

HANDOUT #7

010113027

MICROPROCESSORS & EMBEDDED COMPUTER SYSTEMS

INSTRUCTOR: RSP (rawat.s@eng.kmutnb.ac.th)

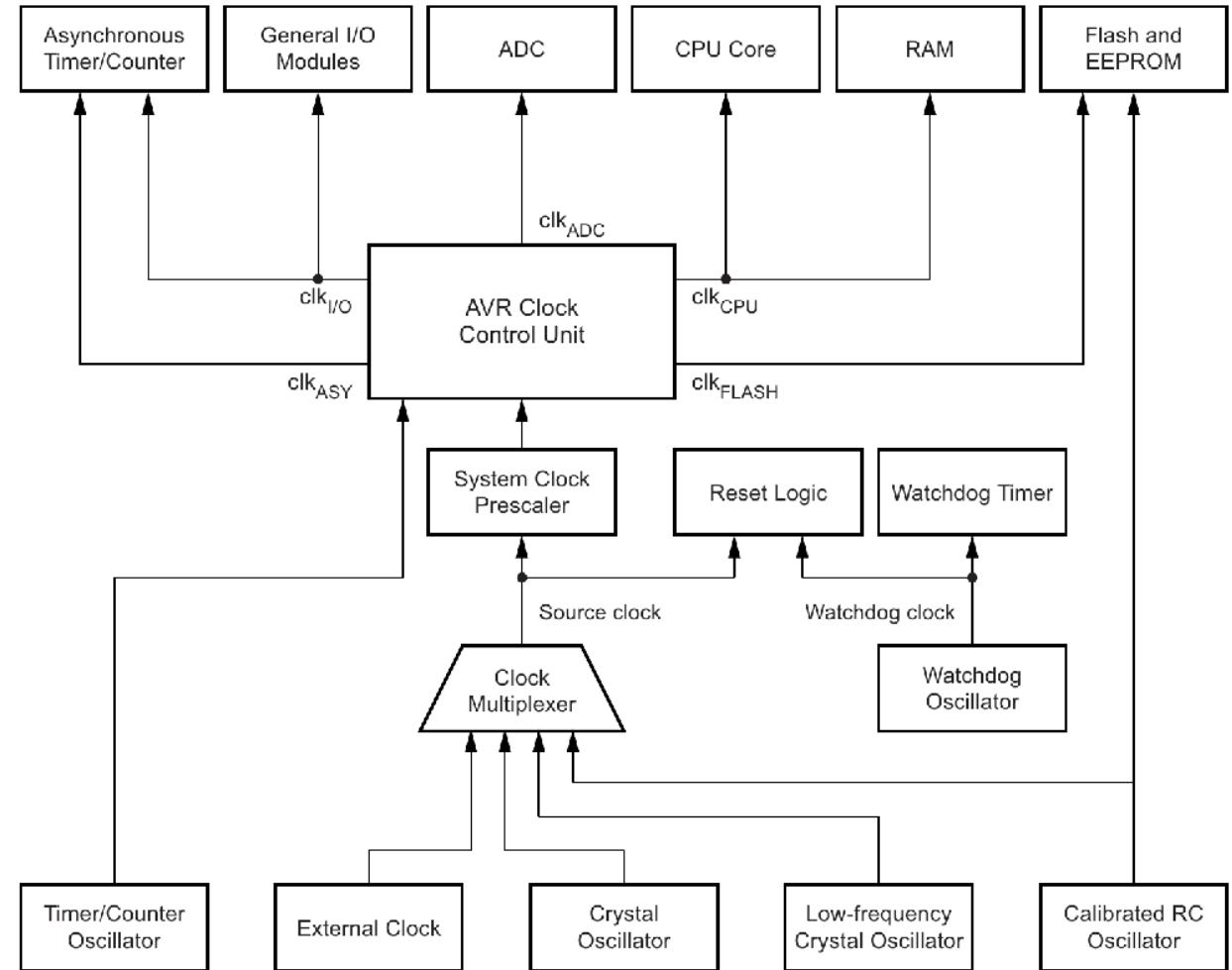
What We Will Learn...

- AVR Clock Sources
- AVR Low-power / Sleep Modes
- AVR Wakeup Sources
- C Code Demo for Sleep / Wakeup Operations
- USART Peripheral for Asynchronous Serial Data Communication
- USART Configuration
- USART Interrupt Handling
- Using `printf()` with USART
- C++ Class Implementation for Simple Serial Demo

AVR Clock Sources

- This block diagram shows the **overall clock system** in the AVR and the **distribution of clock signals**.
- Not all clocks need to be active at the same time.
- To reduce power consumption, **clocks to unused modules** can be halted by selecting appropriate **sleep modes**.

Clock Distribution



Default clock source: the 8MHz internal RC oscillator with the fuse CKDIV8 programmed, resulting in 1.0MHz system clock.

AVR Clock Sources

- **CPU Clock:**
 - It drives the AVR core (register file, status register, stack pointer).
 - Halting it stops CPU execution and calculations.
- **I/O Clock:**
 - It is used by peripherals such as Timer/Counters, SPI, and USART.
 - Some interrupts and TWI address detection work asynchronously even when clkI/O is stopped.
- **Flash Clock:**
 - It controls Flash memory access and is typically active together with the CPU clock.
- **Asynchronous Timer Clock:**
 - It drives Timer/Counter2 from an external clock or 32 kHz crystal.
 - It can be enabled for real-time counting during sleep modes.
- **ADC Clock:**
 - It is a dedicated clock for ADC operation and allows CPU and I/O clocks to be halted to reduce digital noise and improve ADC accuracy.

Low-Power / Sleep Modes in the AVR

- The AVR MCU such as ATmega328P provides **different sleep modes** designed to **reduce power consumption by shutting down parts of the chip** that are not needed while preserving the contents of RAM and registers.
- The MCU can switch from **Active Mode** to a **power-saving sleep mode**.
- There are **six sleep modes**. Each mode provides a different level of power reduction.
 - **Idle, ADC Noise Reduction, Power-down, Power-save, Standby, and Extended Standby.**
- In all sleep modes, the CPU clock is stopped, preventing instruction execution, while the contents of registers and SRAM are preserved.
- The operation of peripheral modules depends on the selected sleep mode.
- The power-saving level depends on which clock sources and hardware modules are disabled.

Why Are There Different Sleep Modes?

- Different applications have varying requirements for power saving versus functionality.
- Some situations require retaining peripheral activity (e.g., a timer), while others can shut down almost everything for maximum battery life (for a battery operation).
- The AVR MCU provides multiple modes so designers can balance power consumption and required operation.

AVR Sleep Modes

Active Clock Domains and Wake-up Sources in the Different Sleep Modes.

Sleep Mode	Active Clock Domains					Oscillators		Wake-up Sources						Software BOD Disable	
	clk _{CPU}	clk _{FLASH}	clk _{IO}	clk _{ADC}	clk _{ASY}	Main Clock Source Enabled	Timer Oscillator Enabled	INT1, INT0 and Pin Change	TWI Address Match	Timer2	SPM/EEPROM Ready	ADC	WDT	Other/O	
Idle		X	X	X	X	X	X ⁽²⁾	X	X	X	X	X	X	X	
ADC noise Reduction			X	X	X	X	X ⁽²⁾	X ⁽³⁾	X	X ⁽²⁾	X	X	X		
Power-down								X ⁽³⁾	X				X	X	
Power-save				X		X	X ⁽²⁾	X ⁽³⁾	X	X			X	X	
Standby ⁽¹⁾						X		X ⁽³⁾	X				X	X	
Extended Standby				X ⁽²⁾	X	X ⁽²⁾		X ⁽³⁾	X	X			X	X	

- Notes:
1. Only recommended with external crystal or resonator selected as clock source.
 2. If Timer/Counter2 is running in asynchronous mode.
 3. For INT1 and INT0, only level interrupt.

Idle Mode:

- This mode preserves most peripherals and only halts the CPU
 - useful if still keeping timers or serial interfaces running.

Power-down Mode: Current consumption < 1uA

- It disables almost all clocks and peripherals and is the most power-efficient
 - but only certain wake-up sources remain enabled.
 - Clock stopped: CPU, Flash, all synchronous peripherals, main oscillator
 - SRAM and register contents preserved
- **Wake-up sources:**
 - External interrupts: INT0 / INT1, PCINT
 - WDT (periodic wakeup)
 - TWI (I2C address match in slave mode)
 - External reset pin

Power-save Mode

- This mode is identical to Power-down, with one exception: it allows a timer to run asynchronously even while the rest of the system sleeps — useful when a periodic wake-up is needed.
- **Timer/Counter2** is still running.
 - The device can wake up from Timer2 interrupt.
 - Useful for periodic wakeup with Timer2.
 - But only in **asynchronous mode** with external 32.768 kHz crystal on TOSC1/TOSC2 pins.
 - If **Timer/Counter2** is not running, power-down mode is recommended instead of power-save mode.

Registers Related To Sleep Modes

Sleep Mode Control Register (SMCR)

Bit	7	6	5	4	3	2	1	0
0x33 (0x53)	-	-	-	-	SM2	SM1	SM0	SE
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

The sleep mode control register contains control bits for power management.

SM2	SM1	SM0	Sleep Mode
0	0	0	Idle
0	0	1	ADC Noise Reduction
0	1	0	Power-down
0	1	1	Power-save
1	0	0	Reserved
1	0	1	Reserved
1	1	0	Standby ⁽¹⁾
1	1	1	External Standby ⁽¹⁾

Note: 1. Standby mode is only recommended for use with external crystals or resonators.

Registers Related To Sleep Modes

PRR – Power Reduction Register

Bit	7	6	5	4	3	2	1	0
(0x64)	PRTWI	PRTIM2	PRTIM0	-	PRTIM1	PRSPI	PRUSART0	PRADC
Read/Write	R/W	R/W	R/W	R	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

This register can be programmed to disable internal peripheral modules (using clock gating / shutdown) :

- **PRTWI** bit: Power Reduction **TWI (I2C) module**
- **PRTIM2** bit: Power Reduction **Timer/Counter2 module**
- **PRTIM0** bit: Power Reduction **Timer/Counter0 module**
- **PRTIM1** bit: Power Reduction **Timer/Counter1 module**
- **PRSPI** bit: Power Reduction **Serial Peripheral Interface (SPI) module**
- **PRUSART0** bit: Power Reduction **USART0 module**
- **PRADC** bit: Power Reduction **ADC module**

For example, if the **PRADC** bit is set to 1, the ADC is disabled, and if the **PRUSART0** bit is set to 1, the **USART0** module is disabled.

```
PRR |= (1<<PRADC); // disable ADC  
PRR |= (1<<PRUSART0); // disable USART0  
PRR |= (1<<PRTIM1); // disable Timer1
```

Port Pins in Sleep Modes

- Before entering a sleep mode, all I/O port pins should be configured to minimize power consumption. Ensure that no pins source or sink current by driving resistive loads.
- For pins used as analog inputs, the **digital input buffer should be disabled** to reduce power consumption and minimize noise during analog operation.
- **Digital input buffers** on unused or analog pins can be disabled by setting the corresponding bits in the **Digital Input Disable Registers (DIDR0 and DIDR1)**.

On AVR, the **DIDR registers** control the digital input buffer on certain pins.

- **DIDR0**: ADC pins (ADC0–ADC5)
- **DIDR1**: Analog Comparator pins (AIN0, AIN1)

Steps to Enter a Sleep Mode

- Disable unused peripheral modules. Peripheral shutdown is controlled by the **Power Reduction Register (PRR)**.
 - In power-down mode, almost everything is automatically stopped by hardware.
- Select the desired sleep mode by setting the **SM2:0 bits** in **SMCR** to one of the six mode codes (Idle, ADC Noise Reduction, Power-down, etc.).
- Set the **Sleep Enable (SE) bit** in **SMCR** to 1 to enable the execution of the **SLEEP** instruction.
- Execute the **SLEEP** instruction. This causes the MCU to enter the selected sleep mode.

