Embedded Software

CS 145/145L

Caio Batista de Melo

Agenda

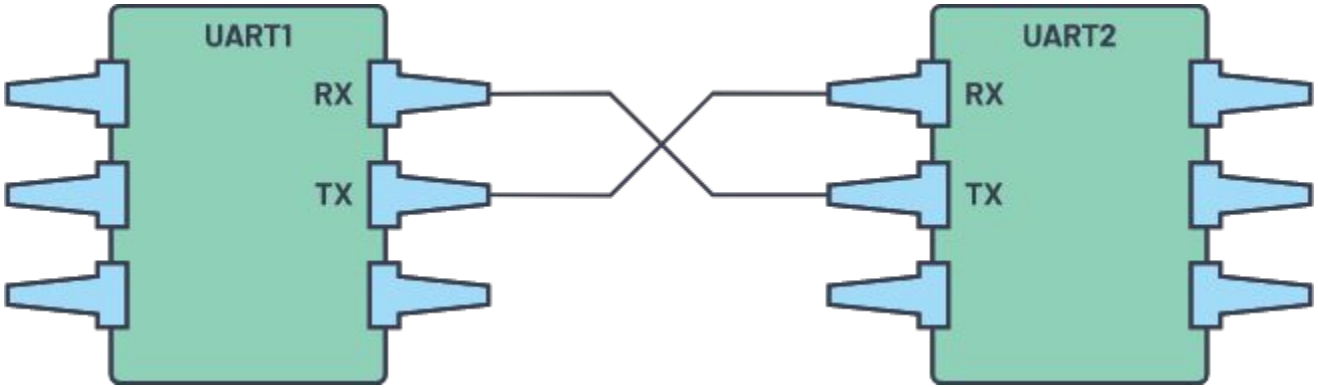
- Universal Asynchronous Receiver/Transmitter (UART)
- UART on ATmega32
- Example

Universal Asynchronous Receiver/Transmitter (UART)

Universal Asynchronous Receiver/Transmitter (UART)

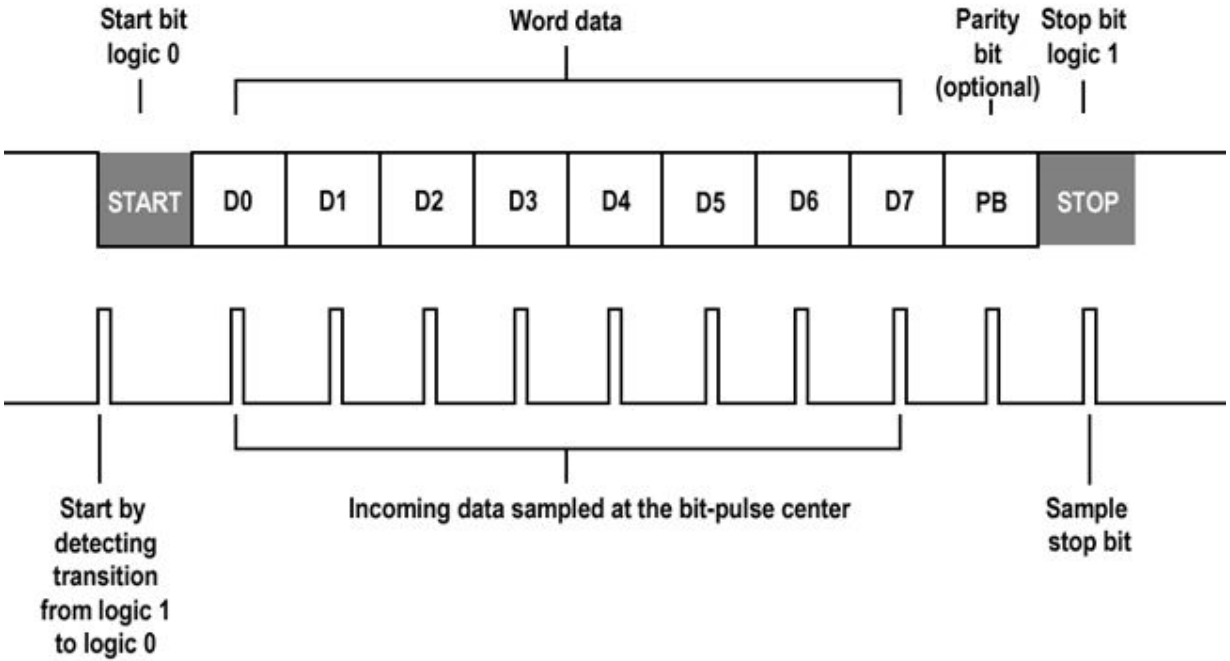
- UART is a protocol for device-to-device communication
- It's the predecessor to USB
- Still used in microcontrollers to communicate with each other and other devices

