Unidad 13 - Puertos

1. Puertos de la Computadora

Un puerto es una **interfaz** a través de la que se puede enviar y recibir diferentes tipos de datos, disponible para la conexión de periférico como dispositivos de entrada y salida.
Un **puerto lógico** se denomina a la zona o localización de la memoria, asociada a un **puerto físico**, que proporciona un espacio de almacenamiento temporal entre la memoria y el canal de comunicación para información a transferir.

Del lado del hardware, es una salida especializada a la que se conecta un dispositivo, proporcionando un método y un medio para **transferir señales**.
Los puertes pueden clasificarse según el **tipo de transferencia**:

- **Puerto Serie**: envían y reciben **un bit** a la vez.
- **Puerto Paralelo**: envían y reciben **varios bits** a través de conjuntos de cables.

Para utilizar los puertos se requiere de un **protocolo de enlace**, que determina el tipo de transferencia, la velocidad de transferencia y otra información necesaria.

1.1. Puerto Paralelo

Su principal característica es que los bits de datos viajan juntos, enviando un **paquete de bytes** a la vez. Es decir que se implementa una vía física para cada bit, formando un bus. Además de enviar y recibir bits de datos, también transmite bits de control y estado.

- **Registro de Datos**: almacena los datos a transferir, y puede ser tanto leído como escrito.
- **Registro de Estado**: es de solo lectura y permite a la computadora comprobar el estado del dispositivo. Está compuesto por
  - Ack: indica que se han recibido datos en el dispositivo y está listo para recibir más datos.
  - Busy: indica que está ocupado.
  - Paper End: indica que no hay papel (???).
  - Select: en alto para seleccionar dispositivo (???). En bajo indica que el dispositivo ha sido seleccionado.
  - Error: indica la ocurrencia de un error en el dispositivo.
  - IRQ: indica una solicitud de interrupción.
- **Registro de Control**: permite gestionar el comportamiento del dispositivo, además de permitir al dispositivo generar interrupciones para solicitar atención.
  - Strobe: indica al dispositivo que puede recibir los datos enviados.
  - AutoFeed: si está bajo, el papel se mueve una línea tras la impresión (???).
  - Init: permite reiniciar el dispositivo.
  - Select: en bajo indica que el dispositivo ha a sido seleccionado (???).
  - IRQ Enable: indica que se pueden utilizar las interrupciones del dispositivo.
  - Bidirec: indica que el puerto bidireccional está habilitado.

1.1.1. Protocolo de Comunicación

Un puerto paralelo toma un ciclo ISA para leer o escribir. Algunos puertos soportan un modo "turbo" que elimina los estados de espera de la CPU, con lo que se duplica la velocidad de lectura/escritura.
Un protocolo para transmitir un byte a una **impresora** utiliza principalmente *strobe*, *ack* y *busy*, y viene dado por:

1. Esperar que el dispositivo no esté ocupado (`busy = 0`).
2. Activar strobe (`strobe = 0`) para que el dispositivo acepte el dato.
3. Introducir el byte en el *registro de datos*.
4. La impresora activa busy (`busy = 1`) para indicar que está procesando el dato.
5. La impresora activa ack (`ack = 0`) para indicar que ha terminado y se puede regresar al paso 1

El protocolo para **intercambiar datos** entre dos computadoras usando el puerto paralelo es el siguiente:

1. La computadora emisora activa DSL para indicar que quiere recibir datos.
2. El receptor contesta activando INI.
3. El emisor carga el dato y realiza un pulso de STB, generando una interrupción en la receptora. La rutina de gestión de la interrupción en el receptor adquiere el dato.
4. El receptor realiza un pulso de STB, generando una interrupción en el emisor.
5. La interrupción en el emisor devuelve el control a la rutina, que repetirá los pasos 3, 4 y 5 hasta que se transmita el último dato.
6. El transmisor desactiva DSL.
7. El receptor desactiva INI.

1.1.2. Estandar IEEE 1284

Basicamente define cinco modos de transferencia: Modo de Compatibilidad, Modo Nibble, Modo Byte, Modo Mejorado, Modo Capacidades Extendidas.

1.2. Puerto Serie

El **puerto serie** es una interfaz de comunicación de datos digitales donde los datos son transmitidos **un bit a la vez**. Teóricamente es más lento que un puerto paralelo y los datos deber ser preprocesador (serializados/deserializados), pero es más flexible, menos costoso y una
menor tendencia a sufrir errores, incluso en distancias largas.

El proceso de transmisión serial toma datos organizados en bytes y los transmite como una secuencia de bits, mientras que en el destino esos bits deben reorganizarse en bytes. Cada unidad de transmisión debe contener un registro de desplazamiento para la conversión paralelo-serie.