How To Wake Up MCU from Sleep

- When the ATmega328P is in a sleep mode, enabled interrupt sources or reset events will wake the MCU.
- Possible wake-up sources which depend on which sleep mode is used:
 - External interrupts (e.g., INT0/INT1, pin change interrupts)
 - Watchdog Timer (WDT) interrupt or reset
 - Timer overflow or compare match (depending on mode)
 - Two-Wire Serial Interface (TWI or I2C) address match
 - ADC conversion complete (in certain modes)
 - External reset

How To Wake Up MCU from Sleep

When the ATmega328P is in a sleep mode, enabled interrupt sources or reset events can wake the MCU.

Wake-up by interrupt

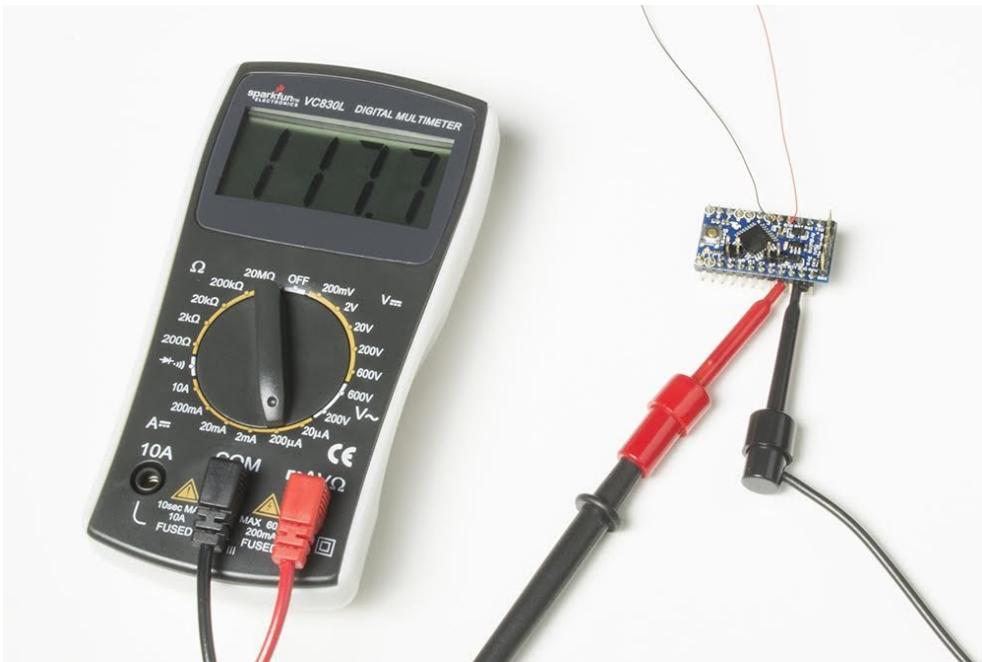
- CPU wakes (SRAM, registers, and I/O registers are retained).
- Interrupt Service Routine (ISR) executes.
- Execution then continues after SLEEP.

Wake-up by reset

- Program restarts from the Reset Vector.
- SRAM and registers are not guaranteed to be preserved.

Reducing Arduino Power Consumption

Using the Low Power library for Arduino: <https://github.com/rocketscream/Low-Power/>



Vcc (V)	Clock Speed (MHz)	Wake Current (mA)	Sleep Current (uA)
5.0	16	13.92	6.2
5.0	8	9.03	6.2
3.3	16	6.48	4.3
3.3	8	3.87	4.3

Source: <https://learn.sparkfun.com/tutorials/reducing-arduino-power-consumption/all>

Code Example: Sleep & Wake Up by WDT

The behavior of the example code is as follows (for **ATmega328P**):

- The **WDT** is used and configured to generate an interrupt (without causing a WDT reset) when a **WDT timeout** occurs.
- The **WDT prescaler** is selected to obtain a WDT timeout period of approximately 1 second.
- After the **WDT** is enabled, the **MCU** enters sleep mode by executing the **SLEEP** instruction, and program execution is halted until the MCU is woken up.
- Each time a **WDT** timeout occurs, the **MCU** is woken up from sleep mode and resumes operation. It also turns off the LED.
- The **MCU** also responds to the interrupt by invoking the **ISR for the WDT**, which is programmed to turn on the **LED**.
 - The ISR definition for a **WDT interrupt** using avr-gcc is as follows:
`ISR(WDT_vect) { ... }`

Code Example: Sleep & Wake Up by WDT

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>

#define nop()    __asm__ __volatile__("nop")
#define wdr()    __asm__ __volatile__("wdr")
#define sleep()   __asm__ __volatile__("sleep")

#define LED_DDR    DDRB
#define LED_PORT   PORTB
#define LED_BIT    PB5

volatile uint8_t wakeup_count = 0;

void WDT_stop(void) {
    cli();                      // Disable global interrupts
    wdr();                      // Reset WDT
    MCUSR &= ~(1 << WDRF); // Clear WDT reset flag
    // Enable configuration changes
    WDTCSR |= (1 << WDCE) | (1 << WDE);
    WDTCSR = 0x00;              // Disable WDT
    sei();                      // Enable global interrupts
}
// Code continues on the next page...
```

Code Example: Sleep & Wake Up by WDT

```
void WDT_init(void) {
    cli();                      // Disable global interrupts
    MCUSR &= ~(1 << WDRF);   // Clear WDT reset flag
    // Enable configuration changes
    WDTCSR |= (1 << WDCE) | (1 << WDE);
    // WDT interrupt mode, ~1 s timeout
    // WDP2 + WDP1 = approx. 1 second
    WDTCSR = (1 << WDIE) | (1 << WDP2) | (1 << WDP1);
    wdr(); // Reset WDT
    sei(); // Enable global interrupts
}

ISR(WDT_vect) {
    wakeup_count++; // Increment wakeup count
    LED_PORT |= (1 << LED_BIT); // Turn on LED
}
```

Code Example: Sleep & Wake Up by WDT

```
int main(void) {
    // Configure LED pins as outputs
    LED_DDR |= (1 << LED_BIT);
    LED_PORT &= ~(1 << LED_BIT);
    _delay_ms(1000);
    WDT_init(); // Initialize Watchdog
    while (1) {
        // Enter power-down sleep mode
        SMCR = (1 << SM1) | (1 << SE);
        sleep(); // Enter sleep mode
        nop();

        _delay_ms(25);

        LED_PORT &= ~(1 << LED_BIT); // Turn off LED

        if (wakeup_count >= 10) {
            LED_PORT |= (1 << LED_BIT); // Turn on LED
            WDT_stop(); // Stop WDT
            while(1);
        }
    }
}
```

Code Example: Sleep & Wake Up by INT0

The behavior of the example code is as follows (for ATmega328P)

- **External Interrupt Request 0 (INT0)**, which corresponds to pin **PD2 (Arduino Uno / Nano D3 pin)**, is used and connected to an active-low push-button circuit.
- The interrupt trigger condition for **INT0** is configured as **low-level**, consistent with the active-low button connection.
- The **MCU** enters sleep mode, during which the LED remains off.
- The **MCU** is woken up by the **INT0 interrupt** and continues execution, blinking the **LED** for 3 times, before returning to sleep mode.
- The associated **ISR** clears the **INT0 enable bit** in the **EIMSK** register to prevent the interrupt from triggering again before the **MCU** enters sleep mode the next time.

Code Example: Sleep & Wake Up by INT0

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>
#include <util/delay.h>

#define nop()    __asm__ __volatile__("nop")

volatile uint8_t woke_up = 0;

ISR(INT0_vect) { // INT0 ISR
    EIMSK &= ~(1 << INT0);    // Disable INT0 immediately
    woke_up = 1;                // Set flag
}

static void peripherals_disable(void) {
    // Disable ADC
    ADCSRA &= ~(1 << ADEN);
    // Disable Analog Comparator
    ACSR |= (1 << ACD);
    // Disable digital input buffers on ADC pins
    DIDR0 = 0x3F;
}
// Code continues on the next page...
```

Code Example: Sleep & Wake Up by INT0

```
static void gpio_init(void) {
    DDRB = (1 << DDB5);
    PORTB = 0x00; // Outputs low for all PORTB pins
    // PC0..PC5 unused: outputs low
    DDRC = 0x3F;
    PORTC = 0x00;
    // PD2/INT0 input with pull-up; output low for other pins
    DDRD = 0xFF;
    DDRD &= ~(1 << DDD2);
    PORTD = (1 << PORTD2); // Enable pull-up on INT0
}

static void enable_int0(void) {
    // Low-level trigger on INT0
    EICRA &= ~(1 << ISC00);
    EICRA &= ~(1 << ISC01);
    EIFR |= (1 << INTF0); // Clear flag
    EIMSK |= (1 << INT0); // Enable INT0
}
// Code continues on the next page...
```

Code Example: Sleep & Wake Up by INT0

```
static void toggle_led(uint8_t times) {
    for (uint8_t i = 0; i < times; i++) {
        PINB = (1 << PB5);
        _delay_ms(50);
    }
}

static void enter_sleep(void) {
    set_sleep_mode(SLEEP_MODE_PWR_DOWN);
    sleep_enable();
    sleep_bod_disable(); // Disable BOD
    sei();
    sleep_cpu();
    sleep_disable();
}
```

```
int main(void) {
    cli();
    gpio_init();
    peripherals_disable();
    enable_int0();
    sei();

    while (1) {
        woke_up = 0;
        enter_sleep();
        while (!(PIND & (1 << PIND2))) {
            _delay_ms(5);
        }
        enable_int0();
        if (woke_up) {
            toggle_led(6);
        }
    }
}
```

Code Example: Sleep & Wake Up by PCINT2

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>
#include <util/delay.h>

volatile uint8_t woke_up = 0;

ISR(PCINT2_vect) {
    if (!(PIND & (1 << PD2))) { // Is PD2 low?
        woke_up = 1; // Set the flag
    }
}

static void peripherals_disable(void) {
    ADCSRA &= ~(1 << ADEN); // ADC off
    ACSR |= (1 << ACD); // Analog comparator off
    DIDR0 = 0x3F; // Disable ADC digital inputs
}

// Code continues on the next page...
```

Code Example: Sleep & Wake Up by PCINT2

```
static void gpio_init(void) {
    // PB5 (LED) output low
    DDRB = (1 << DDB5);
    PORTB = 0x00;
    // PC0..PC5 unused: outputs low
    DDRC = 0x3F;
    PORTC = 0x00;
    // PD2 input with pull-up, others output low
    DDRD = 0xFB; // 0b1111_1011 (only PD2 is input)
    PORTD = (1 << PORTD2); // pull-up on PD2
}

static void pcint_init(void) {
    PCICR |= (1 << PCIE2); // Enable PCINT[23:16]
    PCMSK2 |= (1 << PCINT18); // Enable PCINT18 (PD2)
    PCIFR |= (1 << PCIF2); // Clear pending flag
}

// Code continues on the next page...
```

Code Example: Sleep & Wake Up by PCINT2

```
static void toggle_led(uint8_t times) {  
    while (times--) {  
        PINB = (1 << PB5);  
        _delay_ms(50);  
    }  
}  
  
static void enter_sleep(void) {  
    set_sleep_mode(SLEEP_MODE_PWR_DOWN);  
    sleep_enable();  
    sleep_bod_disable(); // Disable BOD  
    sei();  
    sleep_cpu();  
    sleep_disable();  
}
```