UART Pins



https://www.analog.com/en/analog-dialogue/articles/uart-a-hardware-communication-protocol.html

UART Packet



https://developer.electricimp.com/resources/uart

Baud Rate

- Both devices should sample the channel at the same rate
- This is called the baud rate
- For example, "9600 baud" means the channel is capable of transferring 9600 bps

UART on ATmega32

UART vs USART

- USART: Universal Synchronous Asynchronous Receiver/Transmitter
- USART can be imagined as a superset of UART
  - supports everything UART does
  - can also do synchronous operations
- ATmega32 has built-in USART

ATmega32 USART Registers

- UBRR: USART Baud Rate Register
- UDR: USART Data Register
- UCSRA: USART Control and Status Register A
- UCSRB: USART Control and Status Register B
- UCSRC: USART Control and Status Register C

UBRR: USART Baud Rate Register
Page 164 of the manual

USART Baud Rate Registers – UBRRL and UBRRH

Bit	15	14	13	12	11	10	9	8	
	URSEL	–	–	–	UBRR[11:8]				UBRRH
	UBRR[7:0]								UBRRL
	7	6	5	4	3	2	1	0	
Read/Write	R/W	R	R	R	R/W	R/W	R/W	R/W	
	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	
	0	0	0	0	0	0	0	0	

The UBRRH Register shares the same I/O location as the UCSRC Register. See the "Accessing UBRRH/ UCSRC Registers" on page 158 section which describes how to access this register.

- **Bit 15 – URSEL: Register Select**

This bit selects between accessing the UBRRH or the UCSRC Register. It is read as zero when reading UBRRH. The URSEL must be zero when writing the UBRRH.

UDR: USART Data Register
Page 159 of the manual

USART I/O Data Register – UDR

Bit	7	6	5	4	3	2	1	0	
	RXB[7:0]								UDR (Read)
	TXB[7:0]								UDR (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	

The USART Transmit Data Buffer Register and USART Receive Data Buffer Registers share the same I/O address referred to as USART Data Register or UDR. The Transmit Data Buffer Register (TXB) will be the destination for data written to the UDR Register location. Reading the UDR Register location will return the contents of the Receive Data Buffer Register (RXB).

UCSRA: USART Control and Status Register A
Page 160 of the manual

USART Control and Status Register A – UCSRA

Bit	7	6	5	4	3	2	1	0	
	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	UCSRA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

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

This flag bit is set when there are unread data in the receive buffer and cleared when the receive buffer is empty (i.e., does not contain any unread data). If the receiver is disabled, the receive buffer will be flushed and consequently the RXC bit will become zero. The RXC Flag can be used to generate a Receive Complete interrupt (see description of the RXCIE bit).

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

This flag bit is set when the entire frame in the transmit Shift Register has been shifted out and there are no new data currently present in the transmit buffer (UDR). The TXC Flag bit is automatically cleared when a transmit complete interrupt is executed, or it can be cleared by writing a one to its bit location. The TXC Flag can generate a Transmit Complete interrupt (see description of the TXCIE bit).

