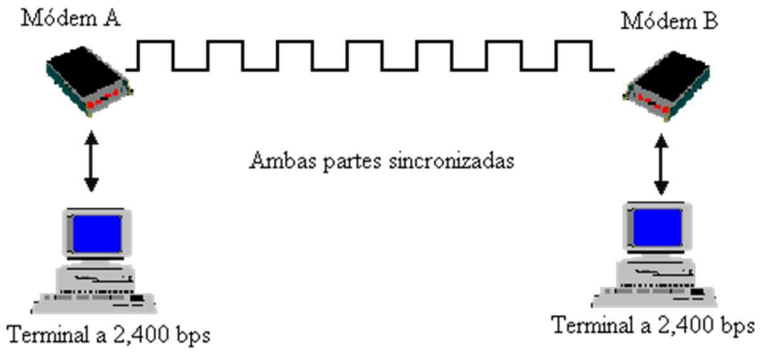
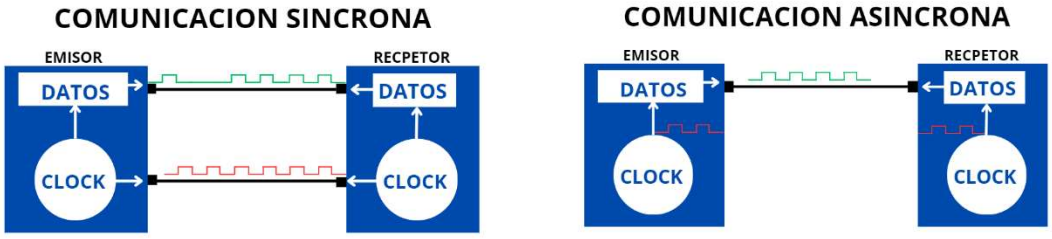
MODULO USART:

Un módulo USART (Universal Synchronous/Asynchronous Receiver/Transmitter) es un componente electrónico que se utiliza en microcontroladores y otros dispositivos electrónicos para facilitar la comunicación serie de datos.



Envió de datos serie entre dispositivos

Es capaz de enviar y recibir datos en serie de forma síncrona o asíncrona, lo que significa que puede trabajar con diferentes protocolos de comunicación, como UART, SPI y I2C. El módulo también puede configurarse para diferentes velocidades de transmisión de datos, desde velocidades muy lentas hasta velocidades muy rápidas.



En resumen, el módulo USART es una herramienta importante para la comunicación serial en dispositivos electrónicos, permitiendo la transmisión y recepción de datos en diferentes protocolos y velocidades de manera eficiente y confiable.

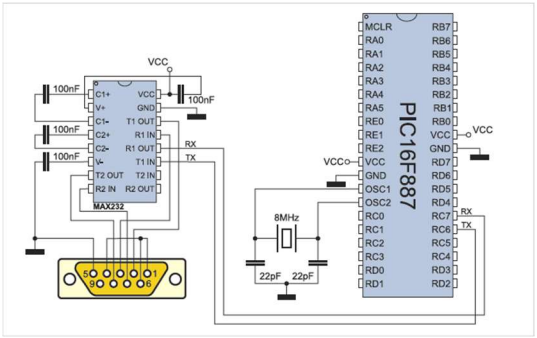
APLICACIONES DONDE SE UTILIZAN MODULOS UART

Los módulos UART proporcionan una forma fácil y económica de comunicación serial asíncrona entre dispositivos. Algunas de las aplicaciones más comunes de los módulos UART son:

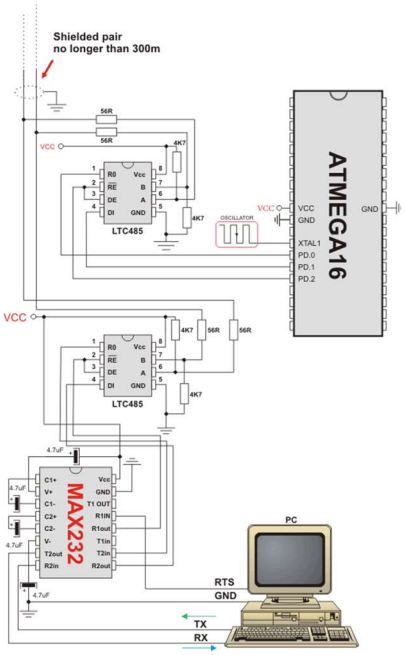
1. **Comunicación entre microcontroladores**: Los módulos UART se utilizan a menudo para establecer la comunicación entre dos o más microcontroladores. Esto permite la transferencia de datos entre ellos, lo que es útil en aplicaciones como la comunicación de control de motores, la comunicación de control de iluminación y la comunicación de sensores.