```
int main(void) {  
    cli();  
    gpi0_init();  
    peripherals_disable();  
    pcint_init();  
    sei();  
  
    while (1) {  
        woke_up = 0;  
        enter_sleep(); // Enter sleep mode  
        if (woke_up) {  
            toggle_led(6); // Blink the LED  
        }  
    }  
}
```

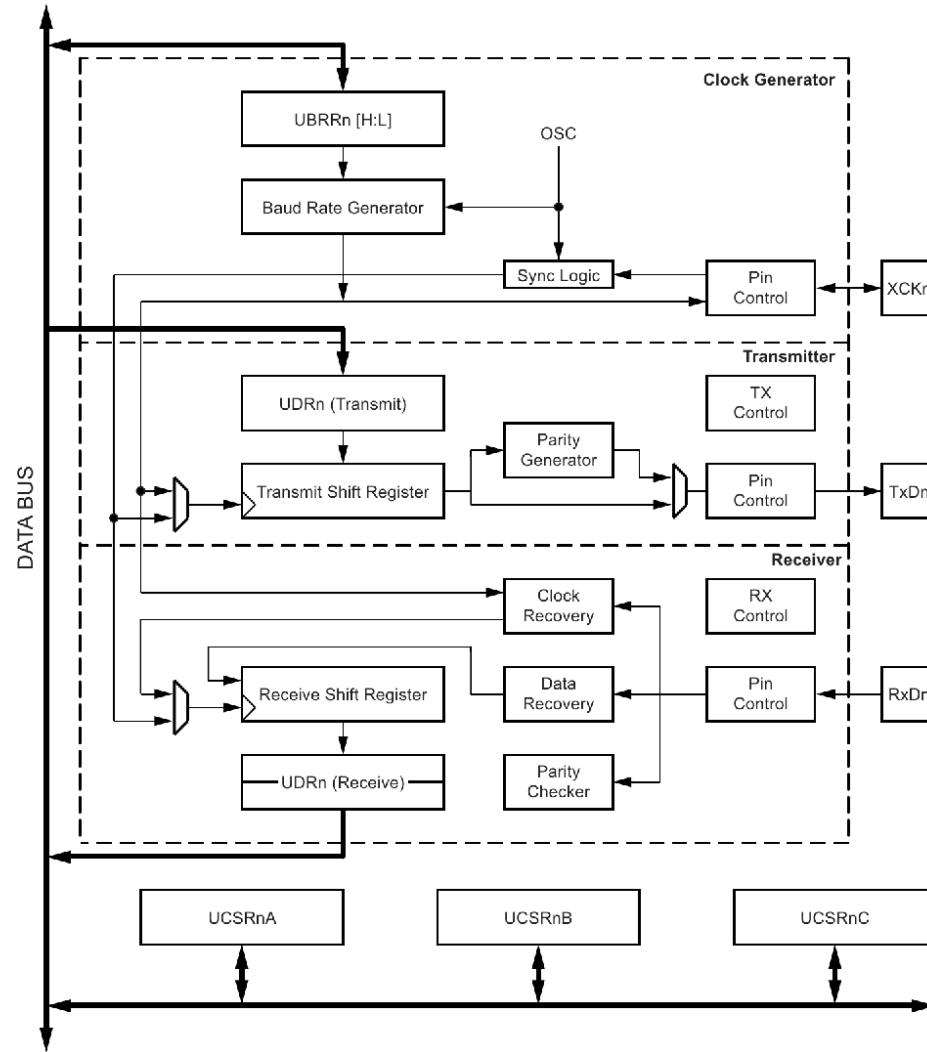
- One common interface is the **USART** (*Universal Synchronous and Asynchronous Receiver and Transmitter*).
- The **ATmega328P** contains **one USART module (USART0)**.
 - Other AVR devices may provide multiple USARTs (for example, **ATmega2560** has four).
- **USART0** features
 - Performs **bit-serial (bit-by-bit) data communication**.
 - Supports both **asynchronous** and **synchronous** operation.
 - Supports **full-duplex** communication (independent **TX** and **RX** lines).

USART Block Diagram

In this simplified block diagram, the following hardware components are shown:

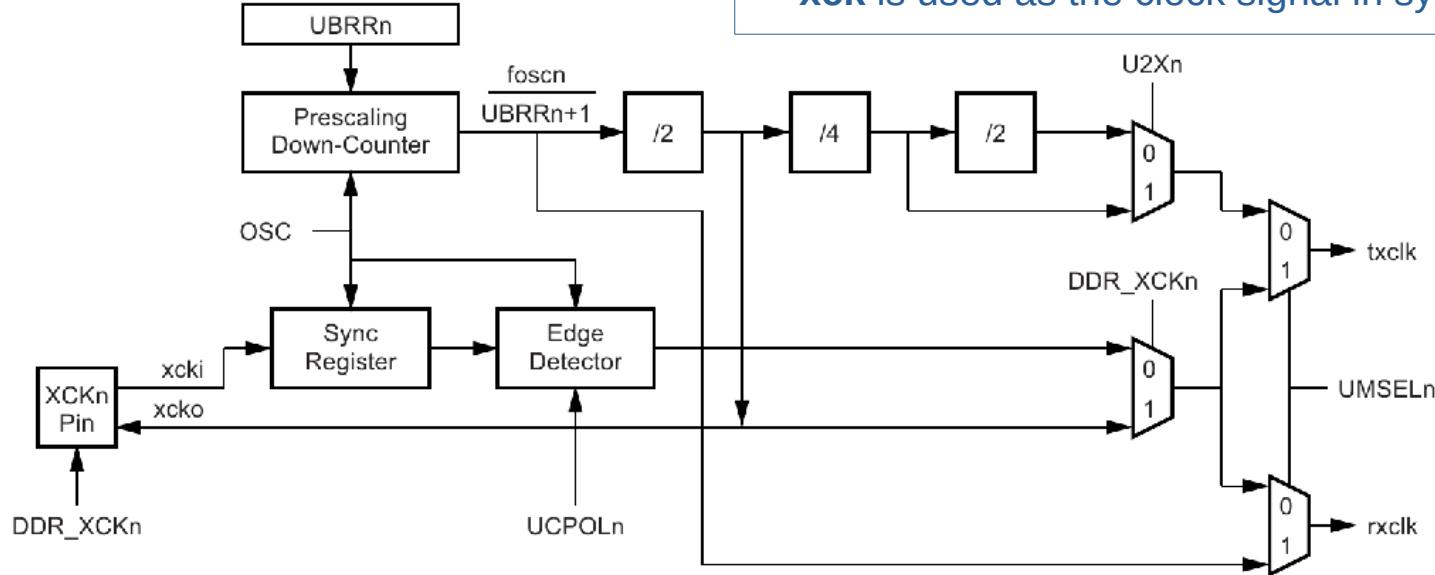
- USART-related registers
- USART interface pins (TXD, RXD, XCK)
- Internal data path

USART Block Diagram¹



USART Block Diagram

Clock Generation Logic, Block Diagram



Note:

- **rxclk** runs at a higher frequency than **txclk**.
- **xck** is used as the clock signal in synchronous mode.

Signal description:

txclk Transmitter clock (Internal Signal).

rxclk Receiver base clock (Internal Signal).

xcki Input from XCK pin (internal Signal). Used for synchronous slave operation.

xcko Clock output to XCK pin (Internal Signal). Used for synchronous master operation.

fosc XTAL pin frequency (System Clock).

Asynchronous vs. Synchronous Modes

Asynchronous mode

- No dedicated clock, only **TXD** and **RXD** lines
- Both devices must match the same baud rate and frame format.
- Full-duplex point-to-point communication
- Uses start/stop bits + data bits

Synchronous mode

- Uses dedicated clock (**XCK**), **TXD** and **RXD** (3-wire)
- Operates in master/slave configuration
 - Master generates the clock.
 - Both devices shift data simultaneously with the clock
 - No start/stop bits

USART0 Peripheral

USART0 supports four operating modes:

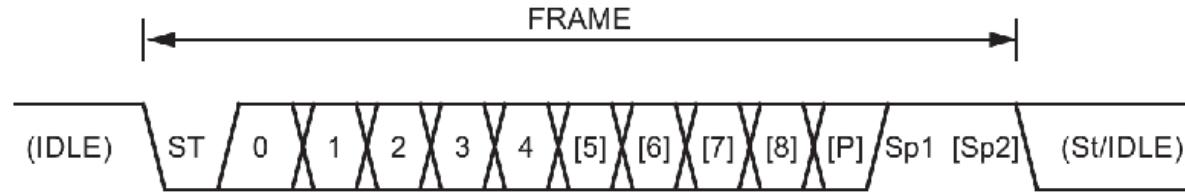
- Asynchronous normal and double (2x) speed
- Synchronous master only: **Master SPI Mode (MSPIM), no Slave-Select (SS) pin**
 - The **ATmega328P** also includes a dedicated **SPI** (*Serial Peripheral Interface*) module.
- In practice, **USART0** is most commonly used for **asynchronous UART-style communication** (e.g., PC serial, USB-to-serial adapters).
- On **Arduino Uno / Nano board**:
 - **Arduino D0 / PD0 pin = RXD** (Receive, digital input)
 - **Arduino D1 / PD1 pin = TXD** (Transmit, digital output)
- These pins are also connected to the on-board **USB-to-Serial bridge chip** for sketch uploading and serial monitoring:
 - **ATmega16U2** on official Uno boards
 - **CH34x or FT232** on many Nano / clones

USART: Asynchronous Mode

- No clock signal is transmitted.
- Both devices must agree on the **baud rate**.
- **Data frame** format consists of:
 - 1 start bit (always logic **0** or **LOW**)
 - 5–9 data bits
 - optional parity bit (even or odd parity)
 - 1 or 2 stop bits (always logic **1** or **HIGH**)
 - **8N1** = 8-bit data, no parity bit, one stop bit
- In the **Idle state** or when no data is transmitted:
 - **TXD = HIGH (logic 1)**
 - **RXD = HIGH (logic 1)**
 - This is called the **MARK state**.

UART Data Frame

Frame Formats



- St** Start bit, always low.
- (n)** Data bits (0 to 8).
- P** Parity bit. Can be odd or even.
- Sp** Stop bit, always high.
- IDLE** No transfers on the communication line (RxDn or TxDn). An IDLE line must be high.

USART: Asynchronous Mode

Steps to use the **USART0** in TX and RX mode.

- Configure **USART baud rate** via the **UBRR0** register (16-bit).
 - **UBRR0** = UART Baud Rate Generator
- Configure the **UCSR0A** register
 - Normal Speed: **U20X0** = 0
 - Double Speed: **U20X0** = 1
- Configure **UCSR0B** and **UCSR0C** registers
 - Set the frame format
 - Choose frame format (e.g. 8N1: 8 data bits, 1 stop bit)
- Enable transmitter and receiver.
 - Enable Tx: **TXEN0** = 1 in **UCSR0B**
 - Enable Rx: **RXEN0** = 1 in **UCSR0B**
- Read or write the **UDR0** register (TX or RX Data Buffer Register)

$$\text{Baudrate (Normal Speed)} = \frac{f_{CPU}}{16 \cdot (UBRR0 + 1)}$$

$$\text{Baudrate (Double Speed)} = \frac{f_{CPU}}{8 \cdot (UBRR0 + 1)}$$

USART: Asynchronous Mode

- **Polling method**
 - **RX operation:**
 - Use the **RXC0 (Receive Complete)** bit in **UCSR0A** to check if a data byte has been received.
 - Read from **UDR0** if **RXC0** is 1.
 - **TX operation:**
 - Wait until transmit register is free, then writes the next data byte.
 - Write to **UDR0** if **UDRE0 (Data Register Empty)** bit in **UCSR0A** is 1.
- **Interrupt-driven method**
 - **RX operation:**
 - Enable RX interrupt and implement the ISR: **USART_RX_vect**
 - **TX operation:**
 - Enable TX interrupt and implement the ISR: **USART_TX_vect**

USART0 Register: UDR0

UDR_n – USART I/O Data Register n

Bit	7	6	5	4	3	2	1	0	UDR _n (Read)	UDR _n (Write)
	RXB[7:0]									
	TXB[7:0]									
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_n. The transmit data buffer register (TXB) will be the destination for data written to the UDR_n register location. Reading the UDR_n register location will return the contents of the receive data buffer register (RXB).

For 5-, 6-, or 7-bit characters the upper unused bits will be ignored by the transmitter and set to zero by the receiver.

The transmit buffer can only be written when the UDREn flag in the UCSR_{nA} register is set. Data written to UDR_n when the UDREn flag is not set, will be ignored by the USART transmitter. When data is written to the transmit buffer, and the transmitter is enabled, the transmitter will load the data into the transmit shift register when the shift register is empty. Then the data will be serially transmitted on the TxD_n pin.

The receive buffer consists of a two level FIFO. The FIFO will change its state whenever the receive buffer is accessed. Due to this behavior of the receive buffer, do not use read-modify-write instructions (SBI and CBI) on this location. Be careful when using bit test instructions (SBIC and SBIS), since these also will change the state of the FIFO.

USART0 Register: UCSR0A

UCSRnA – USART Control and Status Register n A

Bit	7	6	5	4	3	2	1	0	UCSRnA
	RXCn	TXCn	UDREn	FEn	DORn	UPEn	U2Xn	MPCMn	UCSRnA
Read/Write	R	R/W	R	R	R	R	R/W	R/W	

Initial Value

0

0

1

0

0

0

0

0

0

0

- **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 (i.e., does not contain any unread data). If the receiver is disabled, the receive buffer will be flushed and consequently the RXCn bit will become zero. The RXCn flag can be used to generate a receive complete interrupt (see description of the RXCIEn bit).

- **Bit 6 – TXCn: 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 (UDRn). The TXCn 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 TXCn flag can generate a transmit complete interrupt (see description of the TXCIEn bit).

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

The UDREn flag indicates if the transmit buffer (UDRn) is ready to receive new data. If UDREn is one, the buffer is empty, and therefore ready to be written. The UDREn flag can generate a data register empty interrupt (see description of the UDRIEn bit). UDREn is set after a reset to indicate that the transmitter is ready.

- **Bit 4 – FEn: Frame Error**

This bit is set if the next character in the receive buffer had a frame error when received. I.e., when the first stop bit of the next character in the receive buffer is zero. This bit is valid until the receive buffer (UDRn) is read. The FEn bit is zero when the stop bit of received data is one. Always set this bit to zero when writing to UCSRnA.

USART0 Register: UCSR0A

UCSRnA – USART Control and Status Register n A

Bit	7	6	5	4	3	2	1	0	UCSRnA
	RXCn	TXCn	UDREn	FEn	DORn	UPEn	U2Xn	MPCMn	
Read/Write	R	R/W	R	R	R	R	R/W	R/W	
Initial Value	0	0	1	0	0	0	0	0	

- **Bit 3 – DORn: Data OverRun**

This bit is set if a data overrun condition is detected. A data overrun occurs when the receive buffer is full (two characters), it is a new character waiting in the receive shift register, and a new start bit is detected. This bit is valid until the receive buffer (UDRn) is read. Always set this bit to zero when writing to UCSRnA.

- **Bit 2 – UPEn: USART Parity Error**

This bit is set if the next character in the receive buffer had a parity error when received and the parity checking was enabled at that point (UPMn1 = 1). This bit is valid until the receive buffer (UDRn) is read. Always set this bit to zero when writing to UCSRnA.

- **Bit 1 – U2Xn: Double the USART Transmission Speed**

This bit only has effect for the asynchronous operation. Write this bit to zero when using synchronous operation.

Writing this bit to one will reduce the divisor of the baud rate divider from 16 to 8 effectively doubling the transfer rate for asynchronous communication.

- **Bit 0 – MPCMn: Multi-processor Communication Mode**

This bit enables the multi-processor communication mode. When the MPCMn bit is written to one, all the incoming frames received by the USART receiver that do not contain address information will be ignored. The transmitter is unaffected by the MPCMn setting.

USART0 Register: UCSR0B

UCSRnB – USART Control and Status Register n B

Bit	7	6	5	4	3	2	1	0	UCSRnB
	RXCIEn	TXCIEn	UDRIEn	RXENn	TXENn	UCSZn2	RXB8n	TXB8n	UCSRnB
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 7 – RXCIEn: RX Complete Interrupt Enable n**

Writing this bit to one enables interrupt on the RXCn flag. A USART receive complete interrupt will be generated only if the RXCIEn bit is written to one, the global interrupt flag in SREG is written to one and the RXCn bit in UCSRnA is set.

- **Bit 6 – TXCIEn: TX Complete Interrupt Enable n**

Writing this bit to one enables interrupt on the TXCn flag. A USART transmit complete interrupt will be generated only if the TXCIEn bit is written to one, the global interrupt flag in SREG is written to one and the TXCn bit in UCSRnA is set.

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

Writing this bit to one enables interrupt on the UDREn flag. A data register empty interrupt will be generated only if the UDRIEn bit is written to one, the global interrupt flag in SREG is written to one and the UDREn bit in UCSRnA is set.

- **Bit 4 – RXENn: Receiver Enable n**

Writing this bit to one enables the USART receiver. The receiver will override normal port operation for the RxDn pin when enabled. Disabling the receiver will flush the receive buffer invalidating the FEn, DORn, and UPEn flags.

USART0 Register: UCSR0B

UCSRnB – USART Control and Status Register n B

Bit	7	6	5	4	3	2	1	0	UCSRnB
	RXCIEn	TXCIEn	UDRIEn	RXENn	TXENn	UCSZn2	RXB8n	TXB8n	UCSRnB
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 3 – TXENn: Transmitter Enable n**

Writing this bit to one enables the USART transmitter. The transmitter will override normal port operation for the TxDn pin when enabled. The disabling of the transmitter (writing TXENn 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 TxDn port.

- **Bit 2 – UCSZn2: Character Size n**

The UCSZn2 bits combined with the UCSZn1:0 bit in UCSRnC sets the number of data bits (character size) in a frame the receiver and transmitter use.

- **Bit 1 – RXB8n: Receive Data Bit 8 n**

RXB8n is the ninth data bit of the received character when operating with serial frames with nine data bits. Must be read before reading the low bits from UDRn.

- **Bit 0 – TXB8n: Transmit Data Bit 8 n**

TXB8n is the ninth data bit in the character to be transmitted when operating with serial frames with nine data bits. Must be written before writing the low bits to UDRn.

USART0 Register: UCSR0C

UCSRnC – USART Control and Status Register n C

Bit	7	6	5	4	3	2	1	0	UCSRnC
	UMSELn1	UMSELn0	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn	
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	

- Bits 7:6 – UMSELn1:0 USART Mode Select

UMSELn Bits Settings

UMSELn1	UMSELn0	Mode
0	0	Asynchronous USART
0	1	Synchronous USART
1	0	(Reserved)
1	1	Master SPI (MSPIM)

- Bits 5:4 – UPMn1:0: Parity Mode

These bits enable and set type of parity generation and check. If enabled, the transmitter will automatically generate and send the parity of the transmitted data bits within each frame. The receiver will generate a parity value for the incoming data and compare it to the UPMn setting. If a mismatch is detected, the UPEn flag in UCSRnA will be set.

UPMn Bits Settings

UPMn1	UPMn0	Parity Mode
0	0	Disabled
0	1	Reserved
1	0	Enabled, even parity
1	1	Enabled, odd parity

- Bit 3 – USBSn: Stop Bit Select

This bit selects the number of stop bits to be inserted by the transmitter. The receiver ignores this setting.

USBS Bit Settings

USBSn	Stop Bit(s)
0	1-bit
1	2-bit

USART0 Register: UCSR0C

UCSRnC – USART Control and Status Register n C

Bit	7	6	5	4	3	2	1	0	UCSRnC
Read/Write	R/W								
Initial Value	0	0	0	0	0	1	1	0	

- Bit 2:1 – UCSZn1:0: Character Size

The UCSZn1:0 bits combined with the UCSZn2 bit in UCSRnB sets the number of data bits (character size) in a frame the receiver and transmitter use.

UCSZn Bits Settings

UCSZn2	UCSZn1	UCSZn0	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

- Bit 0 – UCPOLn: Clock Polarity

This bit is used for synchronous mode only. Write this bit to zero when asynchronous mode is used. The UCPOLn bit sets the relationship between data output change and data input sample, and the synchronous clock (XCKn).

UCPOLn Bit Settings

UCPOLn	Transmitted Data Changed (Output of TxDn Pin)	Received Data Sampled (Input on RxDn Pin)
0	Rising XCKn edge	Falling XCKn edge
1	Falling XCKn edge	Rising XCKn edge

USART0 Register: UBRR0

UBRRnL and UBRRnH – USART Baud Rate Registers

Bit	15	14	13	12	11	10	9	8					
	–	–	–	–	UBRRn[11:8]								UBRRnH
	UBRRn[7:0]									UBRRnL			
Read/Write	7	6	5	4	3	2	1	0					
	R	R	R	R	R/W	R/W	R/W	R/W					
Initial Value	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W					
	0	0	0	0	0	0	0	0					
	0	0	0	0	0	0	0	0					

- **Bit 15:12 – Reserved Bits**

These bits are reserved for future use. For compatibility with future devices, these bit must be written to zero when UBRRnH is written.

- **Bit 11:0 – UBRR11:0: USART Baud Rate Register**

This is a 12-bit register which contains the USART baud rate. The UBRRnH contains the four most significant bits, and the UBRRnL contains the eight least significant bits of the USART baud rate. Ongoing transmissions by the transmitter and receiver will be corrupted if the baud rate is changed. Writing UBRRnL will trigger an immediate update of the baud rate prescaler.

$$\text{Baudrate (Normal Speed)} = \frac{f_{CPU}}{16 \cdot (UBRR0 + 1)}$$

$$\text{Baudrate (Double Speed)} = \frac{f_{CPU}}{8 \cdot (UBRR0 + 1)}$$

Using printf() via UART

- Using the **AVR libc** and **AVR-GCC compiler toolchain**, the standard C `printf()` function can be redirected to USART by creating a custom `uart_putchar()` function and using:

