Microprocessors & Interfacing

Serial Input/Output

Lecturer : Annie Guo

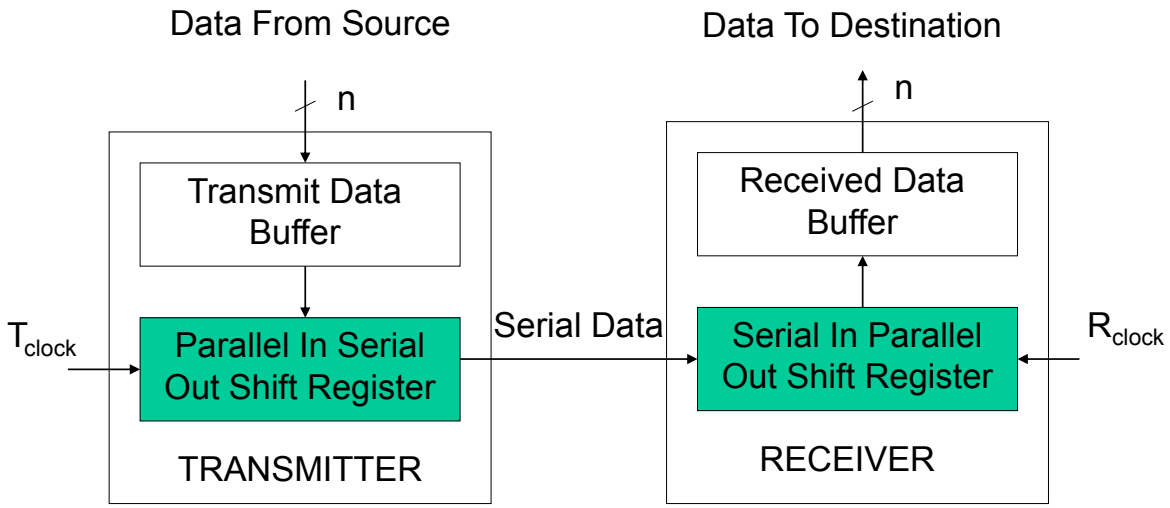
Lecture Overview

- Serial Communication
  - Concepts
  - Standards
- USART in AVR

Why Serial I/O?

- Problems with Parallel I/O:
  - Needs one wire for each bit.
  - When the source and destination are far from each other the parallel cable can be bulky and expensive.
  - Susceptible to reflections and induced noises for long distance communication.
- Serial I/O overcomes these problems.

Serial Communication System Structure



Serial Communication System Structure (cont.)

- At the communication source:
  - The parallel interface transfers data to the transmit data buffer.
  - The data is loaded into the Parallel In Serial Out (PISO) register and T_{clock} shifts the data bits out from the register to the receiver.

Serial Communication System Structure (cont.)

- At the communication destination:
  - R_{clock} shifts each bit received into the Serial In Parallel Out (SIPO) register.
  - After all data bits have been shifted in, they are transferred to the received data buffer.
  - The data in the received data buffer can be read by an input operation via the parallel interface.

