9 Apéndice

A Cyclic redundancy check

La base de la revisión por redundancia cíclica (CRC por sus siglas en inglés), se encuentra en la división de enteros, la cual se encuentra formada por un dividendo, un divisor y un residuo. El proceso de transmisión de un mensaje con CRC consta de un mensaje (dividendo) y la CRC (residuo), por lo tanto la transmisión puede ser verificada si se vuelve a calcular la división del mensaje y los residuos coinciden. Otra manera de verificar el mensaje, siendo ésta la más común, es restarle el residuo al mensaje y verificar si el resultado el residuo es cero.

El concepto de división puede ser también aplicado a los polinomios, ya que el CRC trata al mensaje como un polinomio, por ejemplo, el mensaje 11001001 daría como polinomio el $x^7+x^6+x^3+1$, pero para que la CRC pueda ser posible el transmisor y el receptor deben estar en concordancia con el divisor que se va a utilizar para el mensaje.

La elección de este divisor no es fácil ya que dependiendo de las características de este dependerá la calidad de detección de los errores de transmisión. Una particularidad del divisor es que el residuo de la división será siempre menor al orden del divisor, por lo tanto, el número de *bits* a utilizar para la CRC será el orden de su polinomio divisor.

Para el protocolo USB existen dos divisores:

- El CRC5, utilizado para los paquetes tipo *token*[1], consta de 5 bits, siendo su polinomio x^5+x^2+1.
- El CRC16, que es utilizado para los paquetes de datos, consta de 16 bits, siendo su polinomio el $x^{16}+x^{15}+x^2+1$.

Un problema del CRC es que al enfrentar un mensaje que contenga únicamente "ceros" éste sería tomado como correcto. El problema se supera aplicando las siguientes medidas:

- El registro es cargado con "unos" antes de iniciar la operación, esto es similar a sumar una constante al dividendo. Al no aplicar esta instrucción los primeros "ceros" no serían protegidos por el algoritmo.
- El residuo es invertido antes de ser anexado al mensaje, esto es similar a sumar una constante al residuo. Sin emplear esta medida el algoritmo no protegería los últimos "ceros" del mensaje.

  Estas medidas deben ser proporcionadas al transmisor y al divisor para ser efectivas.

[1] Token: tipo de paquete en el protocolo USB (véase la sección 2.3.1.2)

B Datos calibración sensor ventilación

Flujo [ml/min]	Voltaje [V]	Flujo [ml/min]	Voltaje [V]	Flujo [ml/min]	Voltaje [V]
0	2.504	0	2.503	320	2.524
1243	2.539	1450	2.593	1300	2.533
1465	2.542	1807	2.597	1620	2.539
1640	2.547	2323	2.606	1810	2.543
1852	2.55	3356	2.623	2090	2.546
1945	2.551	4093	2.633	2310	2.55
2131	2.552	4362	2.636	2480	2.552
2391	2.557	5031	2.646	2780	2.561
2619	2.56	5270	2.647	3500	2.561
2804	2.561	5867	2.657	3150	2.564
3120	2.565	7007	2.675	3380	2.564
3520	2.569	7443	2.684	3740	2.57
2817	2.574	7923	2.686	4130	2.576
4050	2.579	8629	2.7	4370	2.581
4715	2.587	9368	2.71	4600	2.584
5182	2.594	14211	2.726	4840	2.591
5745	2.6	15535	2.74	5010	2.593
6360	2.612	15776	2.744	5460	2.598
6678	2.618	16591	2.758	5730	2.602
6945	2.62	17011	2.763	5980	2.606
7737	2.635	17351	2.766	6210	2.609
8402	2.647	18360	2.779	6400	2.613
8837	2.649	18968	2.79	6700	2.617
9210	2.659	20310	2.808	6850	2.618
9868	2.67	20910	2.814	7130	2.625
10420	2.677	21990	2.829	7340	2.626
11055	2.691	22910	2.837	8070	2.636
11446	2.691	23710	2.845	8270	2.642
12513	2.709	24621	2.858	8820	2.649
13619	2.724	26483	2.881	8990	2.65
14725	2.74	28664	2.903	9510	2.66
15166	2.746	30476	2.919	10100	2.666
15943	2.758	33483	2.955	10800	2.676
16299	2.759	35885	2.98	11500	2.69
17169	2.768	39736	3.02	12300	2.702
18113	2.78	40836	3.031	13100	2.702
18916	2.791	43422	3.059	14000	2.725
20201	2.809	44873	3.069	14800	2.735
21195	2.82	45206	3.077	15400	2.745
22230	2.83	46333	3.081	16700	2.763
23100	2.845	47127	3.09	17900	2.777

25235	2.869	47753	3.094	18600	2.784
29173	2.91	49093	3.109	19600	2.796
33200	2.956	50775	3.126	20800	2.803
39742	3.029	51547	3.133	21700	2.824
41990	3.05	52703	3.14	23700	2.824
46370	3.085	54545	3.156	23400	2.84
49014	3.115	57223	3.188	24700	2.855
54419	3.169	59069	3.196	27000	2.882
60198	3.216	61036	3.21	37100	2.995
65729	3.263	64166	3.239	46200	3.082
69025	3.286	67761	3.27	49000	3.106
72194	3.311	70456	3.294	52400	3.142
77507	3.355	73314	3.314	56700	3.181
81274	3.399	76572	3.339	60300	3.211
84953	3.41	78391	3.352	64900	3.248
87275	3.424	79916	3.364	68400	3.27
89548	3.443	81220	3.371	73100	3.313
93179	3.469	83304	3.388	78000	3.355
96301	3.489	86721	3.412	80200	3.381
98250	3.503	89871	3.437	86800	3.417
102182	3.535	91934	3.454	91100	3.442
104005	3.545	94034	3.469	93600	3.457
105229	3.556	95097	3.471	97400	3.492
106043	3.561	96646	3.486	99400	3.501
106585	3.563	97606	3.492	101400	3.523
		98560	3.5	104700	3.542
		100165	3.512	106200	3.554
		101424	3.517	107900	3.564
		102304	3.524		
		103.369	3.533		
		104.407	3.539		
		105.602	3.557		
		106.464	3.555		

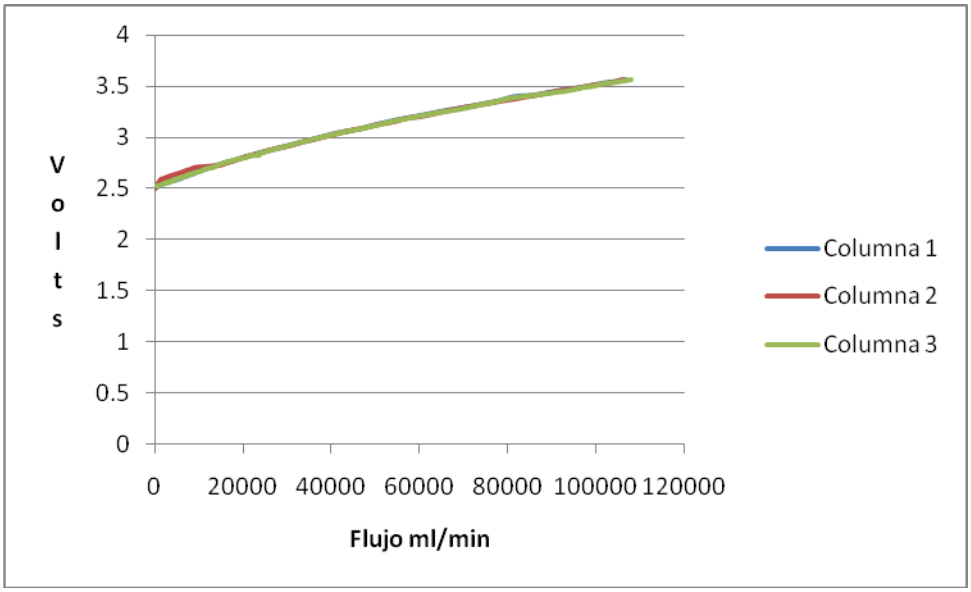


Figura B.1. Gráfica de voltaje vs Flujo.