- **Bit 5 – UDRE: USART Data Register Empty**

The UDRE Flag indicates if the transmit buffer (UDR) is ready to receive new data. If UDRE is one, the buffer is empty, and therefore ready to be written. The UDRE Flag can generate a Data Register empty Interrupt (see description of the UDRIE bit).

UDRE is set after a reset to indicate that the transmitter is ready.

UCSRB: USART Control and Status Register B
Page 161 of the manual

USART Control and Status Register B – UCSRB

Bit	7	6	5	4	3	2	1	0	
	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	UCSRB
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	

- **Bit 4 – RXEN: Receiver Enable**

Writing this bit to one enables the USART Receiver. The Receiver will override normal port operation for the RxD pin when enabled. Disabling the Receiver will flush the receive buffer invalidating the FE, DOR, and PE Flags.

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

Writing this bit to one enables the USART Transmitter. The Transmitter will override normal port operation for the TxD pin when enabled. The disabling of the Transmitter (writing TXEN to zero) will not become effective until ongoing and pending transmissions are completed, i.e., when the transmit Shift Register and transmit Buffer Register do not contain data to be transmitted. When disabled, the transmitter will no longer override the TxD port.

UCSRC: USART Control and Status Register C
Page 162 of the manual

USART Control and Status Register C – UCSRC

Bit	7	6	5	4	3	2	1	0	
	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	UCSRC
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	1	0	0	0	0	1	1	0	

- **Bit 7 – URSEL: Register Select**

This bit selects between accessing the UCSRC or the UBRRH Register. It is read as one when reading UCSRC. The URSEL must be one when writing the UCSRC.

- **Bit 2:1 – UCSZ1:0: Character Size**

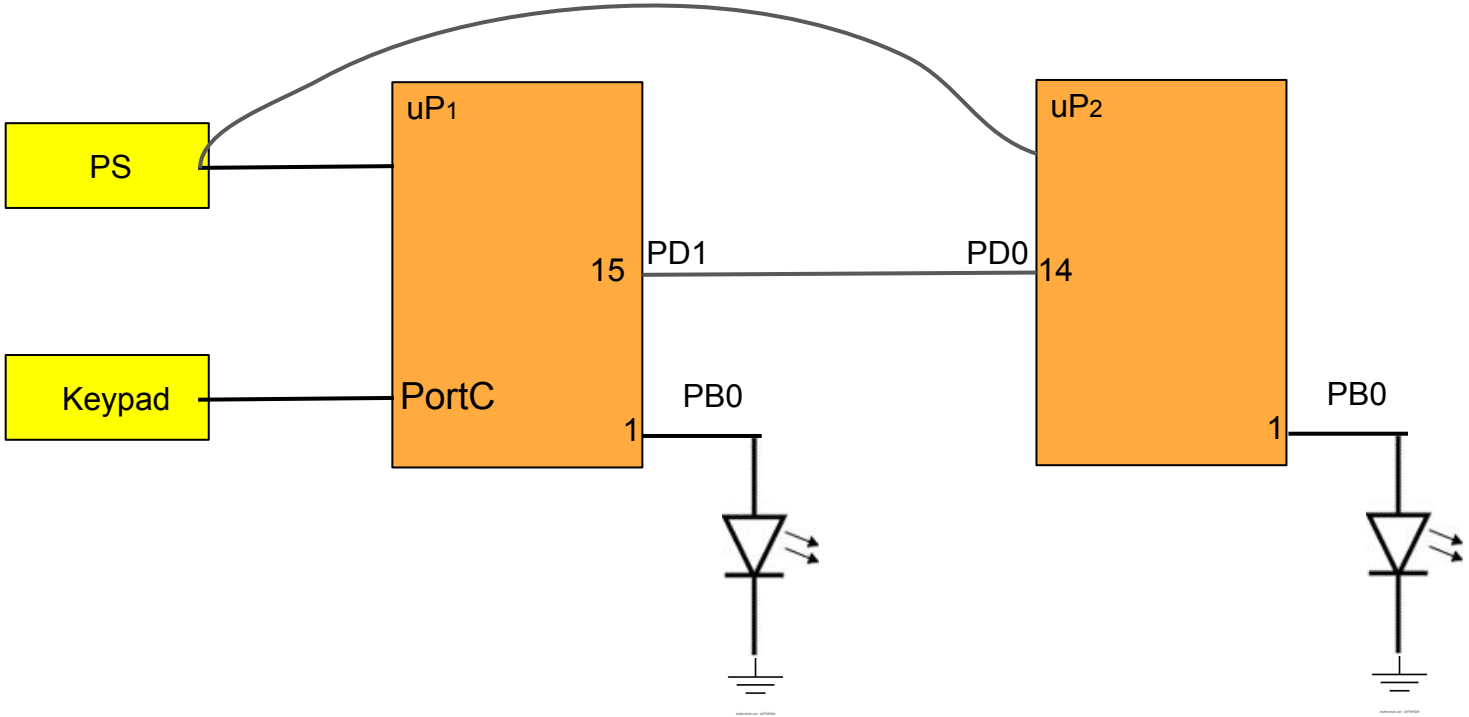
The UCSZ1:0 bits combined with the UCSZ2 bit in UCSRB sets the number of data bits (Character Size) in a frame the Receiver and Transmitter use.