Comunicación entre dos microcontroladores

2. Transferencia de datos a través de cables: Los módulos UART se utilizan a menudo para transferir datos a través de cables, ya que la comunicación serial asíncrona es simple y robusta. Los módulos UART pueden utilizarse para la transferencia de datos a través de puertos serie RS-232, RS-485 o USB, lo que es útil en aplicaciones como la transferencia de datos entre sistemas de control industrial y la transferencia de datos de sensores a un ordenador.

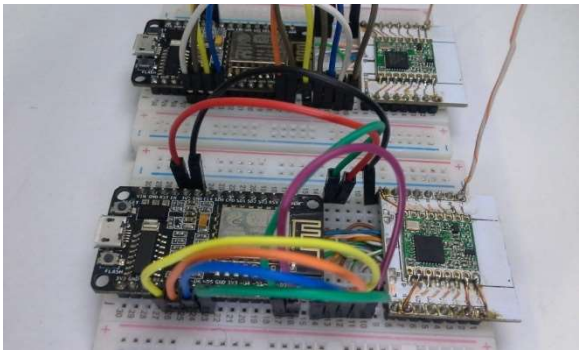


COMUNICACIÓN RS232



COMUNICACIÓN RS485

3. Comunicación inalámbrica: Los módulos UART también se utilizan a menudo para la comunicación inalámbrica entre dispositivos, mediante la transmisión de datos a través de módulos transceptores RF (radiofrecuencia). Los módulos UART pueden utilizarse para establecer comunicación inalámbrica entre dispositivos como sensores, actuadores, controladores y sistemas de automatización.



4. Comunicación con dispositivos periféricos: Los módulos UART se utilizan a menudo para establecer la comunicación con dispositivos periféricos como pantallas, teclados y ratones. Esto permite la transferencia de datos de entrada y salida entre el dispositivo periférico y el microcontrolador, lo que es útil en aplicaciones como la interfaz de usuario en sistemas embebidos.

Pantallas TFT

Teclado por UART

Mouse UART

5. Interconexión de dispositivos IoT: Los módulos UART también se utilizan a menudo para la interconexión de dispositivos de Internet de las cosas (IoT). Los módulos UART pueden utilizarse para la transferencia de datos entre sensores y actuadores en redes de IoT, lo que permite la monitorización y control remoto de sistemas.

MODULO GPRS

MODEM RS232

En resumen, los módulos UART son muy útiles para establecer comunicación serial asíncrona entre dispositivos, lo que los hace ampliamente utilizados en una gran variedad de aplicaciones, desde la transferencia de datos entre microcontroladores hasta la comunicación inalámbrica y la interconexión de dispositivos IoT.

Conceptos Básicos de los módulos UART

Los módulos UART (Universal Asynchronous Receiver/Transmitter) son dispositivos que permiten la comunicación serial asíncrona entre dos dispositivos electrónicos. A continuación, se detallan algunos de los conceptos básicos asociados a los módulos UART:

1) **Comunicación serial asíncrona**: se refiere a una forma de comunicación en la que los datos se transmiten sin un reloj común entre los dispositivos que se comunican. En la comunicación serial asíncrona, se utiliza un bit de inicio, un número fijo de bits de datos (generalmente 8 bits), un bit de paridad opcional y uno o más bits de parada para delimitar cada byte de datos transmitido.

2) **Velocidad de transmisión (Baudrate)**: se refiere a la tasa de transferencia de datos a través de la comunicación serial. Se mide en bits por segundo (baudios) y se puede ajustar para adaptarse a diferentes velocidades de procesamiento y requisitos de transferencia de datos.

