TUGAS AKHIR

UJIAN KOMPREHENSIF

Tugas Akhir ini Diajukan Guna Memenuhi Sebagian Persyaratan
Memperoleh Gelar Sarjana Teknik (S-1)
Fakultas Teknik Universitas Muhammadiyah Surakarta

Disusun Oleh :

NAMA : TAUFIK RAHMAD HIDAYANTO

NIM : D 400 010 135

FAKULTAS TEKNIK JURUSAN ELEKTRO

UNIVERSITAS MUHAMMADIYAH SURAKARTA

2009

BAB I

KOMUNIKASI KOMPUTER DENGAN PERANGKAT LUAR MELALUI UNIVERSAL SERIAL BUS (USB)

1.1 Pendahuluan

Universal Serial Bus (USB) adalah salah satu standar interkoneksi antara komputer dengan peralatan eksternal yang mampu mendukung kecepatan sampai 480 Mbps. (bandingkan dengan serial yang hanya berkecepatan 20 Kbps). USB adalah port yang sangat diandalkan saat ini karena bentuknya yang kecil dan kecepatan transfernya yang tinggi. Sehingga dapat menghubungkan hingga 127 produk USB dalam satu komputer. Didalam bab II akan dijelaskan tentang mekanikal USB, pada bab III akan dijelaskan tentang elektrikal USB, pada bab IV akan dijelaskan tentang protokol-protokol USB dan bab V akan dijelaskan contoh pengiriman data melalui USB.

USB mempunyai beberapa kelebihan, diataranya :

a . penggunaannya mudah.
   Cukup tancapkan *peralatan* USB ke konektor USB, computer akan langsung mendeteksi adanya peralatan tersebut, tanpa perlu me-*restart* komputer.

b . mendukung 3 tipe kecepatan, yaitu *low*, *full* dan *high speed*.

c . adanya powerdown (*suspend*).
   apabila tidak digunakan, secara otomatis, *peralatan* USB akan mengalami suspend, sehingga konsumsi daya bisa lebih kecil.

d . USB mensuply daya ke peralatan USB dengan arus sebesar 500 mA.
   Sehingga apabila sebuah *peralatan* memerlukan daya sebesar 500 mA, maka *peralatan* tersebut tidak memerlukan tambahan daya

e . USB bersifat *multiplatform.* USB mendukung hampir semua sistem operasi.

Kelemahan pada USB adalah :
panjang kabel untuk koneksi ke USB relatif pendek apabila dibandingkan *interface* lain.

Perbandingan USB dengan *interface* lain :

Interface	**Jumlah Peralatan Maksimum**	**Panjang Kabel (kaki)**	**Kecepatan (bps)**	**Fungsi Peralatan**
USB	127	16	1.5M, 12M, 480M	*Mouse, keyboard, modem, printer, scanner, hard disk,* dll
RS-232	2	50-100	20K	*Modem, mouse*
RS-485	32	4.000	10M	Sistem akuisisi data dan kendali
IrDA	2	6	115K	*Printer*
Port Paralel	2, atau 8	10-30	8M	*Printer, scanner*
IEEE-1394 (FireWire)	64	15	400M	*Video*
ISA		-	5M	Kartu suara, *modem,* LAN
PCI			524M	Kartu suara, *modem,* LAN, *video,* dll

Tabel 1. Perbandingan perangkat USB dengan perangkat lain.

Umumnya, konfigurasi sistem USB terdiri atas sebuah *host* (dalam hal ini : komputer) dan beberapa peralatan USB yang dihubungkan melalui kabel USB. Host memiliki sebuah hub terintegrasi yang disebut *root hub*. *Root hub* memiliki 2 buah port USB.

Host bertanggung jawab terhadap transfer data pada bus, sehingga host harus mampu mendeteksi peralatan apa saja yang terhubung dan kemampuan peralatan tersebut.

USB 1.1 mendukung 2 type kecepatan, yaitu *full-speed* pada 12 Mbps dan *low-speed* pada 1.5 Mbps. Sementara USB 2.0 mendukung kecepatan sampai dengan 480 Mbps yang dikenal dengan *High-Speed Mode*.