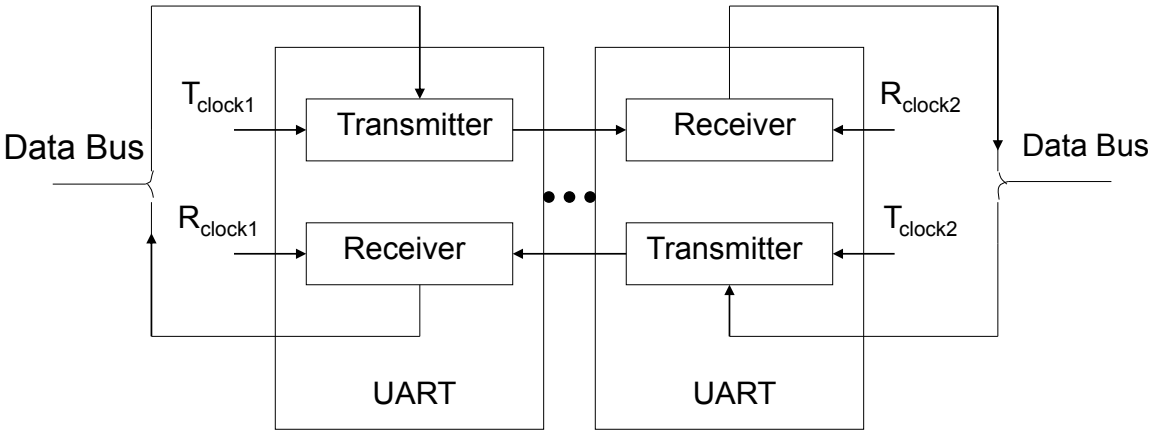
Synchronous VS Asynchronous

- Synchronous
  - Transmitter and receiver clocks are synchronized
    - Need extra hardware for clock synchronization
  - Having faster data transfer rate
- Asynchronous
  - Transmitter and receiver use different clocks. No clock synchronization is required.
  - Used in many applications, such as keyboard, mouse and modem.
  - The rest of this lecture mainly focuses on Asynchronous communication

UART

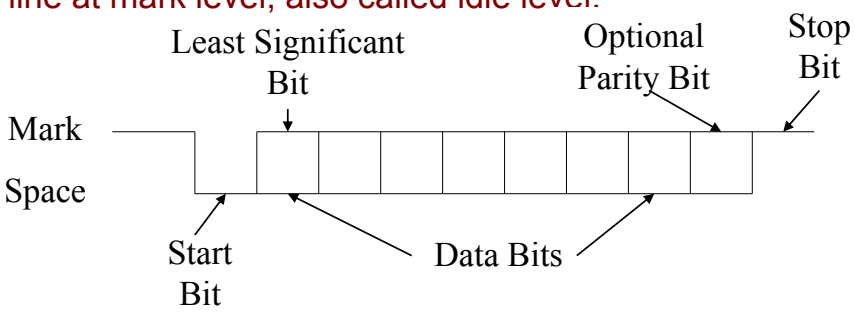
- The device that implements both transmitter and receiver in a single integrated circuit is called UART (Universal Asynchronous Receiver/Transmitter).
- UART uses the least-significant-bit-first order
  - The least significant bit of the data is transferred first
- Data are transmitted asynchronously
  - Clocks on both sides are not synchronized
  - But the receiver has a way to synchronise the data receiving operation with the data transmission operation
- UART is the basis for most serial communication hardware.

Communication Logic Structure with UART



UART Data Frame Format

- Before transmission, data should be encoded
  - Many encoding schemes, such as ASCII
- Each encoded data item is encapsulated with two types of bits
  - Start bit and stop bit
- ***Mark and space***: the logic one and zero levels are called mark and space.
  - When the transmitter is not sending anything, it holds the line at mark level, also called idle level.



UART Data Frame Format (cont.)

- Typical bits in data transmission:
  - Start bit: When the transmitter has data to send, it first changes the line from the mark to the space level for one bit time. This is to synchronise the receiver with the transmitter. When the receiver detects the start bit, it starts to clock in the serial data bits.
  - Data bits: representing a data item, such as a character
  - Parity bit: used to detect errors in the data
    - For odd parity: this bit added to the data to make the total number of 1s in the data odd
    - For even parity: this bit added to the data to make the total number of 1s in the data even.
  - Stop bit: added at the end of data frame. It gives one bit-time between successive data transmissions. Some systems require more than one stop bit.

Data Transmission Rate

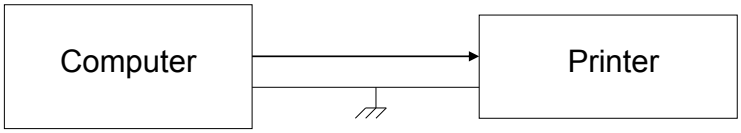
- The rate at which bits are transmitted, also called ***baud rate***, measured in *bits per second.*

Communication Connection Types

- Three serial communication connection types:
  - Simplex
  - Full-duplex (FDX)
  - Half-duplex (HDX)

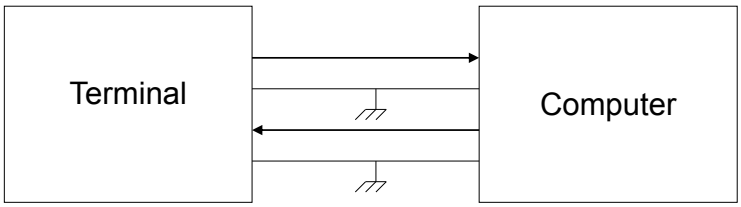
Simplex Connection

- Data are sent in one direction only
  - For example, computer to a serial printer.
- Simple
  - If the sender does not send data faster than the receiver can accept it, no handshaking signals are required.