C Modelado cardiovascular

La simulación del sistema cardiovascular se basa en el flujo de sangre en los vasos sanguíneos provocado por cambios de presión torácica. La simulación se encuentra basada en el trabajo hecho por Babbs[2], en el cual planteó que el sistema está formado por cámaras que se encuentran interconectadas por vasos sanguíneos por medio de una resistencia. Para este proyecto se plantearon siete cámaras, como se muestra en la figura C.1.

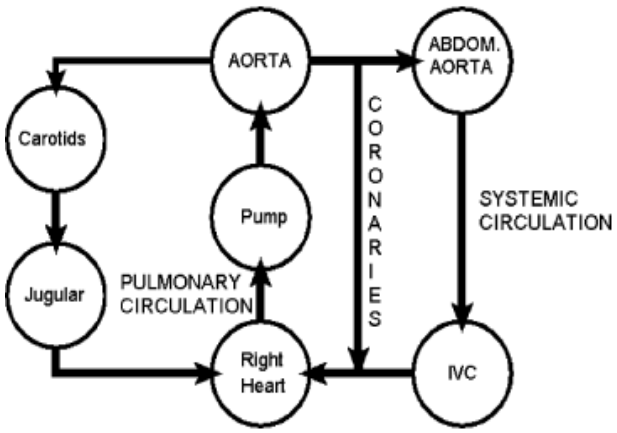


Figura C.1. Modelo del sistema circulatorio humano.

[2] Babbs Charles, "CPR techniques that combine chest and abdominal compression and decompression: hemodynamic insights from a spreandsheet model", Circulation, vol 100, No. 21, nov, 1999, USA, p. 2146-2152.

Se planea que cada cámara se pueda expandir, incrementando su volumen y a su vez la presión interna de la misma, lo que nos lleva a definir la complianza de la cámara:

$$C = \frac{\Delta V_{\text{olumen}}}{\Delta P_{\text{resión}}}$$

La elevación de la presión en las cámaras depende del flujo "i", el cual depende de la diferencia de presión de las cámaras y de la resistencia entre ellas, por lo tanto surgen siete ecuaciones en el sistema que describen el aumento de presión en cada cámara. Estas ecuaciones son las siguientes:

Cámara aorta abdominal (AA)

$$(1)\quad \Delta P_{AA} = \Delta P_{abd} + \frac{1}{C_{AA}}(i_a - i_s)\Delta t = \Delta P_{abd} + \frac{\Delta t}{C_{AA}}\left[\frac{1}{R_a}(P_{Ao} - P_{AA}) - \frac{1}{R_s}(P_{AA} - P_{IVC})\right].$$

Cámara vena cava inferior (IVC)

$$(2)\quad \Delta P_{IVC} = \Delta P_{abd} + \frac{1}{C_{IVC}}(i_s - i_v)\Delta t = \Delta P_{abd} + \frac{\Delta t}{C_{IVC}}\left[\frac{1}{R_s}(P_{AA} - P_{IVC}) - \frac{1}{R_v}(P_{IVC} - P_{RH})\right]$$

Cámara carótida (Car)

$$(3)\quad \Delta P_{car} = \frac{1}{C_{car}}(i_c - i_h)\Delta t = \frac{\Delta t}{C_{car}}\left[\frac{1}{R_c}(P_{Ao} - P_{car}) - \frac{1}{R_h}(P_{car} - P_{jug})\right]$$

Cámara yugular (jug) Siendo N = 1 normalmente e N = 0 cuando se efectúan las compresiones

$$(4)\quad \Delta P_{jug} = \frac{1}{C_{jug}}(i_h - i_j)\Delta t = \frac{\Delta t}{C_{jug}}\left[\frac{1}{R_h}(P_{car} - P_{jug}) - N\frac{1}{R_j}(P_{jug} - P_{RH})\right]$$

Cámara aórtica (Ao) Siendo E = 1 cuando la válvula aórtica se encuentra abierta.

$$(5)\quad \Delta P_{Ao} = \Delta P_{chest} + \frac{1}{C_{Ao}}(i_o - i_c - i_a - i_{ht})\Delta t$$
$$= \Delta P_{chest} + \frac{\Delta t}{C_{Ao}}\left[E\frac{1}{R_o}(P_p - P_{Ao}) - \frac{1}{R_c}(P_{Ao} - P_{car}) \right.$$
$$\left. - \frac{1}{R_a}(P_{Ao} - P_{AA}) - \frac{1}{R_{ht}}(P_{Ao} - P_{RH}) \right],$$

Cámara corazón derecho (RH) Siendo F = 1 cuando la válvula pulmonar se encuentra abierta.

Cámara de bomba (P)

$$(6)\quad \Delta P_{RH} = \Delta P_{chest} + \frac{1}{C_{RH}}(i_j + i_v + i_{ht} - i_i)\Delta t$$
$$= \Delta P_{chest} + \frac{\Delta t}{C_{RH}}\left[N\frac{1}{R_j}(P_{jug} - P_{RH}) + \frac{1}{R_v}(P_{IVC} - P_{RH}) \right.$$
$$\left. + \frac{1}{R_{ht}}(P_{Ao} - P_{RH}) - F\frac{1}{R_i}(P_{RH} - P_p) \right],$$

$$(7)\quad \Delta P_p = \Delta P_{chest} + \frac{1}{C_p}(i_i - i_o)\Delta t$$
$$= \Delta P_{chest} + \frac{\Delta t}{C_p}\left[F\frac{1}{R_i}(P_{RH} - P_p) - E\frac{1}{R_o}(P_p - P_{Ao}) \right].$$

La presión en el pecho es variable y determinada por el sensor y su calibración.

Estas ecuaciones deben ser evaluadas a intervalos de tiempos pequeños para que sean convergentes, y con las mismas se puede obtener la presión en cualquier cámara así como el flujo entre las mismas.

D Modelado de ventilación

El modelado respiratorio se basa en el intercambio de gases como ocurre con los alvéolos de los pulmones. Este modelado depende de la fracción parcial de oxígeno aspirada FiO2, así como de presiones parciales de oxígeno y dióxido de carbono, tanto en el alvéolo como en los vasos sanguíneos que están en contacto.

Se utilizó como ejemplo un metabolismo constante durante la simulación, es decir, el consumo de oxígeno y la producción de dióxido de carbono es constante. La figura D.1 se basa en el trabajo de Tehrani[3].

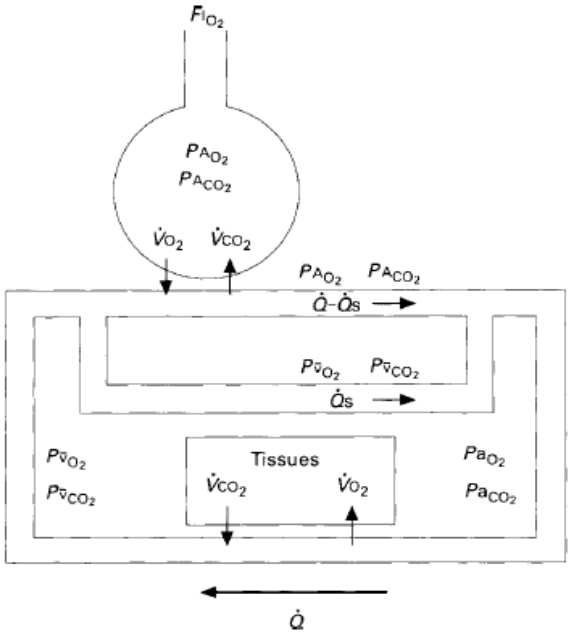


Figura D.1. Modelo del sistema respiratorio humano.

La ecuación del flujo de masa en los alvéolos para el dióxido de carbono:

$$(C_{VTCO_2} - C_{aCO_2})(1-\alpha)Q = \frac{v}{P_b - 47}\frac{dP_{ACO_2}}{dt} + \frac{P_{ACO_2} - P_{ICO_2}}{P_b - 47}\frac{dv}{dt}$$

Y para el oxígeno:

$$(C_{VTO_2} - C_{aO_2})(1-\alpha)Q = \frac{v}{P_b - 47}\frac{dP_{AO_2}}{dt} + \frac{P_{AO_2} - P_{IO_2}}{P_b - 47}\frac{dv}{dt}$$

[3] Tehrani Fleur, "Dinamic modelling of the human respiratory system", PhD Thesis, University of London, 1981.