```
fdevopen(&uart_putchar, NULL);
```

- The user-defined `uart_putchar()` uses the **USART** module in asynchronous mode to send chars.
- This links **stdout** (standard output stream) to **USART** so that calls to `printf()` send text bytes to the **USART**.

Code Example: USART – Serial TX Mode

Code Snippet 1

```
#define F_CPU 16000000UL
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <stdio.h>

#define SPEED_2X
#define BAUD 115200

#ifndef SPEED_2X
#define UBRR_VALUE ((F_CPU / (8UL * BAUD)) - 1)
#else
#define UBRR_VALUE ((F_CPU / (16UL * BAUD)) - 1)
#endif

void usart_init(void) {
    // Set baud rate
    UBRR0H = (uint8_t)(UBRR_VALUE >> 8);
    UBRR0L = (uint8_t)UBRR_VALUE;
#ifndef SPEED_2X
    UCSR0A = (1 << U2X0);      // Enable double speed
#else
    UCSR0A = 0;
#endif
    UCSR0C = (1 << UCSZ01) | (1 << UCSZ00); // Use 8N1
    UCSR0B = (1 << RXEN0) | (1 << TXEN0); // Enable both TX & RX
}
// Code continues on the next page...
```

Code Example: USART – Serial TX Mode

Code Snippet 2

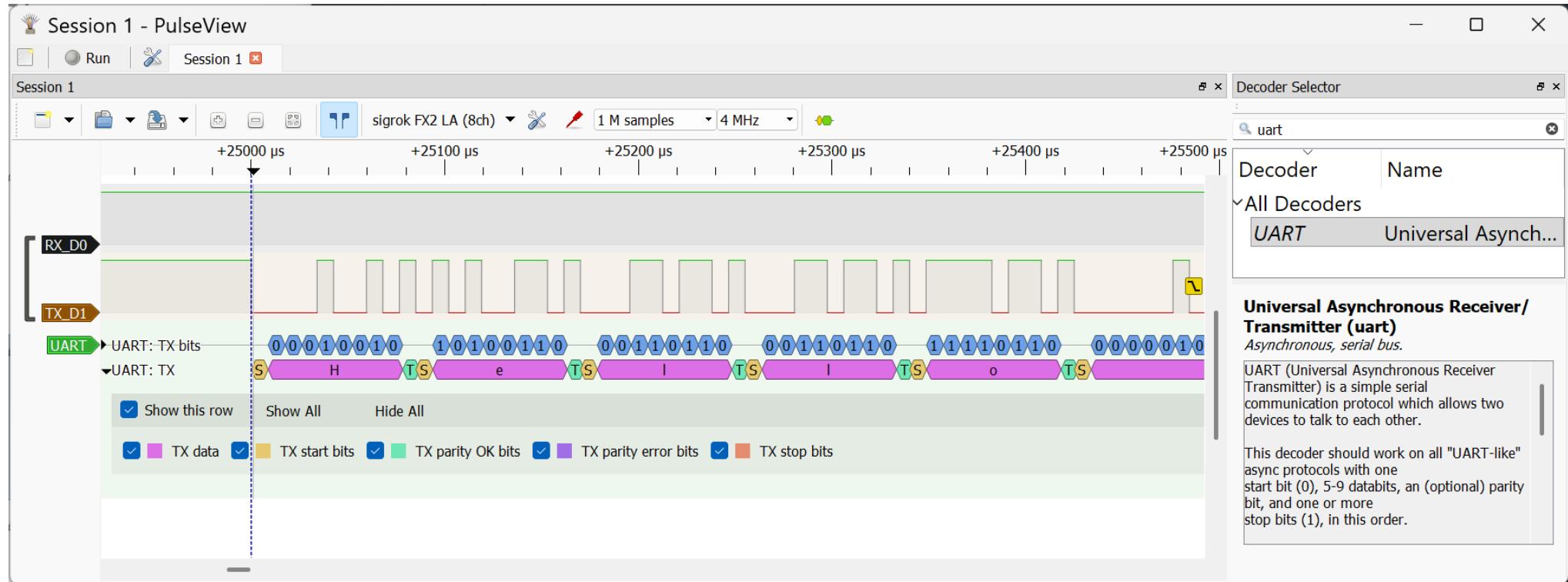
```
void usart_putchar(char c) {
    while (!(UCSR0A & (1 << UDRE0))); // Wait TX buffer empty
    UDR0 = c; // Write char to the data register
}

void usart_send_string(const char *s) {
    while (*s) {
        usart_putchar(*s++);
    }
}