Low Speed - Peralatan interaktif - 10 – 100 kbps	Keyboard. Mouse, peralatan buat game	Cost sangat rendah, mudah digunakan, dinamik attach-dettach, multiple peripheral
Full Speed - Telepon, Audio, Kompresi Video - 500 kbps – 10 Mbps	Broadband, audio, mikrophone	Cost rendah, mudah digunakan, dinamik attach-dettach, multiple peripheral,
High Speed - Video, media penyimpan data - 25 – 400 Mbps	Video, harddisk, imaging, broadband	Cost sangat rendah, mudah digunakan, dinamik attach-dettach, multiple peripheral

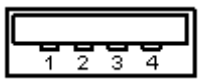
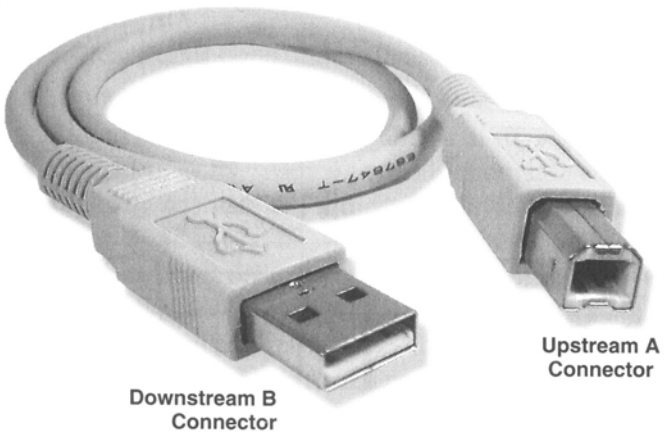
Tabel 2. Jenis kecepatan USB dan penggunaannya.

1.2 Mekanikal

Pada bab ini akan menjelaskan tentang mekanikal dan elektrikal untuk kabel, konektor dan perakitan untuk peralatan USB.

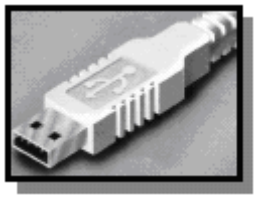
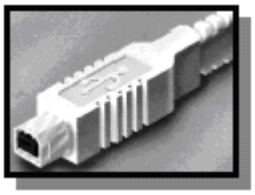
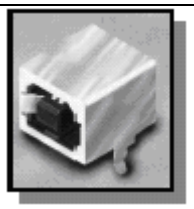
Perangkat *high-speed* dan perangkat *full-speed* memerlukan kabel berpelindung (*shielded*) untuk penghantar daya dan kabel yg melilit (*twisted*) untuk penghantar sinyal. Perangkat low-speed juga sama, tapi tidak memerlukan kabel lilit (*twisted*) sebagai penghantar sinyal.

USB mempunyai 2 tipe konektor, yaitu konektor tipe A dan konektor tipe B. konektor tipe A terhubung ke host secara upstream sementara konektor tipe B terhubung ke peralatan secara downstream. Kedua konektor tersebut terhubung oleh sebuah kabel USB.

		
konektor tipe A	konektor tipe B	kabel USB

Tabel 3. Konektor USB dan kabel USB.

Wujud fisik dari kedua konektor adalah sebagai berikut :

Konektor seri A	Konektor seri B
 Dari perangkat USB	 Dari Host
 Dari USB Host atau Hub	 Input ke perangkat USB atau Hub

Tabel 4. Wujud fisik konektor USB.

Kabel USB terdiri empat penghantar, 2 penghantar daya dan 2 penghantar sinyal.

Standar warna kabel USB adalah sebagai berikut :

Nomor PIN	Warna Kabel	Fungsi
1	Merah	V_{bus} (+5V)
2	Putih	D-
3	Hijau	D+
4	Hitam	Ground

Tabel 5. Standar warna kabel USB

Pada koneksi USB dengan kecepatan 12 Mb/detik, digunakan kabel *twisted-pair* dengan impedansi 90 Ω ± 15% dan delay maksimumnya 26 ns. Impedansi pada driver-nya harus bernilai antara 28 Ω hingga 44 Ω. Jika diukur, arus yang masuk dan keluar perangkat USB 1.1 tidak boleh melebihi 10,71 V_{OH}mA. Tegangan logika yang dimasukkan ke D+ dan D- tidak boleh

melebihi 0,3 V_{OH} untuk logika rendah dan harus turun sebesar 0,7 V_{OH} untuk logika tinggi.