El primer término es la deferencia de las concentraciones de CO2 y O2, respectivamente, entre las venas y las arterias, multiplicado por un porcentaje *alpha*, que no participa en el intercambio (*shunt fraction*). Y el segundo término es la variación de la concentración parcial en el alvéolo sumado a las presiones parciales del aire inspirado, este valor se vuelve cero durante la expiración. Pb representa la presión barométrica.

Para simplificar el modelo, la relación entre las presiones parciales en el alvéolo y los vasos sanguíneos, es la siguiente:

Al asumir que la mezcla es homogénea entre la sangre venosa y la arterial. Siendo *alpha* la *shunt fraction*.

$$P_{ACO_2} = P_{aCO_2} \qquad P_{AO_2} = P_{aO_2} + K$$

$$C_{amCO_2} = (1-\alpha)C_{aCO_2} + \alpha C_{VTCO_2}$$

$$C_{amO_2} = (1-\alpha)C_{aO_2} + \alpha C_{VTO_2}$$

Y la ecuación para el tejido, la siguiente:

$$C_{VTCO_2}Q = C_{amCO_2}Q + MR_{TCO_2} - S_T\frac{dC_{TCO_2}}{dt}$$

$$C_{VTO_2}Q = C_{amO_2}Q - MR_{TO_2} - S_T\frac{dC_{TO_2}}{dt}$$

Donde MR es la tasa de metabolismo constante, y St una constante de equivalencia de almacenamiento de gas en los tejidos.

Otras ecuaciones que deben ser utilizadas para completar el modelado son las siguientes:

Estas ecuaciones describen las curvas de disociación de los gases.

$$C_{CO_2} = K_3 P_{CO_2} \qquad C_{O_2} = K_4(1 - e^{-K_5 P_{O_2}})^2.$$

E Descripción del protocolo I^2C

Los dos cables, llamados *serial Data* (SDA) y *serial Clock* (SCL), transportan la información entre los dispositivos conectados al *bus*, ambos cables son bidireccionales y se encuentran conectados a una fuente de voltaje positiva. Esta fuente es de corriente o una resistencia *pull up*[4]. Cuando el *bus* está libre las dos líneas se encuentran en su estado alto[5].

La etapa de salida de los dispositivos debe ser de tipo *colector abierto*[6] para que pueda funcionar el arbitraje.

Para que la información sea válida, la línea de datos debe mantenerse estable durante el periodo alto del reloj. El cambio de línea sólo se permite durante el estado bajo[7] del reloj.

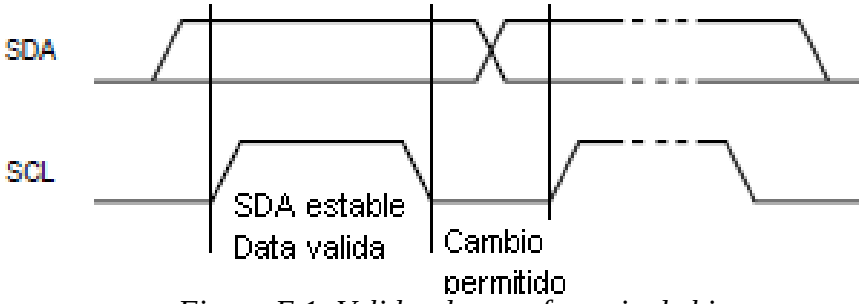


Figura E.1: Validez de transferencia de bit

Toda transferencia debe ser iniciada con una condición de inicio (*Start*), y una condición de fin (*Stop*). Todas estas generadas por el maestro, la condición de inicio se encuentra definida con una transición de alto a bajo de la línea SDA, mientras la línea SCL se encuentra en su estado alto; y la condición de fin estando definida por la transición de bajo a alto durante el estado alto de SCL como se muestra en la figura E.2.

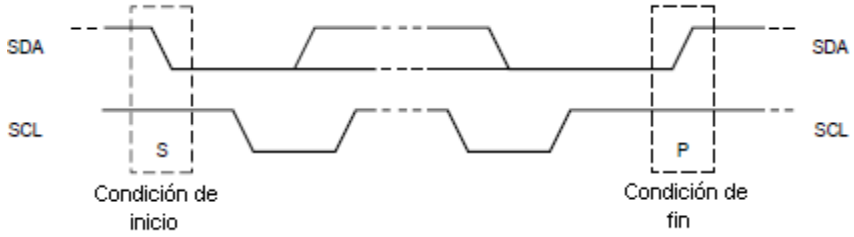


Figura E.2. Condiciones de inicio y de fin en el bus I^2C

---

[4] Se le llama resistencia en *Pull up*, ya que se dice que la fuente 'jala hacia arriba' el voltaje de las terminales de la resistencia al mismo potencial cuando no existe paso de corriente.

[5] Se llama estado alto cuando en los sistemas digitales el valor del voltaje se encuentra arriba del umbral especificado para su valor '1' lógico.

[6] El colector abierto es un tipo de salida de los circuitos integrados, donde en vez de mandar como salida un voltaje o corriente específico, se externaliza directamente la terminal del transistor de salida.

[7] Se llama estado bajo cuando en los sistemas digitales el valor del voltaje se encuentra abajo del umbral especificado para su valor '0' lógico.

Hay que recordar que durante la transmisión de los datos no se debe modificar la línea SDA durante el estado alto de SCL pues se generarían falsos positivos sobre las condiciones de inicio y fin de una transmisión. Es posible repetir una condición de inicio sin tener que dar una condición de fin con la finalidad de no liberar el *bus*.

El formato de los datos debe ser de 8 *bits*[8], enviando primero aquel más significativo (*Most Significant Bit First*), el número de *bytes* (8 *bits*) que puede ser transmitido no es restringido, todos deben ser seguidos por un *bit* de recibido (*ACK*[9]).

El *bit* ACK indica que el *byte* ha sido recibido satisfactoriamente y por lo tanto puede enviarse el siguiente *byte*, es generado por el maestro, pero este libera la línea SDA, para que el esclavo pueda llevarla hacia nivel bajo para generar el *bit* de ACK. Si la línea permanece en alto significa que el *byte* no fue bien recibido (*NACK*[10]) debido a:

- No existe un dispositivo con la dirección generada por el maestro así que ningún dispositivo contestará al *bit* ACK.
- El dispositivo no está listo para recibir porque se encuentra realizando otra tarea.
- Durante la transmisión el dispositivo recibe datos o comandos que no comprende.
- Durante la transferencia, el dispositivo ya no puede recibir más datos.
- El maestro termina la transferencia de datos que el esclavo se encuentra transmitiendo.

La transmisión de datos debe seguir el formato mostrado en la figura E.3.

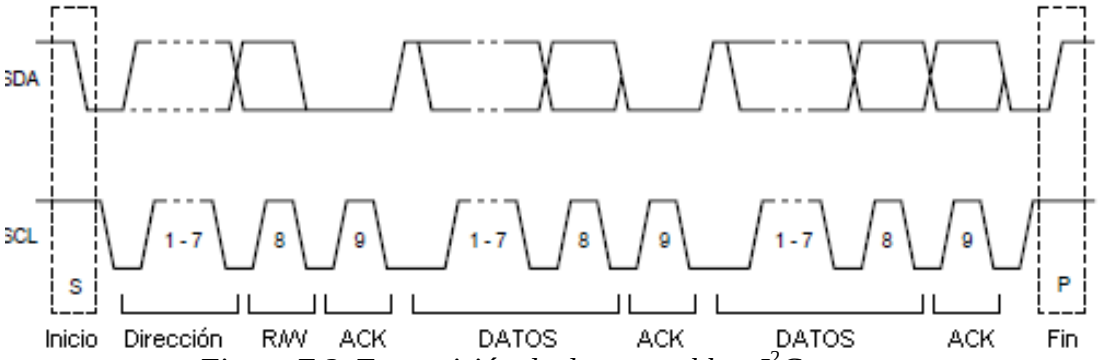


Figura E.3. Transmisión de datos en el bus I^2C

Después de la condición de inicio la dirección del esclavo debe ser enviada, la dirección consta de 7 *bits* de identificación y un *bit* de dirección de la información, el "cero" en este *bit* quiere decir que la información se dirige al esclavo (*write*), el "uno" significa que el esclavo debe transmitir información (*read*).

