

UTN.BA

UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

Técnicas Digitales II - R4054

Informe de protocolo USB

Estudiante: Lugano Damian – *Leg.:* 1756990

Docente: Furfaro Alejandro

Ayudante: Ferreyro Luciano

Fecha de realización: 26/10/23

Fecha de entrega de informe:

1 Objetivos

El objetivo del presente informe es conectar un dispositivo a un puerto del sistema para capturar y analizar los paquetes de configuración del protocolo USB que se intercambian entre el dispositivo y el Host.

2 Análisis

Paso 1: Conexión y reconocimiento

Se conecta un pendrive a un puerto USB del sistema. El llamado Host o Hub detecta al device mediante la detección del cambio en la impedancia de la línea.

Una vez detectado, se considera al device en estado *Attached* y el Host le provee una alimentación inicial de 100mA. En este punto el device se encuentra en estado *Powered*. En caso de que el device sea self-powered comienza directamente en estado *Powered*.

Cada hub utiliza un endpoint de interrupción para reportar eventos al host. Dicho reporte indica qué puerto del hub experimentó algún evento. En este caso se observa que el address del device es 1.1.0 , esto quiere decir que es el device 1 configurado en el endpoint 0.

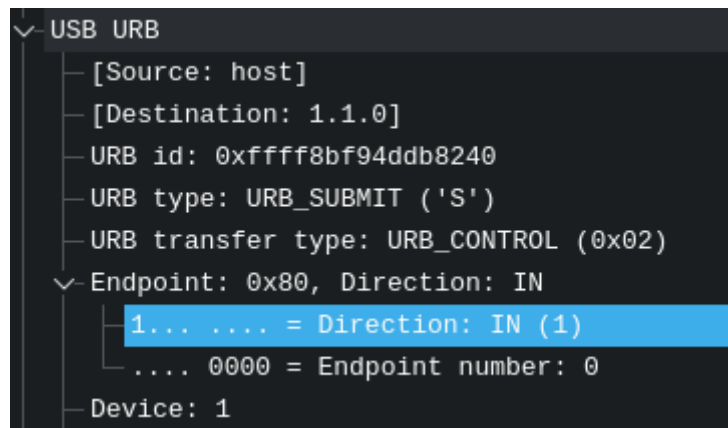


Figura 2.1 - Device Address

Luego, el Hub envía un Get Status Request para determinar si un nuevo device fue conectado a dicho puerto. Además, el hub detectará la velocidad del device al monitorear la tensión en las líneas, la cual se informará en el próximo Get Status Request enviado por el Host.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	host	1.1.0	USBHUB	64	GET_STATUS Request [Port 1]
2	0.000022	1.1.0	host	USBHUB	68	GET_STATUS Response [Port 1]
3	0.000025	host	1.1.0	USBHUB	64	CLEAR_FEATURE Request [Port 1: C_PORT_CONNECTION]
4	0.000050	1.1.0	host	USBHUB	64	CLEAR_FEATURE Response [Port 1: C_PORT_CONNECTION]


```

> Frame 2: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface usbmon0, id 0
> USB URB
  Port Status: 0x0501, PORT_CONNECTION, PORT_POWER, PORT_HIGH_SPEED
    ....1 = PORT_CONNECTION: True
    ...0. = PORT_ENABLE: False
    ...0.. = PORT_SUSPEND: False
    ...0... = PORT_OVER_CURRENT: False
    ...0.... = PORT_RESET: False
    ...1..... = PORT_POWER: True
    ...0..... = PORT_LOW_SPEED: False
    ...1..... = PORT_HIGH_SPEED: True
    ...0..... = PORT_TEST: False
    ...0..... = PORT_INDICATOR: Default colors
  Port Change: 0x0001, C_PORT_CONNECTION
    ....1 = C_PORT_CONNECTION: True
    ...0. = C_PORT_ENABLE: False
    ...0.. = C_PORT_SUSPEND: False
    ...0... = C_PORT_OVER_CURRENT: False
    ...0.... = C_PORT_RESET: False

```

Figura 2.2 - Get Port Status

En la Figura 2.2 se puede observar la respuesta del Get Status, donde se encuentran las configuraciones iniciales como PORT_CONNECTION, PORT_POWER y PORT_HIGH_SPEED.

Paso 2: El hub resetea al device

El host envía al hub un Set Port Feature Request en el cual solicita enviar la condición de reset. El device acepta y se reinicia la máquina de estados del dispositivo.