3) **Bits de datos**: se refiere al número de bits que se utilizan para representar cada carácter transmitido. El número de bits de datos puede variar entre 5 y 9 bits.

4) **Bits de parada**: se refiere al número de bits que se utilizan para indicar el final de un carácter transmitido. Generalmente se utiliza un bit de parada, pero es posible utilizar más de un bit de parada.

5) **Bit de paridad**: se refiere a un bit adicional que se puede agregar a cada carácter transmitido para detectar errores de transmisión. El bit de paridad puede ser par, impar o no utilizado.

6) **Modos de transmisión y recepción**: los módulos UART pueden operar en modos de transmisión y recepción asíncrona, síncrona o de medio dúplex. En el modo de transmisión asíncrona, el módulo UART transmite datos de forma unidireccional. En el modo de recepción asíncrona, el módulo UART recibe datos de forma unidireccional. En el modo de transmisión y recepción síncrona, el módulo UART utiliza un reloj para sincronizar la transmisión y recepción de datos. En el modo de medio dúplex, el módulo UART puede transmitir y recibir datos, pero no al mismo tiempo.

Configuración de los módulos UART en el microcontrolador ATMEGA328PB.

El microcontrolador ATMEGA328PB cuenta con dos módulos USART, cada uno con sus propios registros que se encargan de controlar la configuración y operación de la comunicación serial asíncrona. A continuación, se detallan los principales registros de los módulos USART del ATMEGA328PB:

1) **UDRn**: este registro es el Registro de Datos del modulo USART y se utiliza para la transmisión y recepción de datos. Esto quiere decir, cuando se escribe un dato en este registro, el módulo USART lo transmite por el pin Tx. Cuando se recibe un dato por el pin Rx, se almacena en este registro.

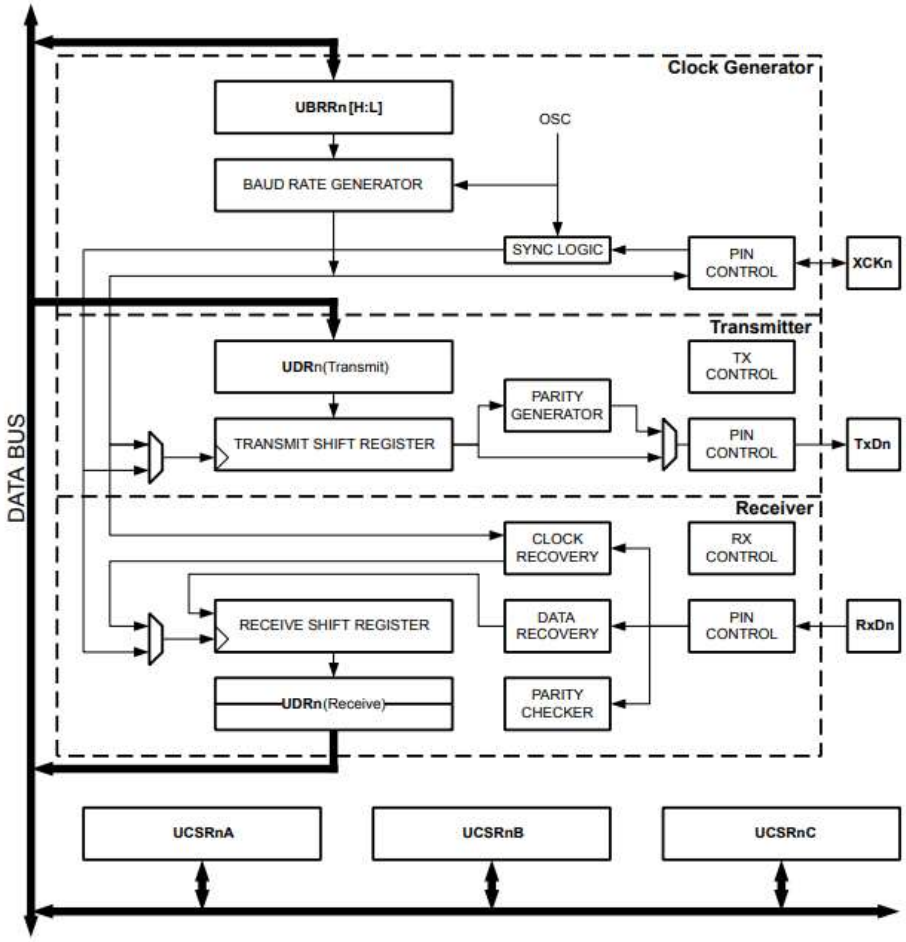
2) **UCSRA**: este registro es el Registro de Estado del USART A y contiene información sobre el estado actual de la transmisión/recepción. Incluye bits de estado como el indicador de transmisión completa, el indicador de recepción completa, el indicador de desbordamiento de recepción y el indicador de error de paridad.