El esclavo también tiene la capacidad de pausar la transmisión, obligando a la línea SCL a mantenerse en su estado bajo. Esta característica permite que el esclavo tenga tiempo suficiente para procesar la información recibida o preparar la información a transferir.

Cuando se utilizan varios maestros en el mismo *bus* es necesario un método para saber quién va tomar el control del *bus* y realizar su transmisión, esto se logra a través de la

[8] El bit es el acrónimo de **B**inary Dig**it**, que significa Digito binario.
[9] El ACK es el acrónimo de Acknowledge, que significa recibido.
[10] El NACK es el acrónimo de No Acknowledge, que significa no recibido.

sincronización del reloj y el arbitraje. En sistemas de un solo maestro esto no es necesario.

La sincronización del reloj es posible gracias a un AND cableado[11]. Con esto se quiere decir que una transición de alto a bajo en SCL provocará que los demás maestro comiencen a contar su periodo bajo del reloj, este maestro mantendrá SCL en bajo hasta que termine su periodo bajo, pero si algún otro maestro continúa con su periodo bajo la línea mantendrá el mismo estado.

Los maestros que ya hayan acabo de contar su periodo bajo quedarán en estado de espera (*stand by*). Cuando se libera SCL cambia su estado a alto, el contador de estado alto de todos los maestro se inicia y el primero que termine cambia el estado de la línea a bajo. Por lo tanto el periodo del reloj está definido por el contador del maestro con el periodo bajo más largo y el contador del maestro con el periodo alto más corto.

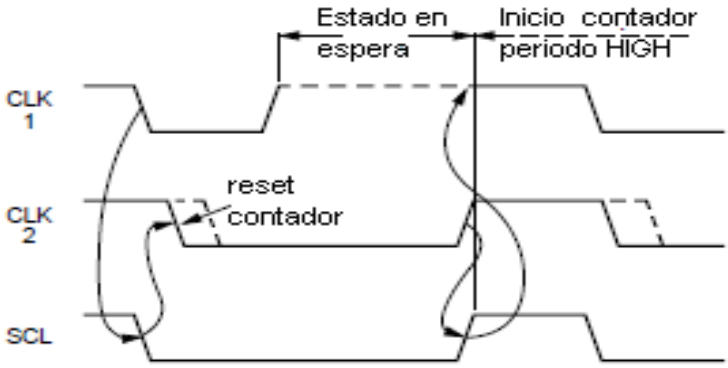


Figura E.4. Sincronización de reloj

El arbitraje entra en operación cuando dos maestros dan una condición de inicio antes del tiempo mínimo[12] de reten de la posición de inicio, por lo tanto el arbitraje debe entrar en operación para determinar qué maestro completará su transmisión.

Se procede *bit* por *bit*, cada maestro revisa la línea SDA para ver si ésta coincide con lo que él está mandando. Si en algún momento la línea no coincide, es porque el maestro ha perdido el arbitraje, entonces deberá cesar la transmisión de información y pasar de inmediato al modo esclavo, por lo tanto no hay pérdida de información.

Es importante reconocer que la prioridad en el arbitraje la tienen los datos de menor valor, sea en la dirección del esclavo o en la información a transmitir.

[11] El AND-cableado es una configuración donde la funcional lógica del AND se encuentra implícita en el cableado.

[12] Para mayor información sobre el tiempo mínimo consultar The I2C Bus especification, versión 2.1, 2000

F Descripción del protocolo USB

El cable está formado por terminales; alimentación (Vcc), tierra (Gnd), D+ y D-. Estos dos último son utilizados para transferir la información.

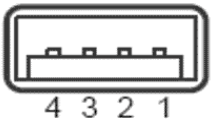


Figura F.1. Numeración de pines, USB specification, rev 2.0, 2000

Al igual que el *bus* I^2C, la comunicación para el *bus* universal serial es iniciada únicamente por el *host*[13] No se pueden conectar dos *host* entre ellos, por lo tanto se han diseñado dos tipos de conectores una para el *host* tipo A y otro para el dispositivo tipo B. La forma de los conectores se muestra en la figura F.2.

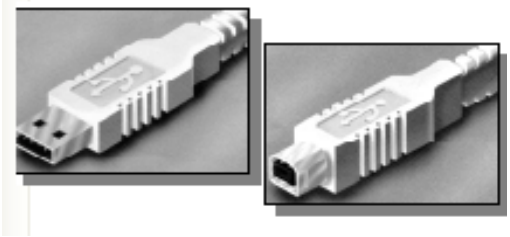


Figura F.2. Tipos de conectores

La transmisión de los datos se hace vía diferencial, es decir, se mide la diferencia entre los dos cables y este dato proporciona el valor de la señal. Este modo de transferencia permite mayores velocidades de transmisión, con cables relativamente largos.

En este proyecto sólo se utilizó el modo *Full Speed*, entonces será el descrito aquí.

La señal diferencial debe contener las siguientes características para que pueda ser detectado como un “uno” diferencial o un “cero” diferencial, respectivamente.

Estado del Bus	Niveles de las señales		
	En la salida del Host	En la entrada del dispositivo	
		Requerida	Aceptable
Diferencial "1"	D+ > 2.8 V y D- < 0.3 V	(D+) - (D-)> 200mV y D+ > 2.0V	(D+) - (D-)> 200mV
Diferencial "0"	D- > 2.8 V y D+ < 0.3 V	(D-) - (D+)> 200mV y D- > 2.0V	(D-) - (D+)> 200mV

Tabla F.1 Niveles de las señales en el bus USB

[13] Se le llama Host al computador que funciona como punto de inicio y final de las transferencias de datos.

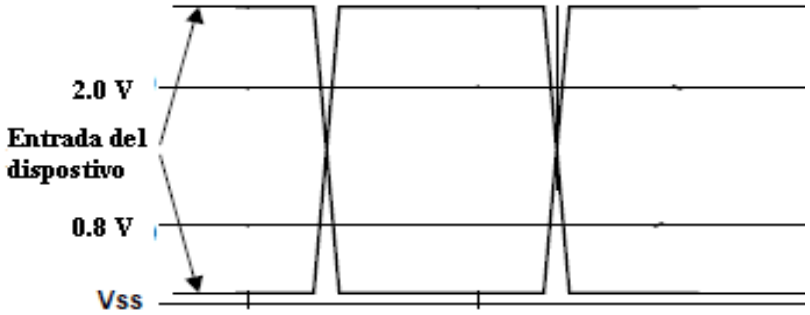
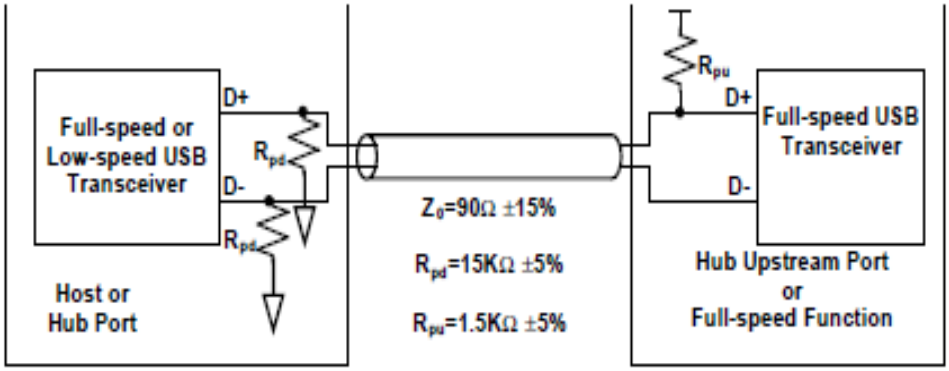
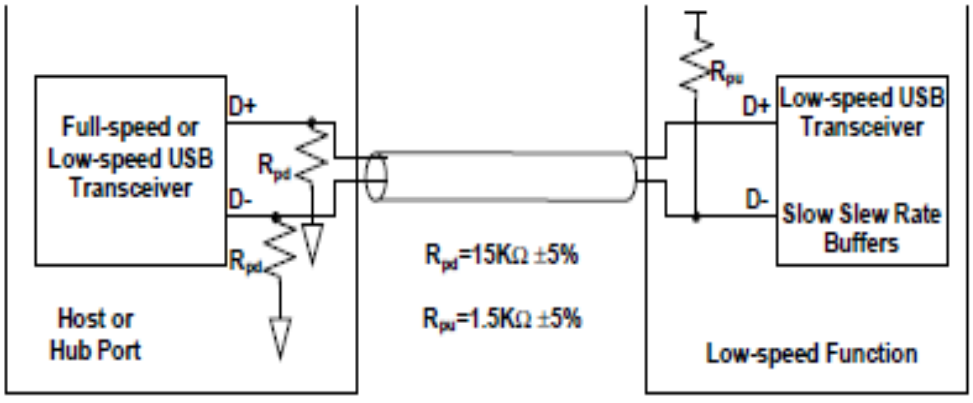


