first setting everything and only as the last thing enabling the receiver and/or transmitter. Changing the communication parameters on the fly is not a good idea – even the risk of garbled communication is low, it _could_ happen and your application could fail as a result. It would be _your_ fault then … *(But, at the same time, did you read the Legal Notice datasheet section? No? Do it.)*

Secondly, and particularly for these two examples, I would prefer passing arguments the same way in assembly as it is done in C (and what Atmel recommends to do for assembler ↔ C coexistence). *(Find out, how…)*

Inicialization example

```
#define FOSC 1843200            // Clock Speed
#define BAUD 9600
#define MYUBRR FOSC/16/BAUD-1
void main( void )
{
    ...
    USART_Init ( MYUBRR );
    ...
}
void USART_Init( unsigned int ubrr )
{
    /* Set baud rate */
    UBRRH = (unsigned char)(ubrr>>8);
    UBRRL = (unsigned char)ubrr;
    /* Enable receiver and transmitter */
    UCSRB = (1<<RXEN)|(1<<TXEN);
    /* Set frame format: 8data, 2stop bit */
    UCSRC = (1<<USBS)|(3<<UCSZ0);
}
```