3) **UCSRB**: este registro es el Registro de Control del USART B y se utiliza para configurar la operación del módulo USART. Incluye bits de control como el bit de habilitación de transmisión, el bit de habilitación de recepción, el bit de interrupción de transmisión completa y el bit de interrupción de recepción completa.

4) **UCSRC**: este registro es el Registro de Control del USART C y se utiliza para configurar los parámetros de la comunicación serial, como la velocidad de transmisión, el número de bits de datos, el número de bits de parada y la paridad.

5) **UBRRnH y UBRRnL**: estos registros son los Registros de Baudrate del USART y se utilizan para configurar la velocidad de transmisión. La velocidad de transmisión se calcula a partir de la frecuencia del reloj del sistema y el valor almacenado en estos registros.

Diagrama de bloques del modulo UART en el microcontrolador ATMEGA328PB



Es **importante** tener en cuenta que los nombres de los registros pueden variar dependiendo del módulo USART que se esté utilizando (USART0 o USART1). Además, los registros mencionados son solo una parte de los registros disponibles en los módulos USART del ATMEGA328PB. En la hoja de datos del microcontrolador se pueden encontrar todos los registros y sus detalles completos

Sistema de reloj del modulo UART del microcontrolador ATMEGA328PB

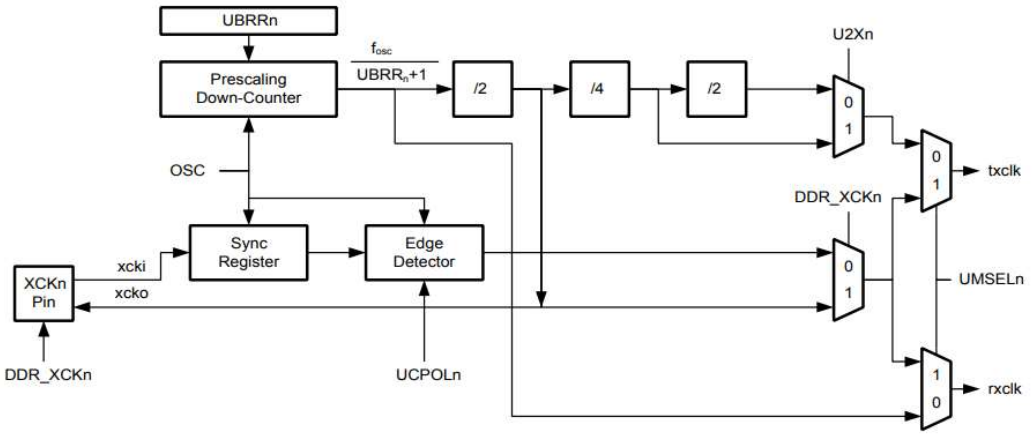
El módulo UART del microcontrolador ATmega328PB utiliza un sistema de generación de reloj a partir de la frecuencia del oscilador del sistema (System Clock). El oscilador del sistema puede ser configurado para trabajar a diferentes frecuencias utilizando los registros de configuración del microcontrolador.

El módulo UART utiliza el System Clock como base para generar el baud rate clock que determina la velocidad de transmisión de datos. Para lograr esto, el microcontrolador utiliza un divisor de frecuencia programable que se configura mediante los registros UBRRH y UBRRL. Estos registros definen el valor del divisor de frecuencia necesario para generar la velocidad de transmisión deseada.