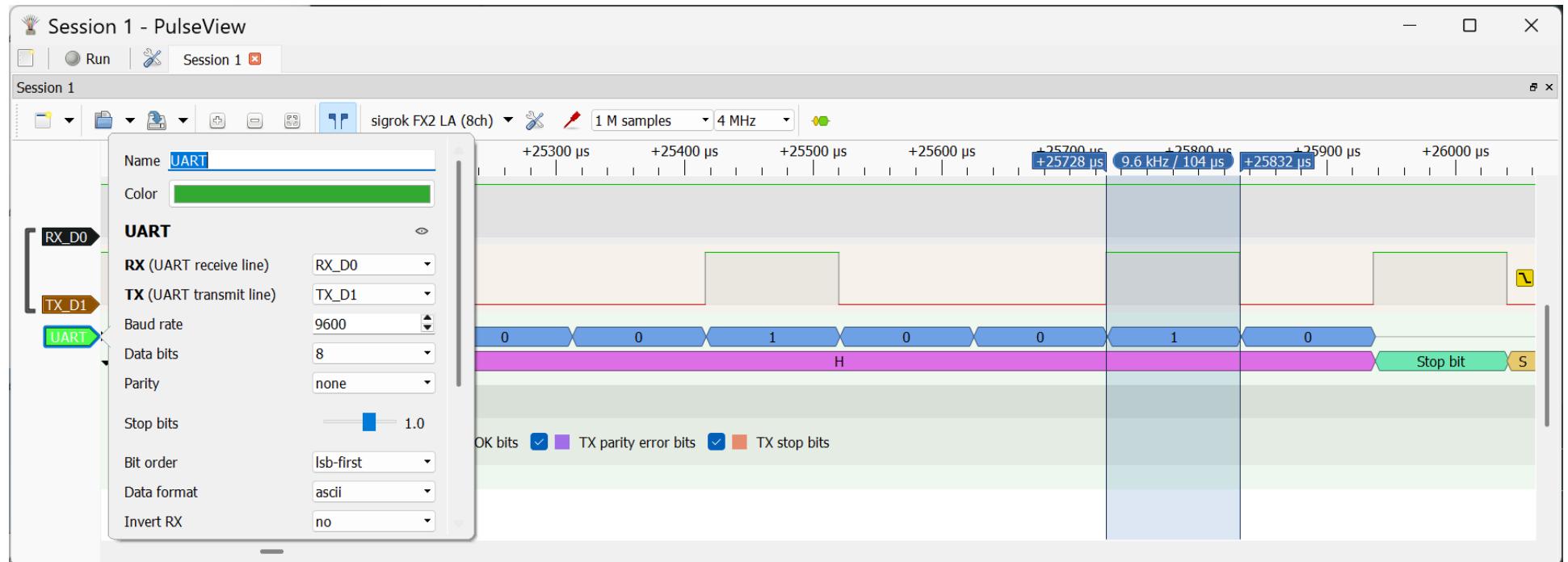
char usart_getchar(void) {
    while (!(UCSR0A & (1 << RXC0))); // Wait for RX incoming byte
    return UDR0;
}
```

```
int main(void) {
    usart_init();
    while (1) {
        usart_send_string("Hello from ATmega328P\r\n");
        _delay_ms(1000); // 1 second blocking delay
    }
}
```

USB Logic Analyzer + PulseView / UART Protocol Analyzer



USB Logic Analyzer + PulseView / UART Protocol Analyzer



Baudrate: 9600

Code Example: printf() via USART

```
// Use Code Snippets 1 & 2

FILE uart_stdout = FDEV_SETUP_STREAM(usart_putchar, NULL, _FDEV_SETUP_WRITE);

int main(void) {
    uint32_t counter = 0;
    char buf[16];
    usart_init();
    stdout = &uart_stdout;
    while (1) {
        snprintf( buf, 16, "Count = %lu\r\n", ++counter);
        printf( buf );
        _delay_ms(1000);
    }
}
```

Wokwi Simulator

New Arduino Uno Project - Wokwi

wokwi.com/projects/new/arduino-uno

WOKWI SAVE SHARE

sketch.ino • diagram.json • main.c • Library Manager