Figura F.3 Forma de onda de las señales

Para distinguir entre las diferentes velocidades de los dispositivos, el *host* revisa cuál es el estado del *bus* cada que es conectado un dispositivo. Si el *bus* se encuentra detecta un "uno" diferencial, se trata de un dispositivo *Full Speed*; si el estado es "cero" diferencial entonces el dispositivo es *Low Speed*. Para un dispositivo *High Speed* primero se debe identificar como un *Full Speed* y después cambiar al modo a *High Speed*. Este comportamiento puede comprenderse al ver el arreglo de resistencias para el bus mostrado en la figura F.4.



a)



b)

Figura F.4. Configuración del cable y resistencias, a) Full speed, b) Low speed

Cuando no hay algún dispositivos conectado las dos líneas diferenciales se encontrarán en un nivel bajo, de esta manera el *host* puede detectar la conexión de un dispositivo cuando alguna de ellas cambia de estado.

Para hablar de los estados lógicos de manera general, deben definirse dos nuevos, así no se tendrá que invertir los estados dependiendo de la tasa de transferencia.

Se definen de la siguiente manera:

	Full Speed	*Low Speed*
Estado **J**	Diferencial "1"	Diferencial "0"
Estado **K**	Diferencial "0"	Diferencial "1'

Tabla F.2 Definición de los estados lógicos en el bus USB
(Elaboración propia a partir de USB Specification, Rev. 2.0, 2000)

Debido a que el *bus* no cuenta con señal de reloj, la transmisión debe ser iniciada con *bit* de sincronización, esto sucede en el inicio de paquete (SOP[14]) y es señalado con el cambio de estado **J**(estado *idle*[15]) a **K**, este representa al primer *bit* que sincroniza. La secuencia llamada SYNC[16] requiere que sean enviados 3**KJ** seguidos por 2**K** para un total de 8 símbolos, como se muestra en la figura 2.13, al terminar la transmisión el paquete es finalizado (EOP[17]) mandando las dos líneas diferenciales a *gnd*, definiendo el estado SE0[18].

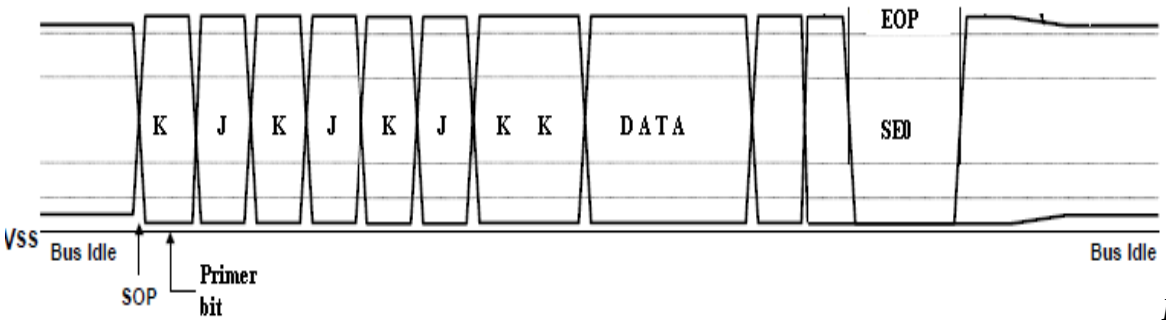


Figura F.5. Forma de onda de paquete en el bus USB

El estado SE0 también es utilizado para mandar señales de *reset* al dispositivo que esté conectado en el puerto.

Los datos son transmitidos mediante la codificación NRZI[19], en esta codificación la falta de cambio de nivel lógico es representada por los "uno" y los "cero" designan un cambio en el nivel. La figura F.6 muestra un ejemplo de la codificación.

[14] SOP es el acrónimo Start of Packet, significa inicio de paquete.
[15] El estado Idle es un estado de espera, es decir se esta a la escucha de un nuevo evento.
[16] SYNC es el acrónimo de Synchronization, que significa sincronización.
[17] EOP es el acrónimo de End of Packet, que significa fin de paquete.
[18] SE0 es el acrónimo de Single Ended Zero, que significa que las dos líneas se encuentran en estado bajo.
[19] NZRI por sus siglas en inglés, Non Return To Zero Inverted

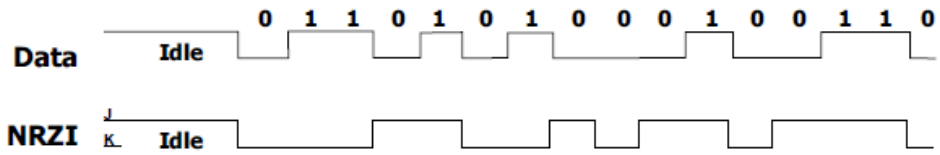


Figura F.6. Ejemplo de codificación NZRI

Para evitar la desincronización de los relojes, un "cero" es agregado después de seis "uno" consecutivos en la información a transmitir, antes de ser codificada en NRZI. El receptor debe reconocer estos *bits* y descartarlos.

Distribución de energía

Los dispositivos conectados al *bus* pueden ser alimentados por tener propia alimentación. Si el dispositivo es alimentado por el *bus* puede obtener hasta 100 mA[20] de éste.

La configuración se encuentra en baja energía pues todo dispositivo recién conectado está configurado como de baja energía. Se debe configurar por medio del *software* si se requiere mayor corriente, con la posibilidad de sustraer hasta 500 mA del *bus*. Ningún dispositivo puede entregar energía al bus, solo le está permitido sustraerla.

El dispositivo tampoco deberá proporcionar energía a las líneas diferenciales. Al no estar presente el *Vbus*[21] se removerá la energía en menos de 10 segundos.

Protocolo

El elemento principal del protocolo es el paquete y debe iniciarse con las secuencia de sincronización SYNC seguido por su identificador de paquete PID[22]. Existen tres tipos de paquetes; *token*, *data* y *handshake*.

Es importante comentar que los *bits* son enviados por el *bus* siendo el menos significativo el primero en salir.

Los paquetes son divididos en estructuras llamadas sub campos, los cuales pueden conformar los distintos paquetes. Un sub campo que debe estar incluido en todos los paquetes es el identificador de éstos.

El identificador de paquete está formado por 4 *bits* seguido por otro grupo de cuatro *bits* como se muestra en la figura F.7.

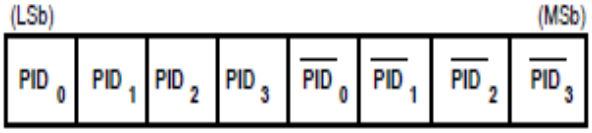


Figura F.7.Formato del identificador del paquete

Los bits de confirmación aseguran la correcta identificación del paquete formándolos con el *complemento uno*[23] del PID original.

Si el PID es inválido se considera que está incorrecto y todo el paquete es descartado.

---

[20] mA es la abreviación de miliAmperes, unidad de medida de corriente eléctrica.
[21] El Vbus es el voltaje de alimentación que otorga el host a los dispositivos.
[22] PID es el acrónimo de Packet Identifier, significa el identificador de paquete.
[23] El Complemento uno es una operación binaria, la cual consta de invertir todos los valores del número binario.