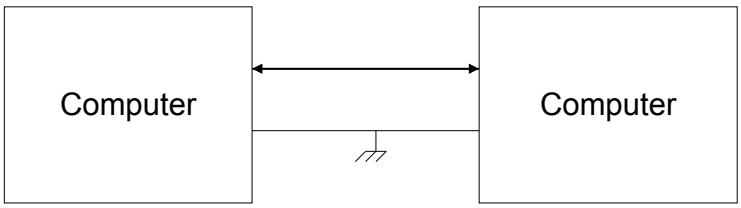
Full-Duplex (FDX) Connection

- Data are transmitted in two directions, each with a separate data line.



Half-Duplex (HDX) Connection

- Data are transmitted in two directions with only one data line.



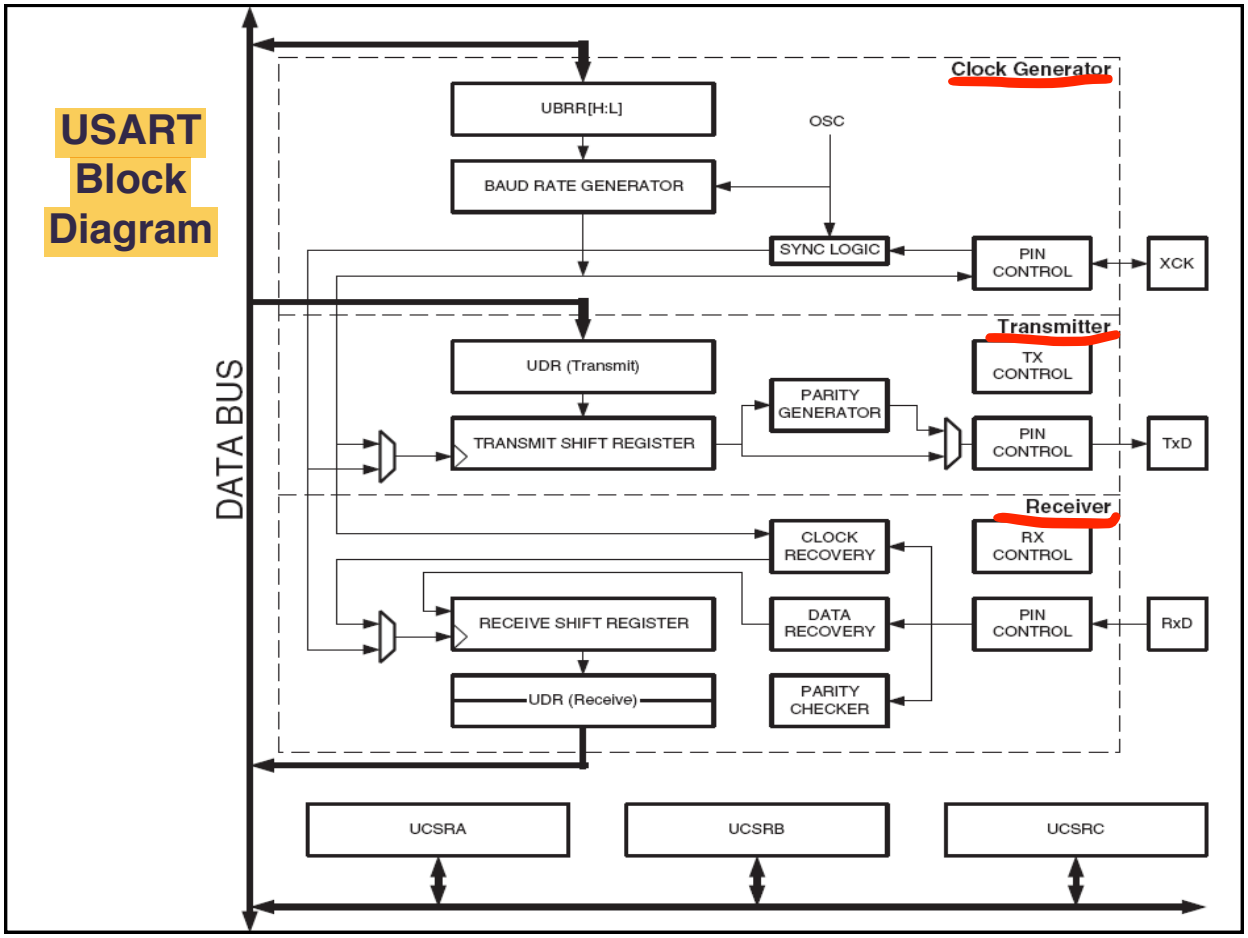
AVR USARTs

- USART: Universal Synchronous Asynchronous Receiver and Transmitter
- Four USART units
  - Units 0-3
- Each unit can be configured for synchronous or asynchronous serial communication

AVR USARTs (cont.)