```
34 void usart_send_string(const char *s) {
35     ...
36 }
37 }
38 }
39 }
40 char usart_getchar(void) {
41     while (!(UCSR0A & (1 << RXC0))); // Wait for RX incoming byte
42     return UDR0;
43 }
44 }
45 FILE uart_stdout = FDEV_SETUP_STREAM(usart_putchar, NULL, _FDEV_SETUP_WRITE);
46 }
47 int main(void) {
48     uint32_t counter = 0;
49     char buf[16];
50     usart_init();
51     stdout = &uart_stdout;
52     while (1) {
53         sprintf( buf, 16, "Count = %lu\r\n", ++counter);
54         printf( buf );
55         _delay_ms(1000);
56     }
57 }
```

Simulation

00:15.450 100%

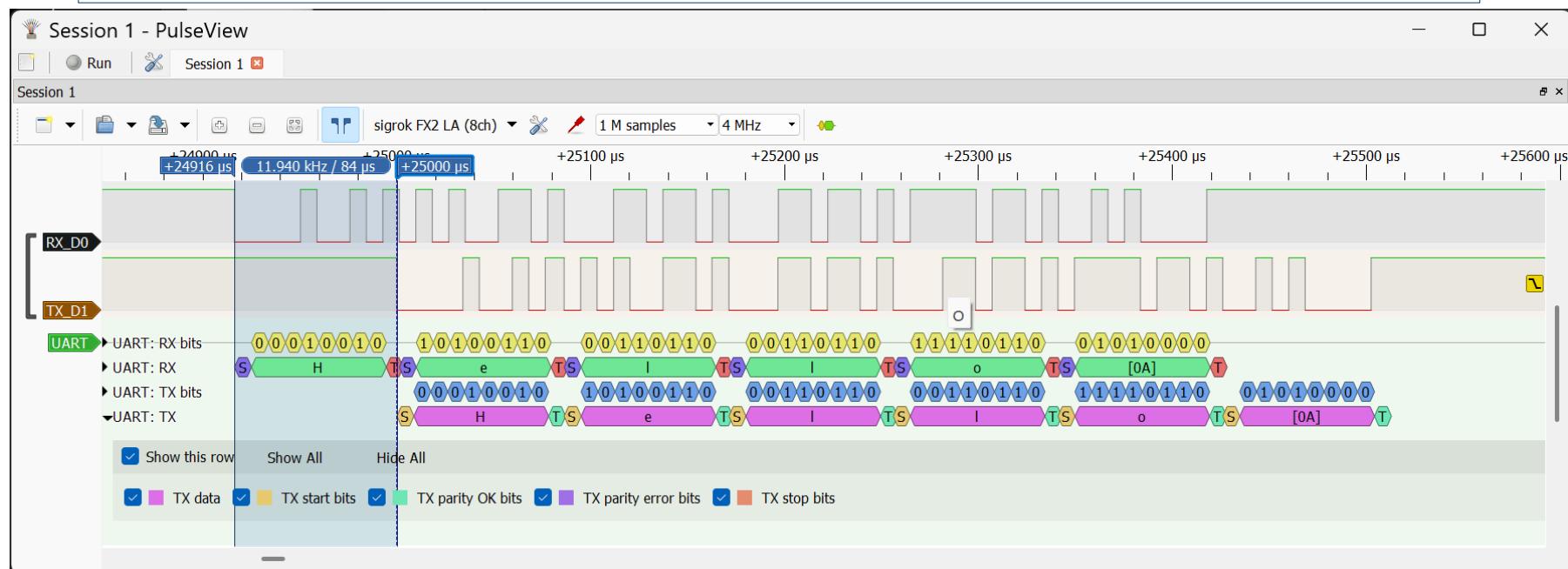
<img alt="Arduino Uno simulation board showing digital pins 13, 12, 11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0 connected to ground. Pin 14 is connected to AREF. Pin 15 is connected to 5V. Pin 16 is connected to GND. Pin 17 is connected to power. Pin 18 is connected to GND. Pin 19 is connected to A0. Pin 20 is connected to A1. Pin 21 is connected to A2. Pin 22 is connected to A3. Pin 23 is connected to A4. Pin 24 is connected to A5. Pin 25 is connected to GND. Pin 26 is connected to GND. Pin 27 is connected to GND. Pin 28 is connected to GND. Pin 29 is connected to GND. Pin 30 is connected to GND. Pin 31 is connected to GND. Pin 32 is connected to GND. Pin 33 is connected to GND. Pin 34 is connected to GND. Pin 35 is connected to GND. Pin 36 is connected to GND. Pin 37 is connected to GND. Pin 38 is connected to GND. Pin 39 is connected to GND. Pin 40 is connected to GND. Pin 41 is connected to GND. Pin 42 is connected to GND. Pin 43 is connected to GND. Pin 44 is connected to GND. Pin 45 is connected to GND. Pin 46 is connected to GND. Pin 47 is connected to GND. Pin 48 is connected to GND. Pin 49 is connected to GND. Pin 50 is connected to GND. Pin 51 is connected to GND. Pin 52 is connected to GND. Pin 53 is connected to GND. Pin 54 is connected to GND. Pin 55 is connected to GND. Pin 56 is connected to GND. Pin 57 is connected to GND. Pin 58 is connected to GND. Pin 59 is connected to GND. Pin 60 is connected to GND. Pin 61 is connected to GND. Pin 62 is connected to GND. Pin 63 is connected to GND. Pin 64 is connected to GND. Pin 65 is connected to GND. Pin 66 is connected to GND. Pin 67 is connected to GND. Pin 68 is connected to GND. Pin 69 is connected to GND. Pin 70 is connected to GND. Pin 71 is connected to GND. Pin 72 is connected to GND. Pin 73 is connected to GND. Pin 74 is connected to GND. Pin 75 is connected to GND. Pin 76 is connected to GND. Pin 77 is connected to GND. Pin 78 is connected to GND. Pin 79 is connected to GND. Pin 80 is connected to GND. Pin 81 is connected to GND. Pin 82 is connected to GND. Pin 83 is connected to GND. Pin 84 is connected to GND. Pin 85 is connected to GND. Pin 86 is connected to GND. Pin 87 is connected to GND. Pin 88 is connected to GND. Pin 89 is connected to GND. Pin 90 is connected to GND. Pin 91 is connected to GND. Pin 92 is connected to GND. Pin 93 is connected to GND. Pin 94 is connected to GND. Pin 95 is connected to GND. Pin 96 is connected to GND. Pin 97 is connected to GND. Pin 98 is connected to GND. Pin 99 is connected to GND. Pin 100 is connected to GND. Pin 101 is connected to GND. Pin 102 is connected to GND. Pin 103 is connected to GND. Pin 104 is connected to GND. Pin 105 is connected to GND. Pin 106 is connected to GND. Pin 107 is connected to GND. Pin 108 is connected to GND. Pin 109 is connected to GND. Pin 110 is connected to GND. Pin 111 is connected to GND. Pin 112 is connected to GND. Pin 113 is connected to GND. Pin 114 is connected to GND. Pin 115 is connected to GND. Pin 116 is connected to GND. Pin 117 is connected to GND. Pin 118 is connected to GND. Pin 119 is connected to GND. Pin 120 is connected to GND. Pin 121 is connected to GND. Pin 122 is connected to GND. Pin 123 is connected to GND. Pin 124 is connected to GND. Pin 125 is connected to GND. Pin 126 is connected to GND. Pin 127 is connected to GND. Pin 128 is connected to GND. Pin 129 is connected to GND. Pin 130 is connected to GND. Pin 131 is connected to GND. Pin 132 is connected to GND. Pin 133 is connected to GND. Pin 134 is connected to GND. Pin 135 is connected to GND. Pin 136 is connected to GND. Pin 137 is connected to GND. Pin 138 is connected to GND. Pin 139 is connected to GND. Pin 140 is connected to GND. Pin 141 is connected to GND. Pin 142 is connected to GND. Pin 143 is connected to GND. Pin 144 is connected to GND. Pin 145 is connected to GND. Pin 146 is connected to GND. Pin 147 is connected to GND. Pin 148 is connected to GND. Pin 149 is connected to GND. Pin 150 is connected to GND. Pin 151 is connected to GND. Pin 152 is connected to GND. Pin 153 is connected to GND. Pin 154 is connected to GND. Pin 155 is connected to GND. Pin 156 is connected to GND. Pin 157 is connected to GND. Pin 158 is connected to GND. Pin 159 is connected to GND. Pin 160 is connected to GND. Pin 161 is connected to GND. Pin 162 is connected to GND. Pin 163 is connected to GND. Pin 164 is connected to GND. Pin 165 is connected to GND. Pin 166 is connected to GND. Pin 167 is connected to GND. Pin 168 is connected to GND. Pin 169 is connected to GND. Pin 170 is connected to GND. Pin 171 is connected to GND. Pin 172 is connected to GND. Pin 173 is connected to GND. Pin 174 is connected to GND. Pin 175 is connected to GND. Pin 176 is connected to GND. Pin 177 is connected to GND. Pin 178 is connected to GND. Pin 179 is connected to GND. Pin 180 is connected to GND. Pin 181 is connected to GND. Pin 182 is connected to GND. Pin 183 is connected to GND. Pin 184 is connected to GND. Pin 185 is connected to GND. Pin 186 is connected to GND. Pin 187 is connected to GND. Pin 188 is connected to GND. Pin 189 is connected to GND. Pin 190 is connected to GND. Pin 191 is connected to GND. Pin 192 is connected to GND. Pin 193 is connected to GND. Pin 194 is connected to GND. Pin 195 is connected to GND. Pin 196 is connected to GND. Pin 197 is connected to GND. Pin 198 is connected to GND. Pin 199 is connected to GND. Pin 200 is connected to GND. Pin 201 is connected to GND. Pin 202 is connected to GND. Pin 203 is connected to GND. Pin 204 is connected to GND. Pin 205 is connected to GND. Pin 206 is connected to GND. Pin 207 is connected to GND. Pin 208 is connected to GND. Pin 209 is connected to GND. Pin 210 is connected to GND. Pin 211 is connected to GND. Pin 212 is connected to GND. Pin 213 is connected to GND. Pin 214 is connected to GND. Pin 215 is connected to GND. Pin 216 is connected to GND. Pin 217 is connected to GND. Pin 218 is connected to GND. Pin 219 is connected to GND. Pin 220 is connected to GND. Pin 221 is connected to GND. Pin 222 is connected to GND. Pin 223 is connected to GND. Pin 224 is connected to GND. Pin 225 is connected to GND. Pin 226 is connected to GND. Pin 227 is connected to GND. Pin 228 is connected to GND. Pin 229 is connected to GND. Pin 230 is connected to GND. Pin 231 is connected to GND. Pin 232 is connected to GND. Pin 233 is connected to GND. Pin 234 is connected to GND. Pin 235 is connected to GND. Pin 236 is connected to GND. Pin 237 is connected to GND. Pin 238 is connected to GND. Pin 239 is connected to GND. Pin 240 is connected to GND. Pin 241 is connected to GND. Pin 242 is connected to GND. Pin 243 is connected to GND. Pin 244 is connected to GND. Pin 245 is connected to GND. Pin 246 is connected to GND. Pin 247 is connected to GND. Pin 248 is connected to GND. Pin 249 is connected to GND. Pin 250 is connected to GND. Pin 251 is connected to GND. Pin 252 is connected to GND. Pin 253 is connected to GND. Pin 254 is connected to GND. Pin 255 is connected to GND. Pin 256 is connected to GND. Pin 257 is connected to GND. Pin 258 is connected to GND. Pin 259 is connected to GND. Pin 260 is connected to GND. Pin 261 is connected to GND. Pin 262 is connected to GND. Pin 263 is connected to GND. Pin 264 is connected to GND. Pin 265 is connected to GND. Pin 266 is connected to GND. Pin 267 is connected to GND. Pin 268 is connected to GND. Pin 269 is connected to GND. Pin 270 is connected to GND. Pin 271 is connected to GND. Pin 272 is connected to GND. Pin 273 is connected to GND. Pin 274 is connected to GND. Pin 275 is connected to GND. Pin 276 is connected to GND. Pin 277 is connected to GND. Pin 278 is connected to GND. Pin 279 is connected to GND. Pin 280 is connected to GND. Pin 281 is connected to GND. Pin 282 is connected to GND. Pin 283 is connected to GND. Pin 284 is connected to GND. Pin 285 is connected to GND. Pin 286 is connected to GND. Pin 287 is connected to GND. Pin 288 is connected to GND. Pin 289 is connected to GND. Pin 290 is connected to GND. Pin 291 is connected to GND. Pin 292 is connected to GND. Pin 293 is connected to GND. Pin 294 is connected to GND. Pin 295 is connected to GND. Pin 296 is connected to GND. Pin 297 is connected to GND. Pin 298 is connected to GND. Pin 299 is connected to GND. Pin 300 is connected to GND. Pin 301 is connected to GND. Pin 302 is connected to GND. Pin 303 is connected to GND. Pin 304 is connected to GND. Pin 305 is connected to GND. Pin 306 is connected to GND. Pin 307 is connected to GND. Pin 308 is connected to GND. Pin 309 is connected to GND. Pin 310 is connected to GND. Pin 311 is connected to GND. Pin 312 is connected to GND. Pin 313 is connected to GND. Pin 314 is connected to GND. Pin 315 is connected to GND. Pin 316 is connected to GND. Pin 317 is connected to GND. Pin 318 is connected to GND. Pin 319 is connected to GND. Pin 320 is connected to GND. Pin 321 is connected to GND. Pin 322 is connected to GND. Pin 323 is connected to GND. Pin 324 is connected to GND. Pin 325 is connected to GND. Pin 326 is connected to GND. Pin 327 is connected to GND. Pin 328 is connected to GND. Pin 329 is connected to GND. Pin 330 is connected to GND. Pin 331 is connected to GND. Pin 332 is connected to GND. Pin 333 is connected to GND. Pin 334 is connected to GND. Pin 335 is connected to GND. Pin 336 is connected to GND. Pin 337 is connected to GND. Pin 338 is connected to GND. Pin 339 is connected to GND. Pin 340 is connected to GND. Pin 341 is connected to GND. Pin 342 is connected to GND. Pin 343 is connected to GND. Pin 344 is connected to GND. Pin 345 is connected to GND. Pin 346 is connected to GND. Pin 347 is connected to GND. Pin 348 is connected to GND. Pin 349 is connected to GND. Pin 350 is connected to GND. Pin 351 is connected to GND. Pin 352 is connected to GND. Pin 353 is connected to GND. Pin 354 is connected to GND. Pin 355 is connected to GND. Pin 356 is connected to GND. Pin 357 is connected to GND. Pin 358 is connected to GND. Pin 359 is connected to GND. Pin 360 is connected to GND. Pin 361 is connected to GND. Pin 362 is connected to GND. Pin 363 is connected to GND. Pin 364 is connected to GND. Pin 365 is connected to GND. Pin 366 is connected to GND. Pin 367 is connected to GND. Pin 368 is connected to GND. Pin 369 is connected to GND. Pin 370 is connected to GND. Pin 371 is connected to GND. Pin 372 is connected to GND. Pin 373 is connected to GND. Pin 374 is connected to GND. Pin 375 is connected to GND. Pin 376 is connected to GND. Pin 377 is connected to GND. Pin 378 is connected to GND. Pin 379 is connected to GND. Pin 380 is connected to GND. Pin 381 is connected to GND. Pin 382 is connected to GND. Pin 383 is connected to GND. Pin 384 is connected to GND. Pin 385 is connected to GND. Pin 386 is connected to GND. Pin 387 is connected to GND. Pin 388 is connected to GND. Pin 389 is connected to GND. Pin 390 is connected to GND. Pin 391 is connected to GND. Pin 392 is connected to GND. Pin 393 is connected to GND. Pin 394 is connected to GND. Pin 395 is connected to GND. Pin 396 is connected to GND. Pin 397 is connected to GND. Pin 398 is connected to GND. Pin 399 is connected to GND. Pin 400 is connected to GND. Pin 401 is connected to GND. Pin 402 is connected to GND. Pin 403 is connected to GND. Pin 404 is connected to GND. Pin 405 is connected to GND. Pin 406 is connected to GND. Pin 407 is connected to GND. Pin 408 is connected to GND. Pin 409 is connected to GND. Pin 410 is connected to GND. Pin 411 is connected to GND. Pin 412 is connected to GND. Pin 413 is connected to GND. Pin 414 is connected to GND. Pin 415 is connected to GND. Pin 416 is connected to GND. Pin 417 is connected to GND. Pin 418 is connected to GND. Pin 419 is connected to GND. Pin 420 is connected to GND. Pin 421 is connected to GND. Pin 422 is connected to GND. Pin 423 is connected to GND. Pin 424 is connected to GND. Pin 425 is connected to GND. Pin 426 is connected to GND. Pin 427 is connected to GND. Pin 428 is connected to GND. Pin 429 is connected to GND. Pin 430 is connected to GND. Pin 431 is connected to GND. Pin 432 is connected to GND. Pin 433 is connected to GND. Pin 434 is connected to GND. Pin 435 is connected to GND. Pin 436 is connected to GND. Pin 437 is connected to GND. Pin 438 is connected to GND. Pin 439 is connected to GND. Pin 440 is connected to GND. Pin 441 is connected to GND. Pin 442 is connected to GND. Pin 443 is connected to GND. Pin 444 is connected to GND. Pin 445 is connected to GND. Pin 446 is connected to GND. Pin 447 is connected to GND. Pin 448 is connected to GND. Pin 449 is connected to GND. Pin 450 is connected to GND. Pin 451 is connected to GND. Pin 452 is connected to GND. Pin 453 is connected to GND. Pin 454 is connected to GND. Pin 455 is connected to GND. Pin 456 is connected to GND. Pin 457 is connected to GND. Pin 458 is connected to GND. Pin 459 is connected to GND. Pin 460 is connected to GND. Pin 461 is connected to GND. Pin 462 is connected to GND. Pin 463 is connected to GND. Pin 464 is connected to GND. Pin 465 is connected to GND. Pin 466 is connected to GND. Pin 467 is connected to GND. Pin 468 is connected to GND. Pin 469 is connected to GND. Pin 470 is connected to GND. Pin 471 is connected to GND. Pin 472 is connected to GND. Pin 473 is connected to GND. Pin 474 is connected to GND. Pin 475 is connected to GND. Pin 476 is connected to GND. Pin 477 is connected to GND. Pin 478 is connected to GND. Pin 479 is connected to GND. Pin 480 is connected to GND. Pin 481 is connected to GND. Pin 482 is connected to GND. Pin 483 is connected to GND. Pin 484 is connected to GND. Pin 485 is connected to GND. Pin 486 is connected to GND. Pin 487 is connected to GND. Pin 488 is connected to GND. Pin 489 is connected to GND. Pin 490 is connected to GND. Pin 491 is connected to GND. Pin 492 is connected to GND. Pin 493 is connected to GND. Pin 494 is connected to GND. Pin 495 is connected to GND. Pin 496 is connected to GND. Pin 497 is connected to GND. Pin 