Si un PID válido llega a un dispositivo que no cuenta con la función solicitada debe ignorar al paquete recibido, por ejemplo, si a un dispositivo le llega una petición de información y éste sólo está programado para recibirla. Los identificadores de paquete son descritos en la tabla F.3.

Tipo PID	Nombre PID	PID <3:0>	Descripción
Token	OUT	0001B	Información de Host → Dispositivo
	IN	1001B	Información de Dispositivo -> Host
	SOF	0101B	Inicio de Frame
	SETUP	1101B	Información de Setup de Host → Dispositivo
Data	DATA0	0011B	Información par
	DATA1	1011B	Información non
Handshake	ACK	0010B	Información recibida sin errores
	NACK	1010B	No se puede recibir ni transmitir información
	STALL	1110B	El endpoint se encuentra desactivado
	NYET	0110B	Sin respuesta

Tabla F.3 Tabla de los valores del identificador de paquete

Los dispositivos se identifican utilizando dos sub campos; la dirección y el *endpoint*[24]. El dispositivo debe decodificar por completo los dos sub campos.

La dirección ADDR[25] es la fuente o el destino del paquete dependiendo del PID del paquete, está formada por 7 *bits* ADDR<6:0> que dan un total de 128 direcciones. La dirección "cero" está reservada ya que es la dirección predeterminada de todo dispositivo recientemente conectado al *bus*, éste después deberá ser enumerado.

El *endpoint* son 4 bits adicionales, para la identificación de la función dentro del dispositivo, por ejemplo una *webcam* puede tener un micrófono, para lo cual podría contener un *endpoint* para el video y otro para el audio. El *endpoint* "cero" se encuentra reservado, como el de control por defecto. Un dispositivo puede soportar hasta 16 *endpoints*.

La información se encuentra contenida en el sub campo *data* y su tamaño puede variar desde 0 *bytes* hasta 1024, y debe ser un número entero de *bytes*.

La verificación de la información se hace mediante el sub campo *Cyclic Redundancy Check*[26] (CRC) teniendo un tamaño de 5 *bits* para el paquete tipo *token* y 16 *bits* para el *data*. Todo paquete que falle esta verificación será descartado.

El paquete tipo *token* consiste de un PID, puede ser IN OUT o SET UP, de una dirección y del *enpoint*. Una transacción OUT o SET UP significa que el subsecuente paquete *data* contendrá información para el dispositivo especificado en este *token*.

Una Transacción tipo IN, significa que el dispositivo especificado en el *token* deberá emitir un paquete tipo DATA.

[24] El endpoint es el nombre que se la da a la entidad final de un canal de información.
[25] ADDR es el acrónimo de Address, que significa dirección.
[26] La Cyclic Redundancy Check es un tipo de verificación de errores en el mensaje, se detalla en el apéndice A.

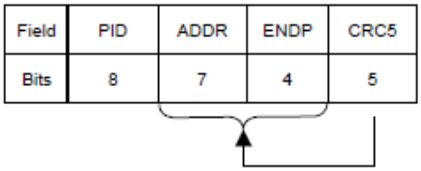


Figura F.8. Formato del paquete Token.

Estando el paquete verificado mediante un CRC, como se muestra en la figura número F.8, solo el *host* podrá emitir los paquetes tipo *token*, es decir que ningún dispositivo podrá iniciar una comunicación.

Los paquetes tipo DATA están formados por un PID, la DATA, que puede contener de 0-1024 *bytes*, y su CRC, como se muestra en la figura F.9. El PID puede variar entre DATA0 y DATA1, esto permite realizar una sincronización por alternado de la información.

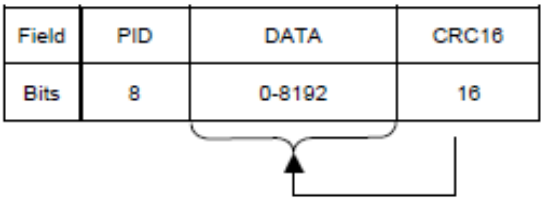


Figura F.9. Formato del paquete Data

Los paquetes tipo *Hand Shake* son utilizados en la última etapa de transmisión para informar el estado de la transacción, están formados únicamente por su PID y sólo están permitidos en algunos tipos de transacciones, además pueden sustituir la etapa de datos. Si después de emitir un paquete tipo *handshake* no se recibe un EOP, entonces después de un tiempo de *bit*, ya terminado el envío, el paquete *handshake* será descartado.

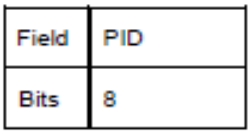


Figura F.10. Formato paquete Handshake

Si el *handshake* es del tipo ACK indica cuál fue el paquete que se transmitió de manera correcta.

Si el *handshake* es del tipo NACK quiere decir que el dispositivo no está listo para recibir el otro paquete o que no hay información para transferir.

Si el *handshake* es del tipo STALL significa que el dispositivo no puede soportar la función solicitada.

Los paquetes que conforman una transacción varía dependiendo del tipo de la misma, existen cuatro tipos de transacciones; *bulk*, *control*, i*nterrupt* e *isochronous*.

Flujo de información

El USB está diseñado para que al usuario le sea simple conectar un dispositivo, como en la figura F.11 inciso a. En realidad es más complejo, como en la figura F.11 inciso b. Su diseño es en forma de capas, por lo tanto cada capa presenta cierta independencia de la otra para que el desarrollador se concentre únicamente en su aplicación.

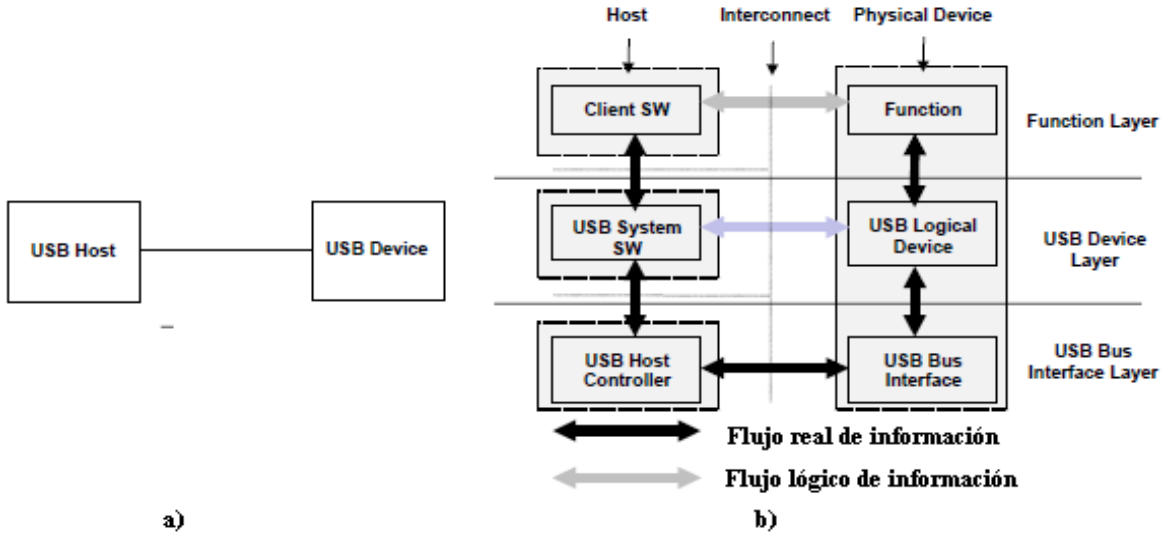


Figura F.11. Flujo de información.
a) Aparente, b) Real

Los dispositivos se encuentran conectados físicamente al *bus* mediante una topología tipo estrella, con puntos de conexión en dispositivos especiales llamados *hubs*. Esta conexión permite hasta siete niveles de interconexión, incluyendo al *root hub*. Por ejemplo, en la figura F.12, el *root hub* pertenece al primer nivel, los dispositivos y *hubs* conectados a él pertenecen a un segundo nivel y los dispositivos conectados a los *hubs* pertenecen a un tercer nivel.