Además, el módulo UART tiene un registro de control de velocidad de transmisión (UCSRC) que permite seleccionar la configuración de bits de datos, bits de parada, y bit de paridad, así como habilitar la generación de ***interrupciones de transmisión y recepción***.

Es importante destacar que la frecuencia del oscilador del sistema y el divisor de frecuencia programable deben ser configurados correctamente para lograr una transmisión y recepción de datos fiable y precisa. Además, el cálculo de la velocidad de transmisión y el valor del divisor de frecuencia requeridos para lograrla puede ser un proceso complejo, por lo que es recomendable utilizar herramientas de software que permitan calcular estos valores de manera automática.

Sistema generación de reloj del ATMEGA328PB



Signal description:

- txclk: Transmitter clock (internal signal).
- rxclk: Receiver base clock (internal signal).
- xcki: Input from XCKn pin (internal signal). Used for synchronous slave operation.
- xcko: Clock output to XCKn pin (internal signal). Used for synchronous master operation.
- f_{osc}: System clock frequency.

Generador de BAUDRATE

En el microcontrolador ATmega328PB, el baudrate se genera mediante un divisor de frecuencia programable. El divisor de frecuencia programable permite ajustar la velocidad de transmisión del UART al dividir la frecuencia del reloj del sistema (normalmente, la frecuencia del oscilador del microcontrolador) por un valor determinado.

La fórmula para calcular el valor del divisor de frecuencia programable es la siguiente:

$$Baudrate = Fosc / (16 * (UBRRn + 1))$$

donde:

- ***Baudrate*** es la velocidad de transmisión deseada en bits por segundo (bps)
- ***Fosc*** es la frecuencia del oscilador del sistema en Hz
- ***UBRRn*** es el valor del registro UBRRn, que es un registro de 16 bits que se utiliza para establecer el valor del divisor de frecuencia programable.

Es importante tener en cuenta que el valor del divisor de frecuencia programable debe ser ajustado para que el error de baudrate no supere el 2%.

El registro UBRRn se puede configurar de diferentes maneras dependiendo del modo de operación del UART.

Modo de Operación	Descripción	Calculo
Asincrono modo normal	En este modo, la transmisión de datos no está sincronizada con una señal de reloj externa, sino que se utiliza un baudrate predeterminado para sincronizar la transmisión y recepción de datos. El baudrate se configura mediante el divisor de frecuencia programable y se establece en función de la frecuencia del oscilador del sistema y el valor del divisor de frecuencia programable. La velocidad de transmisión puede variar según la longitud del cable, la carga y otros factores.	$UBRRn = \frac{f_{osc}}{16BAUD} - 1$
Asincrono doble velocidad	Permite una velocidad de transmisión de datos más rápida que el modo asíncrono normal. El baudrate se duplica con respecto al modo normal y se configura mediante el registro U2Xn (en el caso del microcontrolador ATMEGA328PB). Este modo es útil en aplicaciones que requieren una transmisión de datos rápida, pero puede ser menos confiable en entornos ruidosos.	$UBRRn = \frac{f_{osc}}{8BAUD} - 1$
Sincrono modo Maestro	Se utiliza en aplicaciones que requieren una transmisión de datos precisa y rápida. En este modo, la transmisión de datos está sincronizada con una señal de reloj externa, lo que permite una transmisión más rápida y precisa de datos. El baudrate se configura mediante el divisor de frecuencia programable y la frecuencia de la señal de reloj externa. El microcontrolador actúa como maestro y controla la velocidad y sincronización de la transmisión de datos. Este modo se utiliza en aplicaciones de comunicación de alta velocidad, como Ethernet y USB.	$UBRRn = \frac{f_{osc}}{2BAUD} - 1$

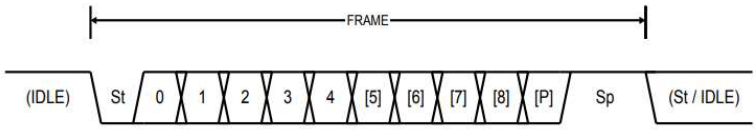
Formatos de trama

Un marco serial se define como un carácter (código ascii) de bits de datos con bits de sincronización (bits de inicio y parada), y opcionalmente un bit de paridad para la verificación de errores. El USART acepta las 30 combinaciones siguientes como formatos de marco válidos:

- 1 bit de inicio
- 5, 6, 7, 8 o 9 bits de datos
- sin bit de paridad, paridad par o paridad impar
- 1 o 2 bits de parada

Un marco comienza con el bit de inicio, seguido de los bits de datos (desde cinco hasta nueve bits de datos en total): primero el bit de datos menos significativo, luego los siguientes bits de datos hasta el bit más significativo. Si está habilitado, el bit de paridad se inserta después de los bits de datos, antes del uno o dos bits de parada. Cuando se transmite un marco completo, puede seguir directamente un nuevo marco o la línea de comunicación puede establecerse en un estado inactivo (alto). La figura a continuación ilustra las posibles combinaciones de los formatos de marco. Los bits dentro de corchetes son opcionales.

Frame Formats



St	Bit de Inicio (Bit start)
(n)	Bits de datos (0 a 8)
P	Bit de Paridad
Sp	Bit de stop
IDLE	Modo espera

El formato de marco utilizado por el USART se establece mediante:

- Los bits de tamaño de carácter (UCSRnC.UCSZn[2:0]) seleccionan el número de bits de datos en el marco.
- Los bits de modo de paridad (UCSRnC.UPMn[1:0]) habilitan y establecen el tipo de bit de paridad.

• El bit de selección de bits de parada (UCSRnC.USBSn) selecciona el número de bits de parada. El receptor ignora el segundo bit de parada.

El receptor y el transmisor usan la misma configuración. Tenga en cuenta que cambiar la configuración de cualquiera de estos bits corromperá toda la comunicación en curso tanto para el receptor como para el transmisor. Un error de marco (FE) solo se detectará en casos en los que el primer bit de parada sea cero.

Calculo del bit de paridad

En el ATmega328PB, el bit de paridad se puede configurar en modo par o impar o deshabilitado. Si se habilita el bit de paridad, el controlador de hardware del módulo USART calculará el bit de paridad antes de enviar los datos y verificará el bit de paridad después de recibir los datos.

Para calcular el bit de paridad, el módulo USART realiza una operación XOR en todos los bits de datos en el marco de transmisión. Si se selecciona el modo de paridad par, se establecerá el bit de paridad para que el número total de unos en el marco (datos más el bit de paridad) sea par. Si se selecciona el modo de paridad impar, se establecerá el bit de paridad para que el número total de unos en el marco sea impar.

Paridad Impar (Parity odd)

En este modo se asegura que el total de "1" enviados sea un numero impar de datos, esto quiere decir que:

D0	D1	D2	D3	D4	D5	D6	D7	P
1	1	1	0	0	0	1	0	1

← En este caso el bit de paridad es 1 para garantizar un numero impar de unos en la DATA a enviar

D0	D1	D2	D3	D4	D5	D6	D7	P
1	1	0	0	0	0	1	0	0

← En este caso el bit de paridad es 0 para garantizar un numero impar de unos en la DATA a enviar

Para determinar el valor del bit de paridad en $\boldsymbol{Parity\ odd}$ usamos la formula:

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

Paridad Par (Parity even)

En este modo se asegura que el total de "1" enviados sea un numero par de datos, esto quiere decir que:

D0	D1	D2	D3	D4	D5	D6	D7	P
1	1	1	0	0	0	1	0	0

← En este caso el bit de paridad es 0 para garantizar un numero par de unos en la DATA a enviar

iSebas

D0	D1	D2	D3	D4	D5	D6	D7	P
1	1	0	0	0	0	1	0	1

En este caso el bit de paridad es 1 para garantizar un numero par de unos en la DATA a enviar

Para determinar el valor del bit de paridad en ***Parity even*** usamos la formula:

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

Inicialización de un módulo UART para el microcontrolador ATMEGA328PB

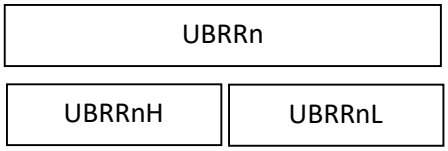
Para indicar una forma de como iniciar la configuración del modulo UART, vamos a realizar unos métodos utilizando sintaxis de lenguaje C. Pero antes de comenzar a crearlos debemos realizar los siguientes pasos:

Paso 1:

Calcular el valor UBRRn a partir de la formula

$$UBRRn = \frac{f_{osc}}{16BAUD} - 1$$

Se debe tener en cuenta que el registro $UBRRn$ esta en formato de 16 bits, eso implica que se dividan en dos registros:



Paso 2:

Configurar los pines correspondientes para Tx y Rx en modo UART mediante la configuración del registro DDRx.