498 is connected to GND. Pin 499 is connected to GND. Pin 500 is connected to GND. Pin 501 is connected to GND. Pin 502 is connected to GND. Pin 503 is connected to GND. Pin 504 is connected to GND. Pin 505 is connected to GND. Pin 506 is connected to GND. Pin 507 is connected to GND. Pin 508 is connected to GND. Pin 509 is connected to GND. Pin 510 is connected to GND. Pin 511 is connected to GND. Pin 512 is connected to GND. Pin 513 is connected to GND. Pin 514 is connected to GND. Pin 515 is connected to GND. Pin 516 is connected to GND. Pin 517 is connected to GND. Pin 518 is connected to GND. Pin 519 is connected to GND. Pin 520 is connected to GND. Pin 521 is connected to GND. Pin 522 is connected to GND. Pin 523 is connected to GND. Pin 524 is connected to GND. Pin 525 is connected to GND. Pin 526 is connected to GND. Pin 527 is connected to GND. Pin 528 is connected to GND. Pin 529 is connected to GND. Pin 530 is connected to GND. Pin 531 is connected to GND. Pin 532 is connected to GND. Pin 533 is connected to GND. Pin 534 is connected to GND. Pin 535 is connected to GND. Pin 536 is connected to GND. Pin 537 is connected to GND. Pin 538 is connected to GND. Pin 539 is connected to GND. Pin 540 is connected to GND. Pin 541 is connected to GND. Pin 542 is connected to GND. Pin 543 is connected to GND. Pin 544 is connected to GND. Pin 545 is connected to GND. Pin 546 is connected to GND. Pin 547 is connected to GND. Pin 548 is connected to GND. Pin 549 is connected to GND. Pin 550 is connected to GND. Pin 551 is connected to GND. Pin 552 is connected to GND. Pin 553 is connected to GND. Pin 554 is connected to GND. Pin 555 is connected to GND. Pin 556 is connected to GND. Pin 557 is connected to GND. Pin 558 is connected to GND. Pin 559 is connected to GND. Pin 560 is connected to GND. Pin 561 is connected to GND. Pin 562 is connected to GND. Pin 563 is connected to GND. Pin 564 is connected to GND. Pin 565 is connected to GND. Pin 566 is connected to GND. Pin 567 is connected to GND. Pin 568 is connected to GND. Pin 569 is connected to GND. Pin 570 is connected to GND. Pin 571 is connected to GND. Pin 572 is connected to GND. Pin 573 is connected to GND. Pin 574 is connected to GND. Pin 575 is connected to GND. Pin 576 is connected to GND. Pin 577 is connected to GND. Pin 578 is connected to GND. Pin 579 is connected to GND. Pin 580 is connected to GND. Pin 581 is connected to GND. Pin 582 is connected to GND. Pin 583 is connected to GND. Pin 584 is connected to GND. Pin 585 is connected to GND. Pin 586 is connected to GND. Pin 587 is connected to GND. Pin 588 is connected to GND. Pin 589 is connected to GND. Pin 590 is connected to GND. Pin 591 is connected to GND. Pin 592 is connected to GND. Pin 593 is connected to GND. Pin 594 is connected to GND. Pin 595 is connected to GND. Pin 596 is connected to GND. Pin 597 is connected to GND. Pin 598 is connected to GND. Pin 599 is connected to GND. Pin 600 is connected to GND. Pin 601 is connected to GND. Pin 602 is connected to GND. Pin 603 is connected to GND. Pin 604 is connected to GND. Pin 605 is connected to GND. Pin 606 is connected to GND. Pin 607 is connected to GND. Pin 608 is connected to GND. Pin 609 is connected to GND. Pin 610 is connected to GND. Pin 611 is connected to GND. Pin 612 is connected to GND. Pin 613 is connected to GND. Pin 614 is connected to GND. Pin 615 is connected to GND. Pin 616 is connected to GND. Pin 617 is connected to GND. Pin 618 is connected to GND. Pin 619 is connected to GND. Pin 620 is connected to GND. Pin 621 is connected to GND. Pin 622 is connected to GND. Pin 623 is connected to GND. Pin 624 is connected to GND. Pin 625 is connected to GND. Pin 626 is connected to GND. Pin 627 is connected to GND. Pin 628 is connected to GND. Pin 629 is connected to GND. Pin 630 is connected to GND. Pin 631 is connected to GND. Pin 632 is connected to GND. Pin 633 is connected to GND. Pin 634 is connected to GND. Pin 635 is connected to GND. Pin 636 is connected to GND. Pin 637 is connected to GND. Pin 638 is connected to GND. Pin 639 is connected to GND. Pin 640 is connected to GND. Pin 641 is connected to GND. Pin 642 is connected to GND. Pin 643 is connected to GND. Pin 644 is connected to GND. Pin 645 is connected to GND. Pin 646 is connected to GND. Pin 647 is connected to GND. Pin 648 is connected to GND. Pin 649 is connected to GND. Pin 650 is connected to GND. Pin 651 is connected to GND. Pin 652 is connected to GND. Pin 653 is connected to GND. Pin 654 is connected to GND. Pin 655 is connected to GND. Pin 656 is connected to GND. Pin 657 is connected to GND. Pin 658 is connected to GND. Pin 659 is connected to GND. Pin 660 is connected to GND. Pin 661 is connected to GND. Pin 662 is connected to GND. Pin 663 is connected to GND. Pin 664 is connected to GND. Pin 665 is connected to GND. Pin 666 is connected to GND. Pin 667 is connected to GND. Pin 668 is connected to GND. Pin 669 is connected to GND. Pin 670 is connected to GND. Pin 671 is connected to GND. Pin 672 is connected to GND. Pin 673 is connected to GND. Pin 674 is connected to GND. Pin 675 is connected to GND. Pin 676 is connected to GND. Pin 677 is connected to GND. Pin 678 is connected to GND. Pin 679 is connected to GND. Pin 680 is connected to GND. Pin 681 is connected to GND. Pin 682 is connected to GND. Pin 683 is connected to GND. Pin 684 is connected to GND. Pin 685 is connected to GND. Pin 686 is connected to GND. Pin 687 is connected to GND. Pin 688 is connected to GND. Pin 689 is connected to GND. Pin 690 is connected to GND. Pin 691 is connected to GND. Pin 692 is connected to GND. Pin 693 is connected to GND. Pin 694 is connected to GND. Pin 695 is connected to GND. Pin 696 is connected to GND. Pin 697 is connected to GND. Pin 698 is connected to GND. Pin 699 is connected to GND. Pin 700 is connected to GND. Pin 701 is connected to GND. Pin 702 is connected to GND. Pin 703 is connected to GND. Pin 704 is connected to GND. Pin 705 is connected to GND. Pin 706 is connected to GND. Pin 707 is connected to GND. Pin 708 is connected to GND. Pin 709 is connected to GND. Pin 710 is connected to GND. Pin 711 is connected to GND. Pin 712 is connected to GND. Pin 713 is connected to GND. Pin 714 is connected to GND. Pin 715 is connected to GND. Pin 716 is connected to GND. Pin 717 is connected to GND. Pin 718 is connected to GND. Pin 719 is connected to GND. Pin 720 is connected to GND. Pin 721 is connected to GND. Pin 722 is connected to GND. Pin 723 is connected to GND. Pin 724 is connected to GND. Pin 725 is connected to GND. Pin 726 is connected to GND. Pin 727 is connected to GND. Pin 728 is connected to GND. Pin 729 is connected to GND. Pin 730 is connected to GND. Pin 731 is connected to GND. Pin 732 is connected to GND. Pin 733 is connected to GND. Pin 734 is connected to GND. Pin 735 is connected to GND. Pin 736 is connected to GND. Pin 737 is connected to GND. Pin 738 is connected to GND. Pin 739 is connected to GND. Pin 740 is connected to GND. Pin 741 is connected to GND. Pin 742 is connected to GND. Pin 743 is connected to GND. Pin 744 is connected to GND. Pin 745 is connected to GND. Pin 746 is connected to GND. Pin 747 is connected to GND. Pin 748 is connected to GND. Pin 749 is connected to GND. Pin 750 is connected to GND. Pin 751 is connected to GND. Pin 752 is connected to GND. Pin 753 is connected to GND. Pin 754 is connected to GND. Pin 755 is connected to GND. Pin 756 is connected to GND. Pin 757 is connected to GND. Pin 758 is connected to GND. Pin 759 is connected to GND. Pin 760 is connected to GND. Pin 761 is connected to GND. Pin 762 is connected to GND. Pin 763 is connected to GND. Pin 764 is connected to GND. Pin 765 is connected to GND. Pin 766 is connected to GND. Pin 767 is connected to GND. Pin 768 is connected to GND. Pin 769 is connected to GND. Pin 770 is connected to GND. Pin 771 is connected to GND. Pin 772 is connected to GND. Pin 773 is connected to GND. Pin 774 is connected to GND. Pin 775 is connected to GND. Pin 776 is connected to GND. Pin 777 is connected to GND. Pin 778 is connected to GND. Pin 779 is connected to GND. Pin 780 is connected to GND. Pin 781 is connected to GND. Pin 782 is connected to GND. Pin 783 is connected to GND. Pin 784 is connected to GND. Pin 785 is connected to GND. Pin 786 is connected to GND. Pin 787 is connected to GND. Pin 788 is connected to GND. Pin 789 is connected to GND. Pin 790 is connected to GND. Pin 791 is connected to GND. Pin 792 is connected to GND. Pin 793 is connected to GND. Pin 794 is connected to GND. Pin 795 is connected to GND. Pin 796 is connected to GND. Pin 797 is connected to GND. Pin 798 is connected to GND. Pin 799 is connected to GND. Pin 800 is connected to GND. Pin 801 is connected to GND. Pin 802 is connected to GND. Pin 803 is connected to GND. Pin 804 is connected to GND. Pin 805 is connected to GND. Pin 806 is connected to GND. Pin 807 is connected to GND. Pin 808 is connected to GND. Pin 809 is connected to GND. Pin 810 is connected to GND. Pin 811 is connected to GND. Pin 812 is connected to GND. Pin 813 is connected to GND. Pin 814 is connected to GND. Pin 815 is connected to GND. Pin 816 is connected to GND. Pin 817 is connected to GND. Pin 818 is connected to GND. Pin 819 is connected to GND. Pin 820 is connected to GND. Pin 821 is connected to GND. Pin 822 is connected to GND. Pin 823 is connected to GND. Pin 824 is connected to GND. Pin 825 is connected to GND. Pin 826 is connected to GND. Pin 827 is connected to GND. Pin 828 is connected to GND. Pin 829 is connected to GND. Pin 830 is connected to GND. Pin 831 is connected to GND. Pin 832 is connected to GND. Pin 833 is connected to GND. Pin 834 is connected to GND. Pin 835 is connected to GND. Pin 836 is connected to GND. Pin 837 is connected to GND. Pin 838 is connected to GND. Pin 839 is connected to GND. Pin 840 is connected to GND. Pin 841 is connected to GND. Pin 842 is connected to GND. Pin 843 is connected to GND. Pin 844 is connected to GND. Pin 845 is connected to GND. Pin 846 is connected to GND. Pin 847 is connected to GND. Pin 848 is connected to GND. Pin 849 is connected to GND. Pin 850 is connected to GND. Pin 851 is connected to GND. Pin 852 is connected to GND. Pin 853 is connected to GND. Pin 854 is connected to GND. Pin 855 is connected to GND. Pin 856 is connected to GND. Pin 857 is connected to GND. Pin 858 is connected to GND. Pin 859 is connected to GND. Pin 860 is connected to GND. Pin 861 is connected to GND. Pin 862 is connected to GND. Pin 863 is connected to GND. Pin 864 is connected to GND. Pin 865 is connected to GND. Pin 866 is connected to GND. Pin 867 is connected to GND. Pin 868 is connected to GND. Pin 869 is connected to GND. Pin 870 is connected to GND. Pin 871 is connected to GND. Pin 872 is connected to GND. Pin 873 is connected to GND. Pin 874 is connected to GND. Pin 875 is connected to GND. Pin 876 is connected to GND. Pin 877 is connected to GND. Pin 878 is connected to GND. Pin 879 is connected to GND. Pin 880 is connected to GND. Pin 881 is connected to GND. Pin 882 is connected to GND. Pin 883 is connected to GND. Pin 884 is connected to GND. Pin 885 is connected to GND. Pin 886 is connected to GND. Pin 887 is connected to GND. Pin 888 is connected to GND. Pin 889 is connected to GND. Pin 890 is connected to GND. Pin 891 is connected to GND. Pin 892 is connected to GND. Pin 893 is connected to GND. Pin 894 is connected to GND. Pin 895 is connected to GND. Pin 896 is connected to GND. Pin 897 is connected to GND. Pin 898 is connected to GND. Pin 899 is connected to GND. Pin 900 is connected to GND. Pin 901 is connected to GND. Pin 902 is connected to GND. Pin 903 is connected to GND. Pin 904 is connected to GND. Pin 905 is connected to GND. Pin 906 is connected to GND. Pin 907 is connected to GND. Pin 908 is connected to GND. Pin 909 is connected to GND. Pin 910 is connected to GND. Pin 911 is connected to GND. Pin 912 is connected to GND. Pin 913 is connected to GND. Pin 914 is connected to GND. Pin 915 is connected to GND. Pin 916 is connected to GND. Pin 917 is connected to GND. Pin 918 is connected to GND. Pin 919 is connected to GND. Pin 920 is connected to GND. Pin 921 is connected to GND. Pin 922 is connected to GND. Pin 923 is connected to GND. Pin 924 is connected to GND. Pin 925 is connected to GND. Pin 926 is connected to GND. Pin 927 is connected to GND. Pin 928 is connected to GND. Pin 929 is connected to GND. Pin 930 is connected to GND. Pin 931 is connected to GND. Pin 932 is connected to GND. Pin 933 is connected to GND. Pin 934 is connected to GND. Pin 935 is connected to GND. Pin 936 is connected to GND. Pin 937 is connected to GND. Pin 938 is connected to GND. Pin 939 is connected to GND. Pin 940 is connected to GND. Pin 941 is connected to GND. Pin 942 is connected to GND. Pin 943 is connected to GND. Pin 944 is connected to GND. Pin 945 is connected to GND. Pin 946 is connected to GND. Pin 947 is connected to GND. Pin 948 is connected to GND. Pin 949 is connected to GND. Pin 950 is connected to GND. Pin 951 is connected to GND. Pin 952 is connected to GND. Pin 953 is connected to GND. Pin 954 is connected to GND. Pin 955 is connected to GND. Pin 956 is connected to GND. Pin 957 is connected to GND. Pin 958 is connected to GND. Pin 959 is connected to GND. Pin 960 is connected to GND. Pin 961 is connected to GND. Pin 962 is connected to GND. Pin 963 is connected to GND. Pin 964 is connected to GND. Pin 965 is connected to GND. Pin 966 is connected to GND. Pin 967 is connected to GND. Pin 968 is connected to GND. Pin 969 is connected to GND. Pin 970 is connected to GND. Pin 971 is connected to GND. Pin 972 is connected to GND. Pin 973 is connected to GND. Pin 974 is connected to GND. Pin 975 is connected to GND. Pin 976 is connected to GND. Pin 977 is connected to GND. Pin 978 is connected to GND. Pin 979 is connected to GND. Pin 980 is connected to GND. Pin 981 is connected to GND. Pin 982 is connected to GND. Pin 983 is connected to GND. Pin 984 is connected to GND. Pin 985 is connected to GND. Pin 986 is connected to GND. Pin 987 is connected to GND. Pin 988 is connected to GND. Pin 989 is connected to GND. Pin 990 is connected to GND. Pin 991 is connected to GND. Pin 992 is connected to GND. Pin 993 is connected to GND. Pin 994 is connected to GND. Pin 995 is connected to GND. Pin 996 is connected to GND. Pin 997 is connected to GND. Pin 998 is connected to GND. Pin 999 is connected to GND. Pin 1000 is connected to GND.</p>

Count = 1
Count = 2
Count = 3
Count = 4
Count = 5
Count = 6
Count = 7
Count = 8
Count = 9

51

Code Example: USART – Loopback Test (Polling)