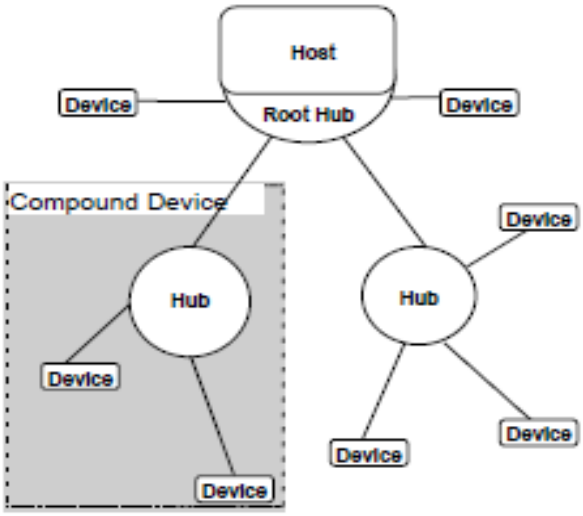


Figura F.12. Topología del bus USB

Estos niveles no son visibles, parecería que todos los dispositivos están conectados directamente al *host*, por lo mismo, es conveniente definir el concepto de tubería o PIPE, estos son puntos de conexión lógicos entre el cliente (*host*) y el dispositivo (*endpoint*). Así, se define una comunicación más simple como la presentada en la figura 2.19, inciso a.

Hay dos tipos de PIPE, los de mensaje y los de tipo *stream*. El tipo *stream* no contienen ninguna estructura, los datos que entran a la tubería son igualmente entregados al dispositivo y son unidireccionales. Los PIPE tipo de mensaje contienen una estructura definida, y aunque los datos no son procesados por el *bus* se necesita que la transmisión inicie con un *request* por parte del *host*, que luego siga una etapa de datos y se termine con el estado de la transmisión. Se puede enviar un *request* a la vez, esto hasta que ya haya sido procesado, el siguiente se podrá enviar en el orden de llegada (*First In First Out*). Este tipo de tubería es bidireccional.

Para dar más flexibilidad a los dispositivos que utiliza el *bus*, hay 4 tipos de transacciones, cada una con sus características y prioridad dentro del *bus*. La primera es la transferencia de *control*. Esta permite el acceso a distintas partes del dispositivo, su intención es soportar comunicaciones del tipo *configuración/comando/estado*, es una tubería de mensaje, por lo tanto, tiene una estructura definida, como se puede observar en la figura F.13. Todos los datos enviados en la misma transferencia deben estar en la misma dirección, la cual, se encuentra indicada en el PID. El tamaño de la etapa de datos debe ser de 8, 16, 32, y 64 *bytes*. Si ocurre un error durante alguna etapa, la transferencia será retransmitida.

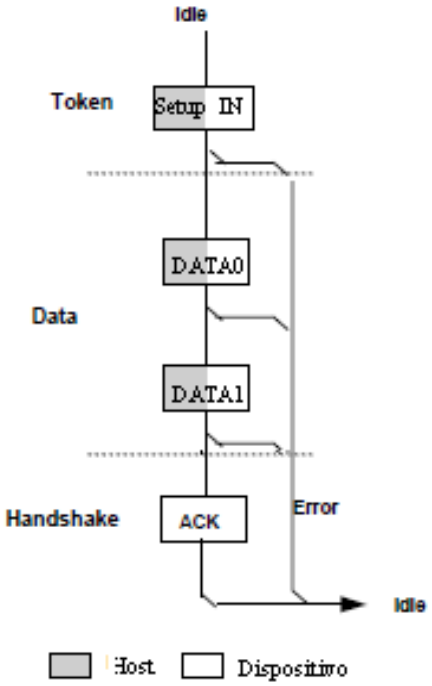


Figura F.13. Ejemplo de transferencia tipo Control

La segunda transferencia es de tipo *Bulk*. Está se encuentra diseñada para transmitir información relativamente grande en tiempos variables cuando el *bus* esté disponible. Es unidireccional, para poder transmitir en dos direcciones deberá utilizar dos tuberías. El tamaño de su etapa de datos debe ser 8, 16, 32 y 64 *bytes*. Al presentarse un error durante alguna etapa la transferencia será retransmitida.

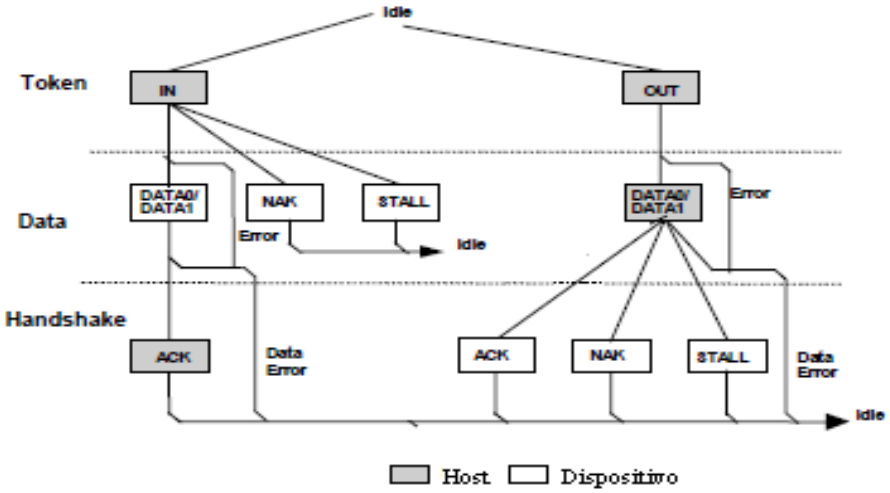


Figura F.14. Ejemplo de transferencia tipo Bulk

Transferencia tipo *Interrupt*. Este tipo de transacción es utilizada en dispositivos que necesitan transmitir información de manera no frecuente que debe ser verificada en periodos específicos. La transferencia tipo *interrupt* es unidireccional, cuenta con un máximo de 64 *bytes* para la etapa de datos. La información se retransmite hasta que se reciba un ACK como respuesta de aceptación.

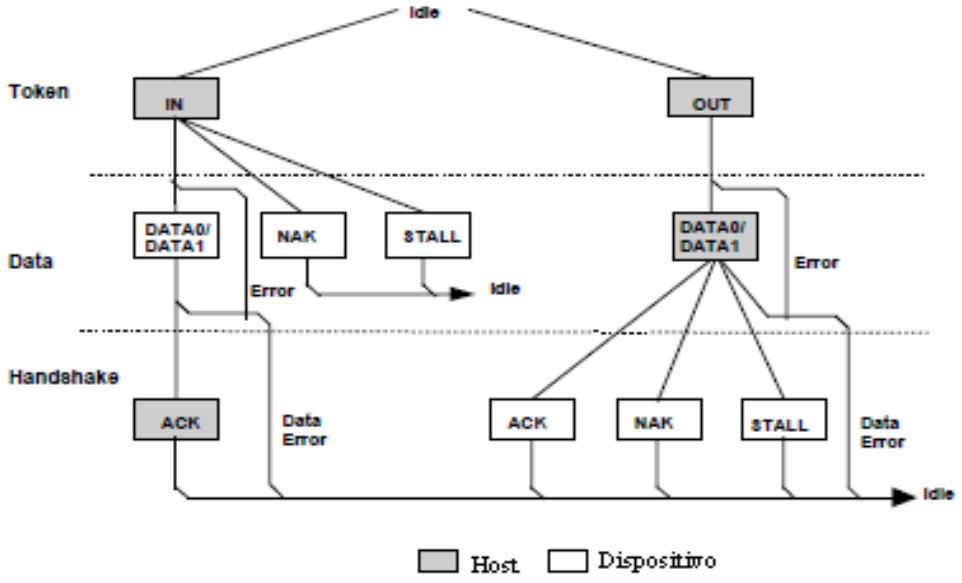
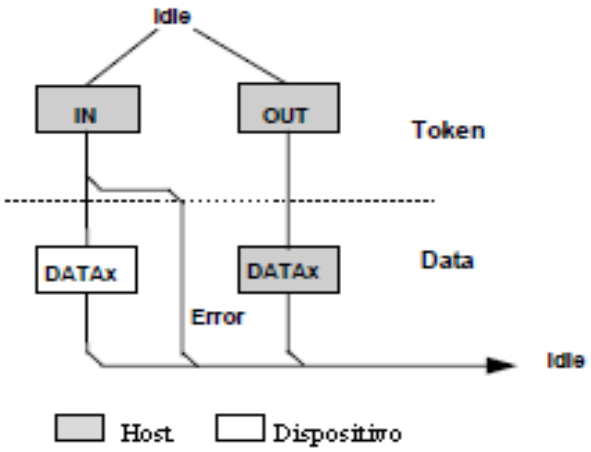


Figura F.15. Ejemplo de transferencia tipo Interrupt.