La comunicación se puede clasificar en:

- Simplex: en una sola dirección, sin posibilidad de que el receptor envía información de vuelta al transmisor.
- Full-Duplex: ambos dispositivos reciben y transmiten al mismo tiempo.
- Half-Duplex: cada dispositivo se turno para transmitir y recibir.

Dado que las señales de control para un puerto serie pueden activarse y desactivarse fácilmente mediante un interruptor, en algunos casos se utilizan para monitores dispositivos externos, sin intercambiar datos.

1.2.1. Transmisión Síncrona

En este tipo de transferencias, se intercambian una o varias señales de control, junto con los paquetes de datos, entre el emisor y receptor. Las señales de control determinan cuándo hay un bit de datos válido en la lines de transmisión (es decir, sincroniza). La transmición es controlada por el emisor, y los datos se transmiten entre dos grupos de datos llamados *delimitadores*.
Ventajas: alto rendimiento, altas velocidades y un flujo de datos regular.
Desventajas: equipos más complejos y costosos.

Existen dos opciones para controlar la sincronización:

- Una señal de **reloj** que controle la sincronización.
- La sincronización se hace a través de un protocolo de solicitud, como el reconocimiento de estructuras de información predeterminadas.

1.2.2. Transmisión Asíncrona

Cuando no hay relación *temporal* entre los relojes del transmisor y el receptor, se debe "sincronizar" con cada dato transmitido a través de estructuras. La ventaja de esto es que el ritmo con el que transmiten los datos no tiene por que coincidir con el ritmo con el que se reciben, por lo que tampoco es necesario garantizar un ancho de banda determinado.
Por ello también es necesario que los datos **contengan la información de temporización** y el receptor muestre la linea a **intervalos regulares y mayores al pulso de reloj**, para detectarlos.
Para su implementación se deben acordar los parámetros de la señal:

- Tipo de operación a realizar, simplex, full-duplex o half-duplex.
- Tamaño del dato (bits por carácter) y orden en que se envían.
- Velocidad de transmisión.
- Código de detección-corrección de errores utilizado.
- Estructura de la información de sincronización.

La transmisión se realiza en bloques de datos (SDU):

1. Un bit de arranque (Start) que indica el inicio del SDU.
2. Los bits del SDU de forma secuencial.
3. Uno o dos bits de parada (Stop) que indican el fin del SDU.

Transmisión: dado un dato, se organiza el paquete de datos (SDU) con la información necesaria (bit de arranca, paridad, dato y bit de parada). El bloque de datos generado se almacena en un buffer que alimenta un registro de desplazamiento, que actua de acuerdo a una señal de reloj.
Recepción: al recibir el bit de arranque, un registro de desplazamiento recompone el dato a partir de los bits recibidos siguiendo la señal de reloj. Una vez se tiene el dato completo, se comprueba que el SDU no contenga errores y se transfiere al buffer.

1.2.2.1. Errores

Durante la transmisión pueden darse diversos errores:

- Error de **estructura** (frame): si el receptor detecta un bit de final incorrecto, entonces el dato recibido con se ajusta a la estructura pactada.
- Error de **rotura** (break): si el receptor detecta que la señal está a un valor distinto del por defecto durante más tiempo que de la duración de un bloque de datos, entonces la conexión se ha roto.

- Error de **sobreescritura** (overrun): si los datos llegan demasiado rápido, un bloque de datos recompuesto pero no leído puede ser sobreescrito por uno posterior.
- Error de **paridad** (parity): si no se detecta ninguno de los errores anteriores, el bloque ha llegado correctamente, pero es necesario comprobar el bloque mismo.

1.2.3. USART

Un Receptor/Transmisor Síncrono/Asíncrono Universar (**Universal Synchronous/Asynchronous Receiver/Transmitter** - USART) es un **emisor/receptor** programable cuyas funciones son la conversión paralelo-serie/serie-paralelo, la generación del reloj para transmisión/recepción de datos, y la generación y validación de los protocolos de transmisión. Al ser programable, el formato de datos y velocidades de transmisión son configurables.