```
// Use Code Snippets 1 & 2
int main(void) {
    usart_init();
    while (1) { // Serial loopback test
        usart_putchar( usart_getchar() ); // Blocking calls
    }
}
```



Code Example: USART – TX Mode (Interrupt-Driven)

```
// Use Code Snippets 1 & 2

volatile const char *tx_buf = 0;
volatile uint8_t tx_busy = 0;

// Start TX using interrupt-driven method
void usart_send_string_async(const char *s) {
    tx_busy = 1;
    tx_buf = s;
    // Enable Data Register Empty interrupt
    UCSR0B |= (1 << UDRIE0);
}

// ISR for USART Data Register Empty
ISR(USART_UDRE_vect) {
    if (*tx_buf) {
        UDR0 = *tx_buf++; // Send next byte
    } else { // no more data to send
        tx_busy = 0;
        // Disable TX interrupt
        UCSR0B &= ~(1 << UDRIE0);
        tx_buf = 0;
    }
}
```

```
int main(void) {
    const char *msg = "Hello from ATmega328\r\n";
    usart_init();

    // Enable UDRE interrupt
    UCSR0B |= (1 << UDRIE0);
    // Enable global interrupts
    sei();

    while (1) {
        if (!tx_busy) {
            usart_send_string_async( msg );
            _delay_ms(100);
        }
    }
}
```

Code Example: USART – RX Buffer & RX Interrupt

```
// Use Code Snippets 1 & 2
#define UART_BUFSIZE (64)

volatile uint8_t rx_buf[UART_BUFSIZE];
volatile uint8_t rx_buf_tail = 0,
                rx_buf_head = 0;

ISR(USART_RX_vect) { // ISR for USART-RX
    // Read the received data byte
    uint8_t data = UDR0;
    // Increment the head pointer
    uint8_t next_head
        = (rx_buf_head + 1) % UART_BUFSIZE;
    // If the buffer is not full,
    // save data byte into buffer.
    if (next_head != rx_buf_tail) {
        rx_buf[rx_buf_head] = data;
        rx_buf_head = next_head;
    }
}
```

```
uint8_t is_uart_buffer_empty() {
    return (rx_buf_head == rx_buf_tail);
}

uint8_t uart_read(void) {
    if (is_uart_buffer_empty()) {
        return 0;
    }
    uint8_t data = rx_buf[rx_buf_tail];
    rx_buf_tail = (rx_buf_tail + 1) % UART_BUFSIZE;
    return data;
}
```

Code Example: USART – RX Buffer & RX Interrupt

```
int main(void) {
    usart_init();

    UCSR0B |= (1 << RXCIE0); // Enable USART RX interrupt
    sei();

    while (1) {
        if (!is_uart_buffer_empty()) {
            // Read one data byte from buffer and write it to UART
            usart_putchar( uart_read() );
        }
        _delay_ms(1);
    }
}
```

Code Example: C++ Class for Simple Serial

```
#ifndef SIMPLE_SERIAL_H
#define SIMPLE_SERIAL_H

#include <avr/io.h>
#include <avr/interrupt.h>
#include <stdint.h>

#define UART_BUFSIZE 64

class SimpleSerial {
public:
    SimpleSerial() {} // Default constructor
    void begin(uint32_t baud);
    void write(char c);
    void print(const char* s);
    char read();
    bool available();
};

#endif
```

SimpleSerial.h

Code Example: C++ Class for Simple Serial

```
#include "SimpleSerial.h"

static volatile uint8_t rx_buf[UART_BUFSIZE];
static volatile uint8_t rx_buf_head = 0;
static volatile uint8_t rx_buf_tail = 0;

void SimpleSerial::begin(uint32_t baud) {
    uint16_t ubrr_value = (F_CPU / (8UL * baud)) - 1;
    UBRR0H = (uint8_t)(ubrr_value >> 8);
    UBRR0L = (uint8_t)ubrr_value;
    UCSR0A = (1<<U2X0); // Double speed
    UCSR0C = (1<<UCSZ01) | (1<<UCSZ00); // 8N1
    // Enable the "RX Complete" interrupt
    UCSR0B = (1<<RXEN0) | (1<<TXEN0) | (1<<RXCIE0);
    sei(); // Enable global interrupts
}
void SimpleSerial::write(char c) {
    while (!(UCSR0A & (1 << UDRE0)));
    UDR0 = c;
}
```

SimpleSerial.cpp

Code Example: C++ Class for Simple Serial

```
void SimpleSerial::print(const char* s) {
    while (*s) { write(*s++); }
}

bool SimpleSerial::available() { // Check if the RX buffer has data
    return (rx_buf_head != rx_buf_tail);
}

char SimpleSerial::read() { // Pull data from the buffer
    if (rx_buf_head == rx_buf_tail) { return 0; }
    uint8_t data = rx_buf[rx_buf_tail];
    rx_buf_tail = (rx_buf_tail + 1) % UART_BUFSIZE;
    return (char)data;
}

ISR(USART_RX_vect) { // ISR for RX receive
    uint8_t data = UDR0;
    uint8_t next_head = (rx_buf_head + 1) % UART_BUFSIZE;
    if (next_head != rx_buf_tail) { // Check whether buffer not full
        rx_buf[rx_buf_head] = data;
        rx_buf_head = next_head;
    }
}
```

SimpleSerial.cpp

Code Example: C++ Class for Simple Serial

```
#include "SimpleSerial.h"

SimpleSerial serial; // Instantiate the object

int main(void) {
    serial.begin(115200);
    serial.print("C++ Serial Class Demo for AVR\r\n");
    while (1) {
        if (serial.available()) { // serial loopback
            serial.write( serial.read() );
        }
    }
    return 0;
}
```

main.cpp

Code Example: C++ Class for Simple Serial

The screenshot shows the Arduino IDE interface with the following details:

- Title Bar:** sketch_feb1a - main.cpp | Arduino IDE 2.3.7
- Menu Bar:** File Edit Sketch Tools Help
- Tool Buttons:** Save, Build, Upload, Select Board (set to Arduino UNO), Refresh, Help.
- Project Structure:** sketch_feb1a.ino, SimpleSerial.h, SimpleSerial.cpp, main.cpp
- Code Editor:** Displays the main.cpp file content:

```
1 #include "SimpleSerial.h"
2
3 SimpleSerial serial; // Instantiate the object
4
5 int main(void) {
6     serial.begin(115200);
7     serial.print("C++ Serial Class Demo for AVR\r\n");
8     while (1) {
9         if (serial.available()) { // loopback
10             serial.write( serial.read() );
11         }
12     }
13     return 0;
14
15 }
```
- Output Panel:** Shows the message "Hello C++ & AVR" received from the serial port.
- Serial Monitor:** Set to 'COM8' at 115200 baud. The message input field contains "Message (Enter to send message to 'Arduino UNO' on 'COM8')".
- Status Bar:** Ln 16, Col 1 | Arduino UNO on COM8 | 3 notifications.

In Arduino IDE, you can have multiple source code files:

- sketch.ino (empty)
- SimpleSerial.h
- SimpleSerial.cpp
- main.cpp