- Support many frames
- Have transmission error detection function
  - Odd or even parity error
  - Framing error
  - Overrun error
- Three interrupts on
  - TX (Transmit) Complete
  - RX (Receive) Complete
  - TX Data Register Empty

USART Block Diagram



AVR USART Structure

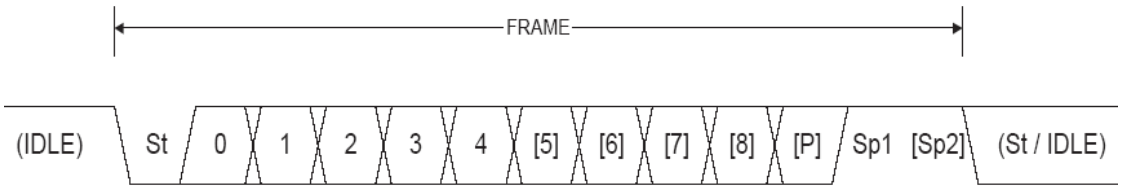
- The USART has three components: clock generator, transmitter and receiver
- Clock generator
  - consists of synchronization logic for external clock input and the baud rate generator
- Transmitter
  - consists of a single write buffer, a serial Shift Register, Parity Generator and Control Logic for handling different frames.

AVR USART Structure (cont.)

- Receiver
  - The receiver is the most complex part of the USART module due to its clock and data recovery units.
  - In addition to the recovery units, the Receiver includes a Parity Checker, Control Logic, a Shift Register and a Receive Buffer (UDR).
  - The Receiver supports the same frame formats as the Transmitter, and can detect Frame Error, Data Over Run and Parity Error.

Frame Format

- Up to 30 different formats available in the USART
  - combinations of
    - 1 start bit (St)
    - 5, 6, 7, 8, or 9 data bits
    - no, even or odd parity bit (P)
    - 1 or 2 stop bits (Sp)



Parity Bit

- Used to check whether the received data is different from the sending data
- Two forms of the parity bit
  - Even parity

$$P_{even} = d_n \oplus d_{n-1} \oplus \cdots \oplus d_1 \oplus d_0 \oplus 0$$

  - Odd parity

$$P_{odd} = d_n \oplus d_{n-1} \oplus \cdots \oplus d_1 \oplus d_0 \oplus 1$$

  - Where d_i in the above two formulas is a data bit, n is the number of data bits.

Control State Registers

- Three control state registers are used in USART operation:
  - UCSRA
    - for storing the status flags of USART
    - for controlling transmission speed and use of multiple processors
  - UCSRB
    - for enabling interrupts, transmission operations
    - for setting frame format
    - for bit extension
  - UCSRC
    - For operation configuration

UCSRA

- USART Control and Status Register A

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

UCSRA Bit Description

- Bit 7 – RXC: USART Receive Complete
  - Set when the receive buffer is not empty
  - The RXC flag can be used to generate a Receive Complete interrupt
- Bit 6 – TXC: USART Transmit Complete
  - Set when the entire frame in the Transmit Shift Register has been shifted out and there are no new data present in the transmit buffer
  - TXC can generate a Transmit Complete interrupt
  - TXC is automatically cleared when a transmit complete interrupt is executed.

UCSRA Bit Description (cont.)

- Bit 5 – UDRE: USART Data Register Empty
  - Set when the transmit buffer (UDR) is empty
  - Can be used to generate a Data Register Empty interrupt
- Bit 4 – FE: Frame Error
  - Set when the character in the receive buffer was transferred in a wrong frame.
- Bit 3 – DOR: Data OverRun
  - Set when a Data OverRun condition is detected.
  - A Data OverRun occurs when the receive buffer is full and a new start bit is detected.

UCSRA Bit Description (cont.)

- Bit 2 – UPE: USART Parity Error
  - Set when the character in the receive buffer had a Parity Error when received and the Parity Checking was enabled
- Bit 1 – U2X: Double the USART Transmission Speed
  - Set for doubling the transfer rate for asynchronous communication
- Bit 0 – MPCM: Multi-processor Communication Mode
  - If set, all the incoming frames received by the USART Receiver that do not contain address information will be ignored.