30	0.109975	host	1.1.0	USBHUB	64	SET_FEATURE Request [Port 1: PORT_RESET]
31	0.297129	1.1.1	host	USB	66	URB_INTERRUPT in
32	0.297134	host	1.1.1	USB	64	URB_INTERRUPT in
33	0.297135	1.1.0	host	USBHUB	64	SET_FEATURE Response [Port 1: PORT_RESET]
34	0.367880	host	1.1.0	USBHUB	64	GET_STATUS Request [Port 1]
35	0.367916	1.1.0	host	USBHUB	68	GET_STATUS Response [Port 1]
36	0.367920	host	1.1.0	USBHUB	64	CLEAR_FEATURE Request [Port 1: C_PORT_RESET]
37	0.367926	1.1.0	host	USBHUB	64	CLEAR_FEATURE Response [Port 1: C_PORT_RESET]


```

> Frame 35: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface usbmon0, id 0
> USB URB
  Port Status: 0x0503, PORT_CONNECTION, PORT_ENABLE, PORT_POWER, PORT_HIGH_SPEED
  Port Change: 0x0010, C_PORT_RESET
    ...0 = C_PORT_CONNECTION: False
    ...0. = C_PORT_ENABLE: False
    ...0.. = C_PORT_SUSPEND: False
    ...0... = C_PORT_OVER_CURRENT: False
    ...1.... = C_PORT_RESET: True

```

Figura 2.3 - Set Feature Request (Reset)

Al salir del estado de reset, el device está listo para responder a los request del host. Su dirección será por default 0x00 hasta que se le asigne una nueva.

Paso 3: El host envía un Get Descriptor Request solicitando el Device Descriptor

Arranca la transferencia de control.



Figura 2.4 - Get Device Descriptor Request

En la figura se puede observar que el Get Descriptor al device de index 0x00

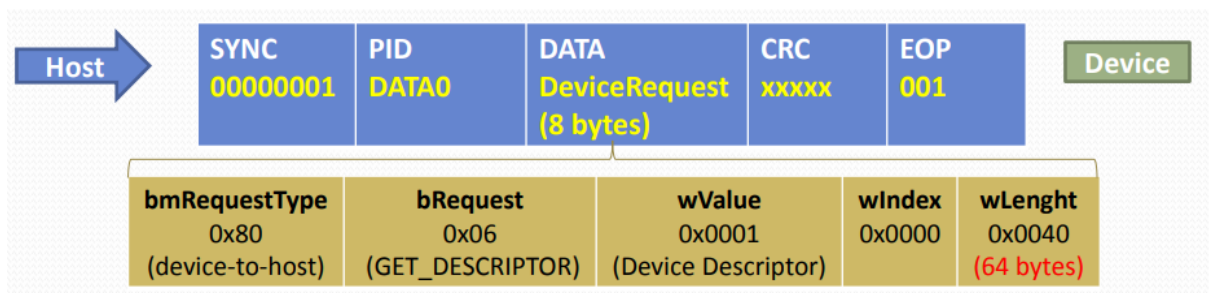


Figura 2.5 - Trama USB del Device Descriptor

La respuesta obtenida del device es la siguiente:

```
38 0.433216 host 1.0.0 USB 64 GET_DESCRIPTOR Request DEVICE
39 0.435646 1.0.0 host USB 82 GET_DESCRIPTOR Response DEVICE
> Frame 39: 82 bytes on wire (656 bits), 82 bytes captured (656 bits) on interface usbmon0, id 0
> USB URB
< DEVICE_DESCRIPTOR
  bLength: 18
  bDescriptorType: 0x01 (DEVICE)
  bcdUSB: 0x0200
  bDeviceClass: Device (0x00)
  bDeviceSubClass: 0
  bDeviceProtocol: 0 (Use class code info from Interface Descriptors)
  bMaxPacketSize0: 64
  idVendor: Kingston Technology (0x0951)
  idProduct: Digital DataTraveler SE9 (0x1665)
  bcdDevice: 0x0100
  iManufacturer: 1
  iProduct: 2
  iSerialNumber: 3
  bNumConfigurations: 1
```

