CprE 288 – Introduction to Embedded Systems

Instructors:
Dr. Zhao Zhang (Sections A, B, C, D, E)
Dr. Phillip Jones (Sections F, G, J)

Overview of the Lecture

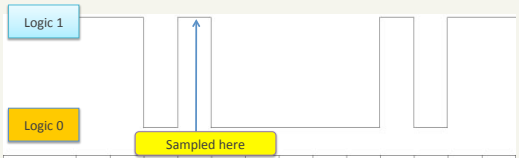
- Concepts behind Serial Communication
- ATmega128 USART Programming Interface
- Initializing USART, transmitting and receiving data

Serial Communication

- USART = Universal Synchronous & Asynchronous Serial Receiver & Transmitter
- Asynchronous (no common clock)
- Can transmit over long link distances
- Uses *start* and *stop* to sandwich data bits
- *parity bit* can be used for error detection

- (on right) RS-232 Serial Cable

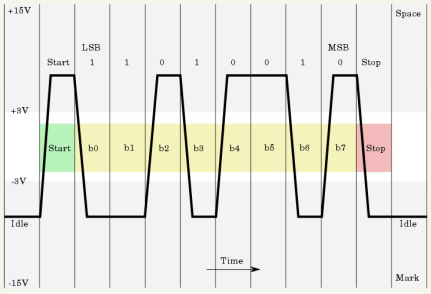
Serial Byte Format



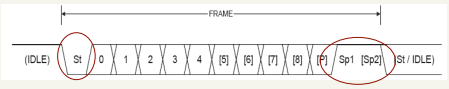
State	Idle	Idle	Active	Active	Active	Active	Active	Active	Active	Active	Active	Idle
Logic Level	1	1	0	1	0	0	0	0	0	1	0	1	1
Bit Type			Start	Data	Data	Data	Data	Data	Data	Data	Data	Stop	
Data Bit Number				0	1	2	3	4	5	6	7		

Example: Sending byte value 01000001

Serial Communication

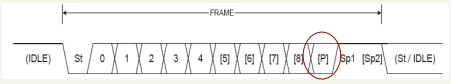


Start and Stop Bits



Idle period: logic high
Start bit: **logic low**, 1 bit
Stop bit: **logic high**, 1 bit or 2 bits

Parity Bit



Three choices: **even, odd, or none**

$$P_{even} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 0$$
$$P_{odd} = d_{n-1} \oplus \ldots \oplus d_3 \oplus d_2 \oplus d_1 \oplus d_0 \oplus 1$$

*If **one bit** is flipped, how to detect it?*

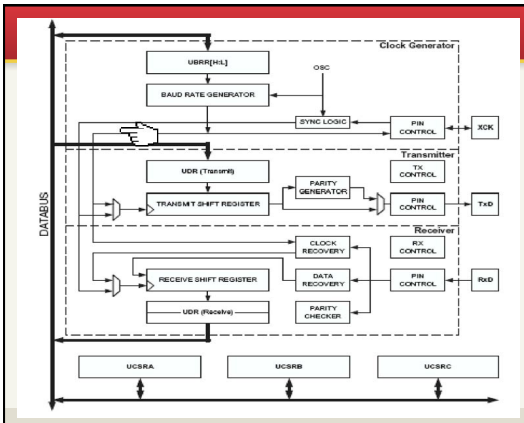
Baud Rate

How to define communication speed?

Baud rate: Number of symbols transferred per second
- Same as bit rate (bps) for USART

Baud rate is **not** data rate

With 56,000 bps, 8-bit frame, two stop bits and parity bit used, what is the maximum data rate?



Programming USART

Both sides of communication should use the same **frame format** and **baud rate**

Frame format:
- Number of bits in the frame: 5, 6, 7, 8 or 9
- Number of of stop bits: 1 or 2
- Parity bit: Odd, Even, or None

USART Programming Interface

UCSRnA, UCSRnB, UCSRnC: Control and Status Registers
- Three 8-bit registers for control and status checking
- *n is either 0 or 1*, e.g. **UCSR0A** is for USART0
- There are two USART units, USART0 and USART1; USART1 used for communication with iRobot Create

UBRRnH and UBRRnL: Baud Rate Registers
- Two 8-bit registers used together as 16-bit register

UDR*n*: 8-bit Register for reading and writing data

Datasheet Page Numbers

- Trying to set UCSRnA/B/C of the USART?
  - Review **pages 188 to 192** of the ATmega128 User Guide
- Setting the baud rate register (UBRR0)?
  - See the **table on page 196** of the ATmega128 User Guide
- Need code examples for reading / writing data?
  - **Page 176** (initialization example)
  - **Page 177** (transmit example)
  - **Page 180** (receive example)
  - Also reproduced on upcoming slides

Serial (ATmega128)

- This time will be spent reviewing the individual bit positions inside of **UCSR0A, UCSR0B, UCSR0C** from **pages 188 to 192** of the ATmega128 data sheet

UCSRnA: Mostly a Status Register

RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM

- **Bit 7 – RXCn:** 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

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

This flag bit is set when the entire frame in the Transmit Shift Register has been shifted out