Table 66. UCSZ Bits Settings

UCSZ2	UCSZ1	UCSZ0	Character Size
0	0	0	5-bit
0	0	1	6-bit
0	1	0	7-bit
0	1	1	8-bit
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Reserved
1	1	1	9-bit

Example

Hardware Setup



Software Setup

Create a new library (uart.h/c) that defines functions for transmission & reception.

Available at:

- https://caiobatista.com/uploads/courses/uci/s22/cs145/uart.h
- https://caiobatista.com/uploads/courses/uci/s22/cs145/uart.c

uart.h

```
#ifndef __UART_H__
#define __UART_H__

#include "avr.h"

// Initialize ATmega32 for transmission and reception.
void uart_init (void);
// Receive a single byte.
void uart_transmit (unsigned char data);
// Send a single byte.
unsigned char uart_receive (void);

#endif
```

uart.c (1/2)

```c
#include "uart.h"

// Calculate UBBR value for 9600 BAUD rate
#define BAUD 9600
#define BAUDRATE ((F_CPU)/(BAUD*16UL)-1)

void uart_init (void) {
    // Configure UART for reception and transmission of 8 bits
    UBRRH = (BAUDRATE>>8);
    UBRRL = BAUDRATE;
    UCSRB|= (1<<TXEN)|(1<<RXEN);
    UCSRC|= (1<<URSEL)|(1<<UCSZ0)|(1<<UCSZ1);
}
```

Calculates and sets baud

Enable reception and transmission

8 bits of data

uart.c (2/2)

Page 147

```c
void uart_transmit (unsigned char data) {
    /* Wait for empty transmit buffer */
    while (!(UCSRA & (1<<UDRE)));
    /* Put data into buffer, sends the data */
    UDR = data;
}
```

Page 150

```c
unsigned char uart_receive (void) {
    /* Wait for data to be received */
    while(!(UCSRA & (1<<RXC)));
    /* Get and return received data from buffer */
    return UDR;
}
```

Main Loops

```
while (1) {
  key = get_key();
  if (key) {
    uart_transmit(key);
    while (key--) {
      SET_BIT(PORTB, 0);
      avr_wait(500);
      CLR_BIT(PORTB, 0);
      avr_wait(500);
    }
  }
}
```

uP1

uP2

```
while (1) {
  key = uart_receive();
  while (key--) {
    SET_BIT(PORTB, 0);
    avr_wait(500);
    CLR_BIT(PORTB, 0);
    avr_wait(500);
  }
}
```

Need to include things, set DDR, define variables, call uart_init() before loops

See you next time :)

Q & A