Karena ada perangkat USB yang berkomunikasi pada kecepatan rendah (1,5 MHz), kombinasi kabel dan perangkat USB harus mengandung kapasitas tunggal dengan nilai 200pF hingga 450 pF di pin D+ dan D-. perambatan delay pada kabel kecepatan rendah harus kurang dari 18 ns. Data sinyal dengan toleransi 10%. Waktu pelaksanaanya berkisar dari 4 ns hingga 20 ns, tergantung kecepatan USB yang digunakan.

Impedansi input di D+ dan D- tanpa terminasi lebih besar dari 300 kΩ. Input kapasitansi port yang diukur pada pin konektor beloh berbeda pada port upstream dan downstream, namun kapasitansi maksimum di downstream adalah 150 pF pada D+ dan D-. nilai ini terdiri dari 75 pF (maksimum) kapasitansi terhadap ground di transceiver dan di konektor, plus tambahan 75 pF kapasitas di masing-masing konduktor di jalur transmisi antara pengirim dan penerima. Untuk upstream, kapasitansi maksimumnya adalah 100 pF di D+ dan D- sehingga tambahan kapasitornya hanya sebesar 25 pF.

1.3 Kelistrikan

Pada bab ini akan dijelaskan tentang kelistrikan dari USB, dimana berisi tentang pensinyalan, pendistribusian daya.

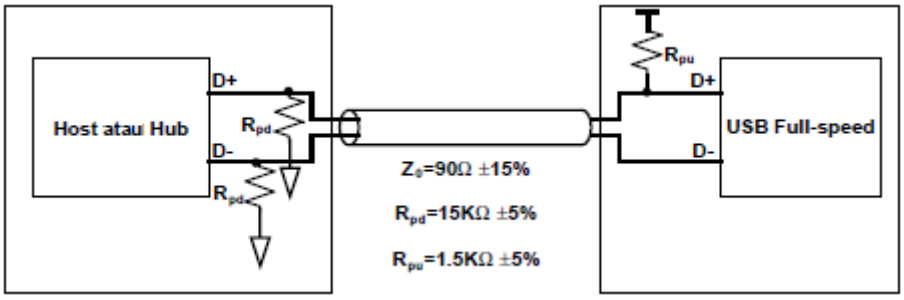
Rentang tegangan kerja sinyal USB adalah 0,3 volt sehingga 3,6 volt dengan bagian beban 1,5 kΩ. Logika tinggi didapat jika tegangan sudah melebihi 2,8 volt terhadap ground pada beban 1,5 kΩ. Pada perangkat USB yang berkecepatan *low-speed* dan *full-speed*, diferensial 1 dikirim dengan menarik D+ hingga lebih besar dari 2,8 volt menggunakan sebuah resistor 1,5 kΩ yang terhubung de ground sambil sekaligus menarik D- hingga di bawah 0,3 volt menggunakan sebuah resistor 1,5 kΩ yang terhubung ke tegangan 3,6 volt. Hal yang sama pada diferensial 0 dilambangkan dengan D- yang menggunakan tegangan lebih besar dari 2,8 volt dan D+ yang menggunakan tegangan lebih rendah dari 0,3 volt. Konfigurasi ini didapatkan dengan resistor *pull-up* dan *pull-down* yang sama.

Dibagian penerima, diferensial 1 didefinisikan sebagai D+ dengan level tegangan lebih besar sebanyak 200 mV dari D- dan diferensial 0 sebagai D+ dengan level tegangan 200 mV lebih kecil daripada D-. pada USB berkecepatan tinggi (480 Mbps) digunakan sumber arus tetap 17,78 mA untuk mengurangi noise.

Data dikirim secara serial sehingga perangkat USB harus mampu menangani gelombang kontinyu. Gelombang ini dihubungkan langsung ke pin data USB dari sebuah sumber tegangan dengan impedansi output 39 Ω. Kemungkinan terburuk konfigurasi sumber tegangan rangkaian terbuka ini adalah adanya *overshoot* dan *undershoot*.