- **Bit 5 – UDREn:** USART Data Register Empty
- **Bit 4 – FEn:** Frame Error
- **Bit 3 – DORn:** Data OverRun
- **Bit 2 – UPEn:** Parity Error
- **Bit 1 – U2Xn:** Double the USART Transmission Speed
- **Bit 0 – MPCMn:** Multi-Processor Communication Mode

UCSRnB: Mostly a Control Register

RXCIE	TXCIIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8

- **RXCIE, TXCIE, UDRIE**: Receive, Transmit, UDR interrupt enable
- **RXEN, TXEN**: Receive, Transmit enable
- **UCSZn2**: To decide number of bits in the frame (see also UCSZn1 and UCSZn0)
- RXB8n and TXB8n: Used in 9-bit frame setting. Not to be used in this course

UCSRnC: A Control Register

-	UMSEL	UPM1	UPM0	USB	UCSZ1	UCSZ0	UCPOL

- **UMSEL**: Asynchronous or Synchronous
  - 0 for Async, 1 for Sync; always use 0 only in this course
- **UPM1-0**: Parity mode – 00 (disabled), 10 (even), 11(odd)
- **USB: stop bit select:** 0 (1 stop bit), 1 (2 stop bits)
- **UCSZ2-0:** Number of bits in a frame
  - 000 (5-bits), 001 (6), 010 (7), 011 (8), 111 (9)
- **UCPOL:** Invert the polarity (invert logic low and high)
  - Use 0 in this course

Serial (ATmega128)

- Baud rate
  - 1 *baud* = 1 *symbol per second*
  - In our case, 8 data bits are book ended by start and stop bits, thus a baud is greater than 8 bits
- **Baud rate** is different from **data rate**
  - Data rate is faster
  - Baud rate includes overhead of start/stop/parity bits

Initialization

- UBRR0 **does not contain the baud rate**
  - In order to keep sameness across microprocessors, AVR uses the following formula to calculate the value stored in the baud rate register

Table 74. 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	$BAUD = \frac{f_{OSC}}{2(UBRR+1)}$	$UBRR = \frac{f_{OSC}}{2BAUD} - 1$

Example UBRR Settings

Baud Rate (bps)	f_{osc} = 16.0000 MHz			
	U2X = 0		U2X = 1	
	UBRR	Error	UBRR	Error
2400	416	-0.1%	832	0.0%
4800	207	0.2%	416	-0.1%
9600	103	0.2%	207	0.2%
14.4k	68	0.6%	138	-0.1%
19.2k	51	0.2%	103	0.2%
28.8k	34	-0.8%	68	0.6%
38.4k	25	0.2%	51	0.2%
57.6k	16	2.1%	34	-0.8%

U2X for double speed

Initialization

```
//Initialize USART0 to a given baud rate
void serial_init(unsigned long baud) {
  baud = (F_CPU / 8 / baud) - 1;
  // baud = (F_CPU / 16 / baud) - 1;
  // if not using double speed mode

  /* Set baud rate */
  UBRR0H = (unsigned char) (baud >> 8);
  UBRR0L = (unsigned char)baud;
  / * Enable double speed mode */
  UCSR0A = 0b00000010;
  /* Set frame format: 8 data bits, 1 stop bits */
  UCSR0C = 0b00000110;
  /* Enable receiver and transmitter */
  UCSR0B = 0b00011000;
  // UCSR0B |= 0b10000000; // optional: receive interrupt enable bit
}
```


Transmitting

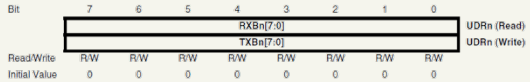
```
//Transmit a piece of data
void serial_putc(char data) {
    /* Wait for empty transmit buffer by checking the UDRE bit */
    while ((UCSR0A & 0b00100000) == 0)
        ;

    /* Put data into transmit buffer; sends the data */
    UDR0 = data;
}
```


Receiving

```
//Receive data
char serial_getc() {
   /* Wait for the receive complete flag (RXC) */
   while ((UCSR0A & 0b10000000) == 0)
      ;
   /* Reads data from the receive buffer; clears the
receive buffer */
   return UDR0;
}
```


Warning!

- UDR0 is actually two registers!
  - One for reading (the receive buffer)
  - One for writing (the transmit buffer)
- Every read of the receive buffer could get a newly received character



Interrupts

- Vector names for interrupts
  - USART0_RX_vect (Receive complete)
  - USART0_TX_vect (Transmit complete)
  - USART0_UDRE_vect (Data register empty)

```
// The serial receive interrupt
ISR (USART0_RX_vect) {
      char received_byte = UDR0;
}
```

Lab 5

- Part I. Receive and Display Text
  - Check frame format and bard rate
  - Optional: Use interrupt
- Part II. Provide Character Echo
  - Send back received characters
- Part III. Push Button Response
  - Send back special messages when a push button is pressed
- Part IV. Bluetooth (57600 baud)
  - Perform USART communication on top of Bluetooth
  - Use a different baud rate

BAM

- The datasheet on your BAM specifies the following USART settings (BAM = Bluetooth Adapter Module)
- See iRobot BAM Datasheet (10542B.pdf)

- "0000" the pairing passcode
- Baud rate: 57,600
- Parity: None
- Data Bits: 8
- Stop Bits: 1
- Hardware Flow Control: None
- Software Flow Control: None