Figura 2.6 - Get Device Descriptor Response

Una vez que el host conoce el tamaño maximo de datos, en este caso bMaxPacketSize=64, genera un estado de reset.

```
38 0.433216 host 1.0.0 USB 64 GET_DESCRIPTOR Request DEVICE
39 0.435646 1.0.0 host USB 82 GET_DESCRIPTOR Response DEVICE
40 0.435681 host 1.1.0 USBHUB 64 SET_FEATURE Request [Port 1: PORT_RESET]
41 0.622646 1.1.0 host USBHUB 64 SET_FEATURE Response [Port 1: PORT_RESET]
42 0.702983 host 1.1.0 USBHUB 64 GET_STATUS Request [Port 1]
43 0.703027 1.1.0 host USBHUB 68 GET_STATUS Response [Port 1]
44 0.703034 host 1.1.0 USBHUB 64 CLEAR_FEATURE Request [Port 1: C_PORT_RESET]
45 0.703040 1.1.0 host USBHUB 64 CLEAR_FEATURE Response [Port 1: C_PORT_RESET]
> Frame 43: 68 bytes on wire (544 bits), 68 bytes captured (544 bits) on interface usbmon0, id 0
> USB URB
< Port Status: 0x0503, PORT_CONNECTION, PORT_ENABLE, PORT_POWER, PORT_HIGH_SPEED
  .... = PORT_CONNECTION: True
  .... = PORT_ENABLE: True
  .... = PORT_SUSPEND: False
  .... = PORT_OVER_CURRENT: False
  .... = PORT_RESET: False
  .... = PORT_POWER: True
  .... = PORT_LOW_SPEED: False
  .... = PORT_HIGH_SPEED: True
  .... = PORT_TEST: False
  .... = PORT_INDICATOR: Default colors
< Port Change: 0x0010, C_PORT_RESET
  .... = C_PORT_CONNECTION: False
  .... = C_PORT_ENABLE: False
  .... = C_PORT_SUSPEND: False
  .... = C_PORT_OVER_CURRENT: False
  .... = C_PORT_RESET: True
```

Figura 2.7 - Get Port Status Response

En la figura se puede observar, en la última línea, que se hace un acknowledge del Reset.

Paso 4: El host asigna una dirección al device

Esto se hace mediante un Set Address Request.

46	0.763360	host	1.0.0	USB	64	SET ADDRESS Request
47	0.767324	1.0.0	host	USB	64	SET ADDRESS Response

>

Frame 46: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface usbmon0, id 0

>

USB URB

>

Setup Data

>

bmRequestType: 0x00

>

0... .. = Direction: Host-to-device

>

.00. = Type: Standard (0x0)

>

...0 0000 = Recipient: Device (0x00)

>

bRequest: SET ADDRESS (5)

>

Device: 3

>

wIndex: 0 (0x0000)

>

wLength: 0

Figura 2.8 - Set Address Request

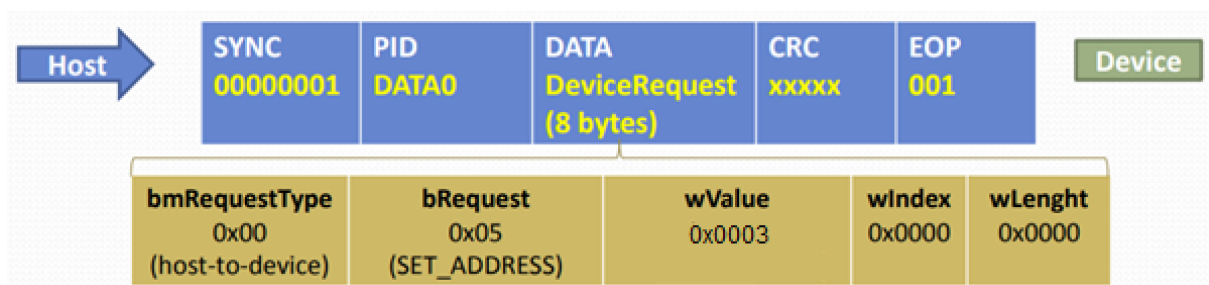


Figura 2.9 - Trama de Set Address Request

En este caso se asignó el address 3 al device. Por lo que a partir de ahora, las transferencias de paquetes se harán a la dirección 3 y se mantendrá así hasta que se desconecte el dispositivo o se reinicie el sistema. El device pasa a modo *Addressed*.

Paso 5: El host solicita todos los descriptores

Un Get Descriptor Request del descriptor de configuración, implica enviar todos sus descriptores subordinados. En este punto un host Windows suele solicitar los primeros nueve bytes de todo el conjunto (lo que corresponde al descriptor de configuración propiamente dicho) para conocer la longitud total. Luego reenvía el request para que el device responda con todos los datos: Configuration + Interface(s) + Endpoint(s).