Paso 3:

Configurar los registros de control de la UART correspondientes, en particular, el registro UCSR0A, UCSR0B y UCSR0C, que controlan la velocidad de transmisión, el modo de transmisión, la paridad, el número de bits de datos y el número de bits de parada.

Paso 4:

Para configurar la velocidad de transmisión en N baudios, primero se debe calcular el valor a cargar en el registro UBRR0. Con el valor obtenido en el PAS 1

Paso 5:

Una vez calculado el valor de UBRR0, se debe cargar este valor en los registros UBRR0H y UBRR0L, que controlan la velocidad de transmisión.

Paso 6:

Configurar el registro UCSR0C para seleccionar el número de bits de datos y el número de bits de parada. Para 8 bits de datos y 1 bit de parada, se debe configurar UCSR0C como sigue:

$$UCSR0C = (1 << UCSZ01) | (1 << UCSZ00)$$

Paso 7:

Configurar el registro UCSR0B para habilitar la transmisión (TXEN0) y la recepción (RXEN0):

$$UCSR0B = (1 \ll TXEN0) | (1 \ll RXEN0)$$

Ejemplo de método para inicializar un modulo UART en el microcontrolador ATMEGA328PB en forma de 8 bits con Tx y Rx habilitado.

```
void UART1_Init(unsigned long baudrate){
    // Calcula el valor del registro UBRR1 según el baudrate
    unsigned int ubrr_val = F_CPU / (16 * baudrate) - 1;

    // Configura el registro UBRR1 con el valor calculado
    UBRR0H = (unsigned char) (ubrr_val >> 8);
    UBRR0L = (unsigned char) ubrr_val;

    // Habilita la transmisión y recepción de datos en 8 bits
    UCSR0B = (1 << RXEN0) | (1 << TXEN0);
    UCSR0C = (1 << UCSZ01) | (1 << UCSZ00);
}
```

Ejemplo de método para enviar datos por modulo UART en el microcontrolador ATMEGA328PB

Aquí te presento un ejemplo de un método en lenguaje C para enviar datos a través del módulo UART en el microcontrolador ATmega328PB:

```
void UART1_Transmit_char(unsigned char data){
    // Espera a que el registro de transmisión esté vacío
    while (!(UCSR0A & (1 << UDRE0)));

    // Carga el dato en el registro de transmisión
    UDR0 = data;

}
```

Este método envía un byte de datos a través del puerto serie. Primero espera a que el registro de transmisión esté vacío, lo que indica que el byte anterior ha sido transmitido completamente. Luego, carga el byte de datos en el registro de transmisión y la transmisión comienza automáticamente.

Para enviar una cadena de caracteres, se puede utilizar un bucle que recorra todos los caracteres de la cadena y llame al método "USART_Transmit_char" para enviar cada uno de ellos.

Ejemplo de un método para recibir datos por modulo UART en el microcontrolador ATMEGA328PB

A continuación te muestro un ejemplo de cómo podrías implementar un método para recibir datos a través del módulo UART en el microcontrolador ATmega328PB:

```
unsigned char UART1_Receive(void){
    while (!(UCSR0A & (1<<RXC0))); // Espera hasta que se reciba un byte
completo
    return UDR0; // Retorna el byte recibido
}
```

En este método, la función UART1_Receive espera hasta que se recibe un byte completo, lo que ocurre cuando el bit RXC1 del registro UCSR1A se activa. Luego, se retorna el byte recibido a través del registro UDR1. Es importante mencionar que el método espera de forma bloqueante, lo que significa que el programa quedará detenido hasta que se reciba un byte completo.