UCSRB

- USART Control and Status Register B

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

UCSRB Bit Description

- Bit 7 – RXCIE: RX Complete Interrupt Enable
  - Set to enable interrupt on the RXC flag
- Bit 6 – TXCIE: TX Complete Interrupt Enable
  - Set to enable interrupt on the TXC flag
- Bit 5 – UDRIE: USART Data Register Empty Interrupt Enable
  - Set to enable interrupt on the UDRE flag.

UCSRB Bit Description (cont.)

- Bit 4 – RXEN: Receiver Enable
  - Set to enable 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 UPE flags.
- Bit 3 – TXEN: Transmitter Enable
  - Set to enable the USART Transmitter
  - The Transmitter will override normal port operations for the TxD pin when enabled. Disabling the Transmitter will not become effective until transmissions are complete.

UCSRB Bit Description (cont.)

- Bit 2 – UCSZ2: Character Size
  - The bit combined with the UCSZ1:0 bits in UCSRC sets the number of data bits in a frame.
- Bit 1 – RXB8: Receive Data Bit 8
  - The ninth data bit of the received character when operating with serial frames with 9-bit data. Must be read before reading the low bits from UDR
- Bit 0 – TXB8: Transmit Data Bit 8
  - The ninth data bit in the character to be transmitted when operating with serial frames with 9-bit data. Must be written before writing the low bits to UDR

UCSRC

- USART Control and Status Register C

Bit	7	6	5	4	3	2	1	0
	–	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL
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

UCSRC Bit Description

- Bit 6 – UMSEL: USART Mode Select
  - 0: Asynchronous Operation
  - 1: Synchronous Operation
- Bit 5:4 – UPM1:0: Parity Mode
  - Set to enable Parity bit operation

UPM1	UPM0	Parity Mode
0	0	Disabled
0	1	Reserved
1	0	Enabled, Even Parity
1	1	Enabled, Odd Parity

UCSRC Bit Description (cont.)

- Bit 3 – USBS: Stop Bit Select
  - 0: 1-bit
  - 1: 2-bit
- Bit 2:1 – UCSZ1:0: Character Size
  - Together with UCSZ2 to determine the number of bits for a character

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

UCSRC Bit Description (cont.)

- Bit 0 – UCPOL: Clock Polarity

UCPOL Sampled	Transmitted Data Changed (Output of TxD Pin)	Received Data (Input on RxD Pin)
0	Rising XCK Edge	Failing XCK Edge
1	Failing XCK Edge	Rising XCK Edge

USART Initialization

- Initialization process consists of
  - Setting the baud rate,
  - Setting the frame format; and
  - Enabling the Transmitter or the Receiver

- For interrupt driven USART operations, the Global Interrupt Flag should be cleared when doing the initialization

Sample Code

- Initialize USART 1

```
USART_Init:
        ; Set baud rate, which is stored in r17:r16
        sts UBRR1H, r17
        sts UBRR1L, r16

        ; Enable receiver and transmitter
        ldi r16, (1<<RXEN1)|(1<<TXEN1)
        sts UCSR1B,r16

        ; Set frame format: 8 bit data, 2 stop bits
        ldi r16, (1<<USBS1)|(3<<UCSZ10)
        sts UCSR1C,r16
```

Data Transmission

- The USART Transmitter is enabled by setting the *Transmit Enable* (TXEN) bit in the UCSRB Register.
  - A data transmission is initiated by loading the transmit buffer with the data to be transmitted.
    - The CPU can load the transmit buffer by writing to the UDR I/O location. The buffered data in the transmit buffer will be moved to the Shift Register when the Shift Register is ready to send a new frame.
      - The Shift Register is loaded with new data if it is in idle state (no ongoing transmission) or immediately after the last stop bit of the previous frame is transmitted. When the Shift Register is loaded with new data, it will transfer one complete frame at the rate given by the baud register, U2X bit or by XCK depending on mode of operation.

Sample code

- Data Transmission
  - The code below uses polling of the *Data Register Empty* (UDRE) flag.
    - When using frames with less than eight bits, the most significant bits written to the UDR are ignored.

```
USART_Transmit:

        ; Wait for empty transmit buffer
        lds r15, UCSR1A
        sbrs r15,UDRE1
        rjmp USART_Transmit

        ; Put data (r16) into buffer, sends the data
        sts UDR1,r16
```

Status of Data Transmission

- The USART Transmitter has two flags that indicate its state:
  - USART Data Register Empty (UDRE)
    - Set: when the transmit buffer is empty and ready to receiver new data
    - Clear: when the transmit butter contains data to be moved into the shift register
  - Transmit Complete (TXC)
    - Set: when data in the Transmit Shift Register has been shifted out and there are no new data currently present in the transmit buffer
    - Clear: otherwise
- Both flags can be used for generating interrupts.