No.	Time	Source	Destination	Protocol	Length	Info
53	0.809788	1.3.0	host	USB	96	GET_DESCRIPTOR Response CONFIGURATION

```

> Frame 53: 96 bytes on wire (768 bits), 96 bytes captured (768 bits) on interface usbmon0, id 0
> USB URB
< CONFIGURATION_DESCRIPTOR
  bLength: 9
  bDescriptorType: 0x02 (CONFIGURATION)
  wTotalLength: 32
  bNumInterfaces: 1
  bConfigurationValue: 1
  iConfiguration: 0
  Configuration bmAttributes: 0x80 NOT SELF-POWERED NO REMOTE-WAKEUP
  bMaxPower: 100 (200mA)
< INTERFACE_DESCRIPTOR (0.0): class Mass Storage
  bLength: 9
  bDescriptorType: 0x04 (INTERFACE)
  bInterfaceNumber: 0
  bAlternateSetting: 0
  bNumEndpoints: 2
  bInterfaceClass: Mass Storage (0x08)
  bInterfaceSubClass: SCSI transparent command set (0x06)
  bInterfaceProtocol: Bulk-Only (BBB) Transport (0x50)
  iInterface: 0
< ENDPOINT_DESCRIPTOR
  bLength: 7
  bDescriptorType: 0x05 (ENDPOINT)
  bEndpointAddress: 0x81 IN Endpoint:1
  bmAttributes: 0x02
  wMaxPacketSize: 512
  bInterval: 0
< ENDPOINT_DESCRIPTOR
  bLength: 7
  bDescriptorType: 0x05 (ENDPOINT)
  bEndpointAddress: 0x02 OUT Endpoint:2
  bmAttributes: 0x02
  wMaxPacketSize: 512
  bInterval: 0

```

Figura 2.10 - Get Configuration Descriptor Response

Del Configuration Descriptor se pueden destacar algunas configuraciones:

Configuration Descriptor:

- Configuration bmAttributes: 0x80 NOT SELF-POWERED NO REMOTE-WAKEUP
 - bMaxPower: 100(200mA)

Interface Descriptor : class Mass Storage (porque es un pendrive)

- bNumEndpoints : 2 => utilizará 2 endpoints
- bInterfaceClass : Mass Storage => Es un dispositivo de almacenamiento
- bInterfaceProtocol : Bulk-Only => Solo admite transferencias Bulk como es esperado de un pendrive.

Endpoint Descriptor: En este caso hay 2 endpoint descriptor

1. bEndpointAddress : 0x81 IN Endpoint:1
2. bEndpointAddress : 0x02 OUT Endpoint:2

Ambos con wMaxPacketSize : 512

Esto quiere decir que hay 2 endpoint para el dispositivo, uno para recibir paquetes y otro para enviarlos.

Paso 6: El host asigna y carga el device driver

A partir de los campos de Vendor ID, Product ID y Clases de los distintos descriptores, el sistema operativo decide qué driver es el más adecuado para el device detectado.

Por lo tanto, en este punto se producen una serie de transferencias de descriptores tipo STRING con datos como el Vendor o el producto.

53	0.809788	1.3.0	host	USB	96	GET_DESCRIPTOR	Response	CONFIGURATION
54	0.809837	host	1.3.0	USB	64	GET_DESCRIPTOR	Request	STRING
55	0.813518	1.3.0	host	USB	68	GET_DESCRIPTOR	Response	STRING
56	0.813559	host	1.3.0	USB	64	GET_DESCRIPTOR	Request	STRING
57	0.818292	1.3.0	host	USB	98	GET_DESCRIPTOR	Response	STRING
58	0.818327	host	1.3.0	USB	64	GET_DESCRIPTOR	Request	STRING
59	0.823180	1.3.0	host	USB	82	GET_DESCRIPTOR	Response	STRING
60	0.823238	host	1.3.0	USB	64	GET_DESCRIPTOR	Request	STRING
61	0.828177	1.3.0	host	USB	114	GET_DESCRIPTOR	Response	STRING

> Frame 59: 82 bytes on wire (656 bits), 82 bytes captured (656 bits) on interface usbmon0, id 0									
> USB URB									
▼ STRING DESCRIPTOR									
bLength: 18									
bDescriptorType: 0x03 (STRING)									
bString: Kingston									

0000	c0	80	db	4d	f9	8b	ff	ff	43	02	80	03	01	00	2d	00	..M...C.....
0010	c6	15	2f	65	00	00	00	00	f3	65	0b	00	00	00	00	00	..e....e.....
0020	12	00	00	00	12	00	00	00	00	00	00	00	00	00	00	00
0030	00	00	00	00	00	00	00	00	00	02	00	00	00	00	00	00
0040	12	03	4b	00	69	00	6e	00	67	00	73	00	74	00	6f	00	..K.i.n.g.s.t.o.
0050	6e	00															n.

