Introducción al Tema:

La interfaz UART (Universal Asynchronous Receiver Transmitter) es un protocolo de comunicación serial ampliamente utilizado en sistemas embebidos para la transmisión y recepción de datos entre microcontroladores y dispositivos externos. En el caso de las placas STM32, como la STM32-NUCLEO-F429ZI, UART es gestionada a través de periféricos USART (Universal Synchronous/Asynchronous Receiver Transmitter), los cuales ofrecen flexibilidad para trabajar en aplicaciones de comunicación tanto síncronas como asíncronas.

En la biblioteca HAL (Hardware Abstraction Layer) de STMicroelectronics, UART se abstrae mediante estructuras y funciones específicas que facilitan su configuración y uso. STM32CubeIDE, como entorno de desarrollo, permite configurar los periféricos UART de manera gráfica y generar automáticamente el código necesario para inicializarlos.

¿Qué es una UART?

La UART es una interfaz de comunicación serial que funciona de manera asíncrona, es decir, no requiere de una señal de reloj compartida entre los dispositivos que se comunican. En su lugar, utiliza dos líneas principales:

TX (Transmisión): Línea para enviar datos. RX
(Recepción): Línea para recibir datos.

Componentes y Configuración de UART

Para que la UART funcione correctamente, es necesario configurar varios parámetros esenciales:

Velocidad de transmisión (BaudRate): Define la cantidad de bits transmitidos por segundo. Los valores más comunes son 9600, 19200, 38400, entre otros. La selección del BaudRate depende de la aplicación y de las capacidades del hardware.

Longitud de palabra (WordLength): Especifica el número de bits de datos en cada trama. Los valores típicos son 8 o 9 bits.

Bits de parada (StopBits): Indican el final de una trama. Se puede utilizar 1 o 2 bits de parada según el protocolo de comunicación.

Paridad (Parity): Es una forma simple de detección de errores.
Puede ser de tres tipos:Ninguna (No Parity, Par (Even Parity), Impar (Odd Parity)

Modo de operación (Mode): Determina si la UART está habilitada para transmisión (TX), recepción (RX) o ambas.

Control de flujo de hardware (HwFlowCtl): Permite gestionar automáticamente el flujo de datos mediante señales adicionales (RTS/CTS), lo cual es útil para evitar la pérdida de datos en comunicaciones de alta velocidad.

Sobremuestreo (OverSampling): Mejora la precisión de la recepción de datos discriminando entre señales válidas y ruido. Puede configurarse en 8 o 16 muestras por bit.

Implementación en el IDE STM32CubeIDE

Estructuras Principales

En STM32CubeIDE, la configuración de UART se basa en dos estructuras principales definidas en la HAL:

UART_HandleTypeDef:

Esta estructura representa el periférico UART y contiene información sobre su configuración actual. Incluye los siguientes campos importantes:

Instance: Puntero al descriptor del periférico USART/UART (por ejemplo, USART2).

Init: Configuración básica de UART, almacenada en una subestructura del tipo UART_InitTypeDef. **pTxBuffPtr y pRxBuffPtr:** Punteros al búfer de transmisión y recepción, respectivamente.

UART_InitTypeDef:

Esta subestructura, contenida dentro de UART_HandleTypeDef, define los parámetros básicos de configuración de UART:

BaudRate

WordLength

StopBits

Parity

Mode

HwFlowCtl

OverSampling

Proceso de Configuración

El proceso de configuración de UART en STM32CubeIDE incluye los siguientes pasos:

Selección del Periférico:

En el entorno de STM32CubeMX (integrado dentro de STM32CubeIDE), se selecciona el periférico USART/UART deseado y se asignan los pines correspondientes

Configuración de Parámetros:

Se ajustan los parámetros básicos, como BaudRate, WordLength, StopBits, etc., según los requerimientos de la aplicación.

Generación de Código:

STM32CubeIDE genera automáticamente el código de inicialización del periférico, incluyendo la configuración de pines GPIO y la habilitación del reloj del periférico.

Inicialización de UART:

La función HAL_UART_Init() asegura que el periférico UART se configure según los parámetros especificados en UART_HandleTypeDef.

Transmisión y Recepción de Datos

Una vez configurado el periférico UART, se pueden utilizar las funciones de la HAL para transmitir y recibir datos. Estas funciones son:

HAL_UART_Transmit: Envía un arreglo de datos a través del periférico UART.

HAL_UART_Receive: Recibe datos entrantes y los almacena en un búfer.

Autor de la modificación:

APLICACIÓN 1.1: Nahuel Beltrán

Tuve problemas al configurar correctamente los tiempos de espera (timeouts) en funciones como HAL_UART_Transmit o HAL_UART_Receive, podrían ocurrir errores de comunicación..

APLICACIÓN 1.2: Jose Gareca

Tuve problemas al implementar UART porque no verifique correctamente cuándo el periférico está listo para transmitir o recibir datos.

APLICACIÓN 1.3: Nahuel Sotelo

Tuve problemas para controlar adecuadamente el evento de transmisión en uart lo cual causo que se enviaban o procesaban datos varias veces

APLICACIÓN 1.4: Daniel Aragón

Tuve dificultad para distinguir correctamente los eventos de transmisión completada (TX Complete) y recepción completada (RX Complete) lo cual causaba errores, como perder datos recibidos o enviar información en momentos incorrectos.

Link al repositorio:

https://github.com/NahuelFSotelo/Grupo-1-TDII-AFP_2024.git