Data Reception

- The USART Receiver is enabled by writing the Receive Enable (RXEN) bit in the UCSRB Register
  - The baud rate, mode of operation and frame format must be set up before doing any transmissions. If synchronous operation is used, the clock on the XCK pin will be overridden and used as transmission clock.

Sample code

- Data reception

```
USART_Receive:

        ; Wait for data to be received
        lds     r10, UCSR1A
        sbrs    r10, RXC1

        rjmp    USART_Receive

        ; Get and return received data from buffer
        lds     r16, UDR1
```

Status of Data Reception

- The Receive Complete (RXC) flag indicates if there are unread data present in the receive buffer.
  - Set: when the unread data exists in the receive buffer
  - Clear: otherwise
- If the receiver is disabled (RXEN = 0), the receive buffer will be flushed and consequently the RXC bit will become zero.

Error Detection

- Three errors are checked on the Receiver side:
  - Frame error
    - By checking whether the first stop bit is correctly received
  - Parity error
    - By checking whether the data received has the same (odd or even) number of 1's as in the data from the transmitter.
  - Data OverRun error
    - By checking if any data is yet read but is overwritten by incoming data frame.

Error Recovery

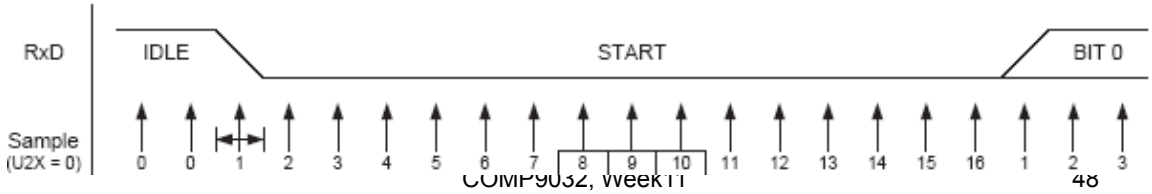
- Main sources of errors in asynchronous data transmission
  - Data reception is “out of sync” with transmission
  - Noise added to the data

Error Recovery (cont.)

- AVR includes a clock recovery and a data recovery unit for handling errors in asynchronous transmission.
  - The clock recovery logic is used for synchronizing the internally generated baud rate clock to the incoming asynchronous serial frames at the RxD pin.
    - Based on the start bit
  - The data recovery logic samples and (low-pass) filters each incoming bit, thereby improving the noise immunity of the Receiver
  - Based on multiple sampling and majority policy for each incoming bit

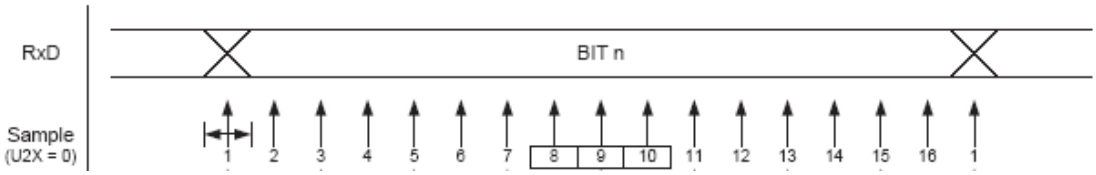
Error Recovery (cont.)

- The following figure gives an illustration
- The sample rate is 16 times the baud rate
  - When the Clock Recovery logic detects a high (idle) to low (start) transition on the RxD (receiver data) line, it uses samples 8, 9 and 10 to decide if it is a valid bit (namely, 0)
    - If the majority of the three bits are 0, the bit is valid; data recovery is followed.
    - If the majority of the three bits are 1, the bit is invalid; the circuit looks for next start bit.



Error Recovery (cont.)

- When the receiver clock is synchronized to the start bit, the data recovery can begin for each subsequent data bit.
- Similarly, the decision of the logic level of each received bit is taken by doing a majority voting of the logic value to the three samples in the center of the received bit.



Reading Material

- Chapter 12: Serial Input/Output. Microcontrollers and Microcomputers by Fredrick M. Cady.
- Mega2560 Data Sheet
  - USART