Por último, tipo de transferencia i*sochronous*. Este indica una tasa de transferencia constante con tolerancia de errores. Necesita un acceso al *bus* con una latencia constante. Tiene un máximo de 1023 *bytes* para la etapa de datos. La información no es retransmitida en caso de error.



F*igura F.16. Ejemplo de transferencia tipo Isochronous*

Para poder manejar los distintos tipos de transferencia, y hacerlo en el momento indicado, el *bus* encapsula las transacciones en *frames*, estos duran un milisegundo, cada transacción tiene un ancho de banda máximo reservado en cada *frame*. Las siguientes tablas muestran algunos ejemplos dependiendo de la carga de datos y tipo de transferencia.

Protocol Overhead (45 bytes)			9 SYNC *bytes*, 9 PID *bytes*, 6 *endpoints* +CRC *bytes*, 6 CRC *bytes*, 8 *setup data bytes* y 7 de retraso entre los paquetes (EOP, etcétera)		
Carga de datos	Número máximo de transferencias	*Bytes* **útiles/no útiles** por *frame*	Ancho de banda*	*Bytes* restantes	Tasa de transferencia
8	28	224 / 1260	4 / 14.9 / 84 %	16	1792 kbps
16	24	384 / 1080	4 / 25.6 / 72 %	36	3072 kbps
32	19	608 / 855	5 / 40.5 / 57 %	37	4864 kbps
64	13	832 / 585	7 / 55.5 / 39 %	83	6656 kbps

Tabla F.4 Limites de transferencia de control

** Ancho de banda porfFrame por transferencia/bytes útiles/bytes no útiles*

Protocol Overhead (13 *bytes*)			3 SYNC *bytes*, 3 PID *bytes*, 2 *endpoints* +CRC *bytes*, 2 CRC *bytes* y 3 de retraso entre los paquetes (EOP, etcétera)		
Carga de datos	Número máximo de transferencias	*Bytes* **útiles/no útiles** por *frame*	Ancho de banda*	*Bytes* restantes	Tasa de transferencia
8	71	568 / 923	1 / 37.8 / 61.5 %	9	4544 kbps
16	51	816 / 663	2 / 54.4 / 44.2 %	21	6528 kbps
32	33	1056 / 429	3 / 70.4 / 28.6 %	15	8448 kbps
64	19	1216 / 247	5 / 81 / 16.4 %	37	9728 kbps

Tabla F.5 Limites de transferencia Bulk
** Ancho de banda por frame por transferencia/Bytes útiles/Bytes no útiles*

Protocol Overhead (9 *bytes*)			2 SYNC *bytes*, 2 PID *bytes*, 2 *endpoints* +CRC *bytes*, 2 CRC *bytes* y 1 de retraso entre los paquetes(EOP, etcétera)		
Carga de datos	Número máximo de transferencias	*Bytes* **útiles/no útiles** por *frame*	Ancho de banda*	Bytes restantes	Tasa de transferencia
128	10	1280 / 90	9 / 85.3 / 6 %	130	10240 kbps
256	5	1280 / 45	18 / 85.3 / 3 %	175	10240 kbps
512	2	1024 / 18	35 / 68.2 / 1.2 %	458	8192 kbps
1023	1	1023 / 9	69 / 68.2 / 16.4 %	468	8184 kbps

Tabla F.6 Limites de transferencia Isochronous
** Ancho de banda por frame por transferencia/Bytes útiles/Bytes no útiles*

Protocol Overhead (13 *bytes*)			3 SYNC *bytes*, 3 PID *bytes*, 2 *endpoints* +CRC *bytes*, 2 CRC *bytes* y 3 de retraso entre los paquetes (EOP, etcétera)		
Carga de datos	Número máximo de transferencias	*Bytes* **útiles/no útiles** por frame	Ancho de banda*	*Bytes* restantes	Tasa de transferencia
8	71	568 / 923	1 / 37.8 / 61.5 %	9	4544 kbps
16	51	816 / 663	2 / 54.4 / 44.2 %	21	6528 kbps
32	33	1056 / 429	3 / 70.4 / 28.6 %	15	8448 kbps
64	19	1216 / 247	5 / 81 / 16.4 %	37	9728 kbps

Tabla F.7 Límites de transferencia Interrupción
** Ancho de banda por Frame por Transferencia / Bytes útiles / Bytes no útiles*

Configuración del dispositivo

Debido a que los dispositivos pueden ser conectados y desconectados dinámicamente del *bus*, este debe pasar por una serie de estados antes de poder ser utilizados:

- *Attached*. Este estado es justo cuando el dispositivo es conectado al *bus.*
- Energizado. Cuando el dispositivo es alimentado y sus líneas D+ y D- se encuentran en el estado J (estado *idle*). El *host* es capaz de detectar el dispositivo en este momento y puede reiniciar el ordenador (*reset*) para poner al dispositivo en el siguiente estado.
- *Default*. El dispositivo puede ser alcanzado mediante la dirección "cero", es capaz de responder a los *request* de su descriptor y configuración.
- Enumerado. El *host* le ha asignado una dirección única, por lo tanto, el dispositivo ya no responderá a la dirección por defecto.
- Configurado. El *host* ha seleccionado la configuración que utilizará del dispositivo y se lo ha indicado.

El dispositivo ahora se encuentra listo para ser utilizado por el *software.* Las configuraciones que tiene el dispositivo deben estar especificadas en el *device descriptor*.

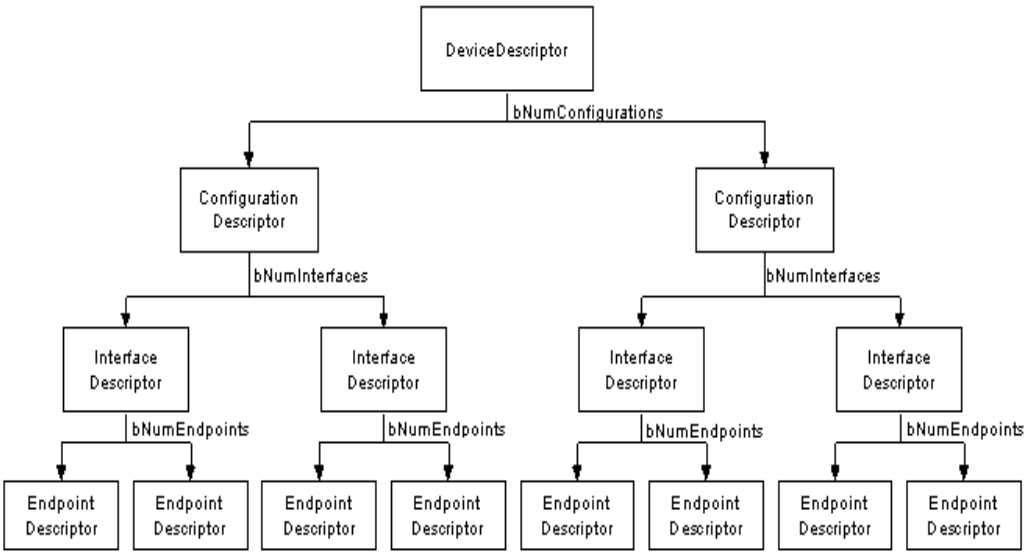


Figura F.17. Arreglo de descriptores en un dispositivo USB

Un dispositivo puede tener una o varias interfaces, estas interfaces con más de una configuración, cada configuración con dos o más *endpoints*, hasta 16. Estos archivos de configuración son abordados en la sección del microcontrolador maestro.