Figura 2.11 - Get String Descriptor Response

Paso 7: El device driver selecciona una configuración.

El host envía al device un Set Configuration Request junto con el índice de la configuración deseada para establecerla como activa.

Una vez recibido el request, el device adopta la configuración solicitada y pasa al estado *Configured*.

62	0.828625	host	1.3.0	USB	64	SET_CONFIGURATION	Request
63	0.832511	1.3.0	host	USB	64	SET_CONFIGURATION	Response

> Frame 62: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface usbmon0, id 0									
> USB URB									
▼ Setup Data									
bmRequestType: 0x00									
0... .. = Direction: Host-to-device									
.00. = Type: Standard (0x0)									
...0 0000 = Recipient: Device (0x00)									
bRequest: SET_CONFIGURATION (9)									
bConfigurationValue: 1									
wIndex: 0 (0x0000)									
wLength: 0									

Figura 2.12 - Set Configuration Request

El device ahora se considera listo para enviar y recibir paquetes de datos a los endpoints configurados. En la figura se observan los BULK_in y los BULK_out que representan dichos paquetes de datos.

70	1.852558	host	1.3.1	USB	64 URB_BULK in
71	1.856097	1.3.1	host	USBMS	100 SCSI: Data In LUN: 0x00 (Inquiry Response Data) [SCSI transfer li
72	1.856140	host	1.3.1	USB	64 URB_BULK in
73	1.860315	1.3.1	host	USBMS	77 SCSI: Response LUN: 0x00 (Inquiry) (Good)
74	1.861001	host	1.3.2	USBMS	95 SCSI: Test Unit Ready LUN: 0x00
75	1.863834	1.3.2	host	USB	64 URB_BULK out
76	1.863916	host	1.3.1	USB	64 URB_BULK in
77	1.867924	1.3.1	host	USBMS	77 SCSI: Response LUN: 0x00 (Test Unit Ready) (Good)
78	1.868080	host	1.3.2	USBMS	95 SCSI: Read Capacity(10) LUN: 0x00
79	1.871626	1.3.2	host	USB	64 URB_BULK out
80	1.871656	host	1.3.1	USB	64 URB_BULK in
81	1.875531	1.3.1	host	USBMS	72 SCSI: Data In LUN: 0x00 (Read Capacity(10) Response Data)
82	1.875565	host	1.3.1	USB	64 URB_BULK in
83	1.879403	1.3.1	host	USBMS	77 SCSI: Response LUN: 0x00 (Read Capacity(10)) (Good)
84	1.879563	host	1.3.2	USBMS	95 SCSI: Mode Sense(6) LUN: 0x00
85	1.883329	1.3.2	host	USB	64 URB_BULK out
86	1.883406	host	1.3.1	USB	64 URB_BULK in
87	1.887237	1.3.1	host	USBMS	100 SCSI: Data In LUN: 0x00 (Mode Sense(6) Response Data)
88	1.887277	host	1.3.1	USB	64 URB_BULK in
89	1.891034	1.3.1	host	USBMS	77 SCSI: Response LUN: 0x00 (Mode Sense(6)) (Good)

Figura 2.13 - URB_BULK in and out

3 Anexo

Para el desarrollo del presente informe se utilizaron las siguientes fuentes de información:

- Peacock, C. (2002) USB in a nutshell - Making Sense of the USB Standard.
Disponible en: <https://sge.frba.utn.edu.ar/upload/Materias/95-0435/archivos/usb-in-a-nutshell.pdf> (Accedido: 26 de octubre del 2023).
- Ing. Ridolfi, P. (2011) Universal Serial Bus – Técnicas Digitales II - UTN FRH
Disponible en: https://gitlab.frba.utn.edu.ar/td_piloto/td2_r4054/publico/-/blob/master/slides/Clase14-2020-11-03-USB/Universal%20Serial%20Bus.pdf?ref_type=heads (Accedido: 26 de octubre del 2023)