La **transmisión** por parte de la USART toma el dato almacenado en el registro o FIFO, lo organiza junto con la información adicional (bits de comienzo, paridad y fin) y utiliza un registro de desplazamiento para serializar el dato. La velocidad de emisión condiciona el control de recepción del otro lado de la conexión, y viceversa.
La **recepción** toma el dato en el registro de emisión (?) y lo transfiere al registro o FIFO, paraleliza el dato de recepción utilizando un registro de desplazamiento y detecta y extrae los bit de inicio, fin y paridad.
Para la **sincronización** del dato recibido se debe comprobar la señal en la mitad del intervalo de tiempo para evitar la lectura de falsas transiciones producto del ruido. Para esto se utiliza un reloj externo de periodo `TC` que cumple la relación `TD = K * TC`, donde `TD` es el periodo de un bit transmitido y `K` el número de periodos `TC` en un periodo `TD`. Para que el bit de datos se compruebe en el momento preciso sin necesidad de conectar una línea adicional de reloj, estos valores deben ser iguales para tanto el receptor como el emisor.

1.3. Puerto Firewire

Es una interfaz estándar para puertos serie para la transferencia **isócrona** y en **tiempo real**. Fue diseñado para obtener un alto desempeño, que obtiene a partir de técnicas de bajo nivel como **acceso directo a memoria**.

Debido a que el tráfico de información en sistema de tiempo real es muy sensible al tiempo, el protocolo FireWire dispone de dos modos de transmisión:

- Modo **asíncrono**: se utiliza para comunicar dispositivos que no necesitan anchos de banda elevados. Se envían cantidades arbitrarias de datos e información de la capa de transacción al nodo destino, segudo por una confirmación del nodo.
- Modo **Isócrono**: garantiza tasas de transferencia predeterminadas para cada dispositivo sin sufrir interrupciones en el flujo de datos. Se envía una cantidad arbitraria de datos a intervalos regulares a un nodo destino y sin requerir confirmación.

E protocolo se basa en tres capas:

- Capa de **Transacciones**: maneja las transferencias a través del bus. Este bus utiliza 64 bits de direccionamiento, donde los 16 más significativos identifican al dispositivo. 10 bits identifican el bus y 6 bit de desplazamiento.
- Capa de **enlace**: gestiona la transferencia de información, que se divide en tres etapas:
    - Arbitraje: un dispositivo envía una petición de acceso al bus de la capa física.
    - Transmisión: el dispositivo transmite un paquete de datos con información del formato, la transacción, dirección de los dispositivos fuente y destino y los datos.
    - Reconocimiento: se envía un código de confirmación indicando que los datos fueron recibidos correctamente.
- Capa física: convierte los niveles lógicos de la capa de enlace en señales eléctricas para el cable y viceversa.

1.4. Universal Serial Bus

El Bus **Universal en Serie** (Universal Serial Bus - USB) es un bus que define los cables, conectores y protocolos usados en un bus para conectar, comuncar y alimentar entre computadoras, preiféricos y dispositivos electrónicos. Entre sus propiedades se encuentra:

- Menor número de conexiones necesarias.
- Configuración más sencilla, permitiendo conexión en caliente, identificando y configurando automáticamente los dispositivos conectados.
- Menor cantidad de recursos necesarios para gestionar los dispositivos.
- Permite un mayor número de dispositivos conectados.
- Permite la ampliación del sistema al permitir conectar distintos tipos de dispositivos con diferentes velocidades de transferencia y proporcionando alimentación

Es un bus desarrollado para simplificar y mejorar la **interfaz** entre las computadores y los periféricos. Es utilizado como estándar de conexión de periféricos y han ido reemplazando cada vez más otros tipos de cargadores de batería de dispositivos portátiles.

Un sistema USB consiste de un "host" con uno o más puertos y múltiples periféricos, formando una topología de estrella. "Hubs" USB adicionales pueden incluirse, hasta cinco niveles del host. Un host puede tener múltiples controladores, cada uno con uno o más puertos.

La gestión del bus es centralizada en el controlador integrado en la computadora (host), que inicia todas las actividades y se comunica con la computadora por medio de **interrupciones**. Ningún dispositivo puede inciar una transacción por sí mismo, de esta forma se evita sobreescribir datos presentes en el bus.

1.4.1. Hub

El hub es un dispositivo que proporciona un puerto de conexión con el host (upstream) y varios puertos de conexión con otros dispositivos o hubs (downstream). Su principal función es extender el sistema, proporcionando nuevos puertos de conexión.
Otras funciones del hub son:

- Repetidor: distribuir el tráfico entre el host y los dispositivos.
- Controlador: regular el tráfico de datos de acuerdo con los dispositivos conectados.

1.4.2. Comunicación

El cable usb consta de un par trenzado que utliizan una señal diferencial (¿?) en **half-dublex**. El USB 3.0 utiliza un segundo pare de hilos, también con señal diferencial, para realizar una comuniación en **full-duplex**. La **sincronización** se logra a partir de los propios datos utilizando codificación Manchester.