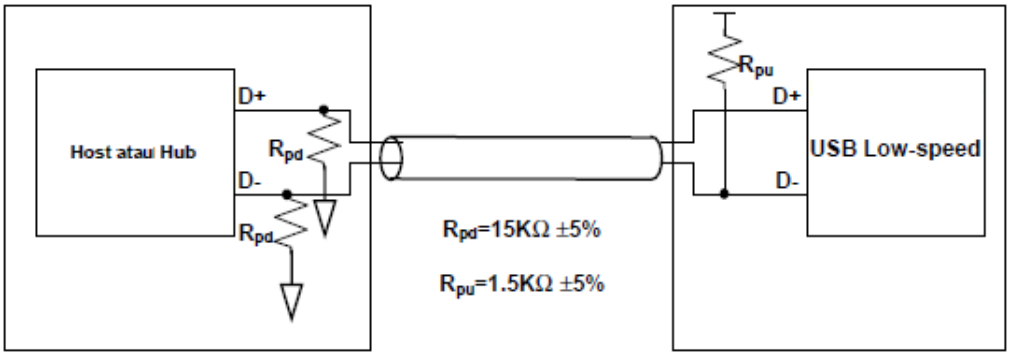
Perangkat USB kecepatan tinggi dan rendah dibedakan oleh posisi resistor *pull-up* di ujung kabel *downstream.* Penggunaan resistor ini digunakan Host / Hub untuk mengenali perangkat yang terhubung kepadanya, sehingga Host atau Hub akan tahu perangkat tersebut berkecepatan *low* atau *full-speed.*

a. Perangkat *full-speed* diterminalkan dengan resistor *pull-up* terhubung di D+ seperti gambar dibawah.



Gambar 1. Gambar resistor pada perangkat USB *full-speed.*

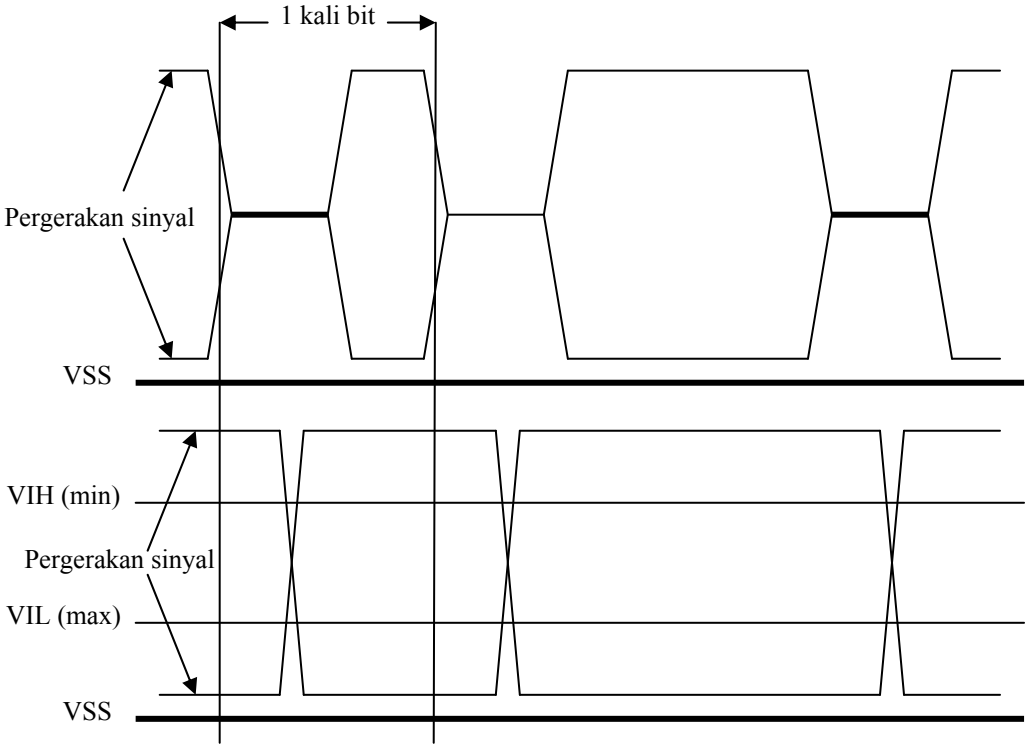
b. Perangkat *low-speed* diterminalkan dengan resistor *pull-up* terhubung di D- seperti gambar dibawah.



Gambar 2. Gambar resistor pada perangkat USB *low-speed.*

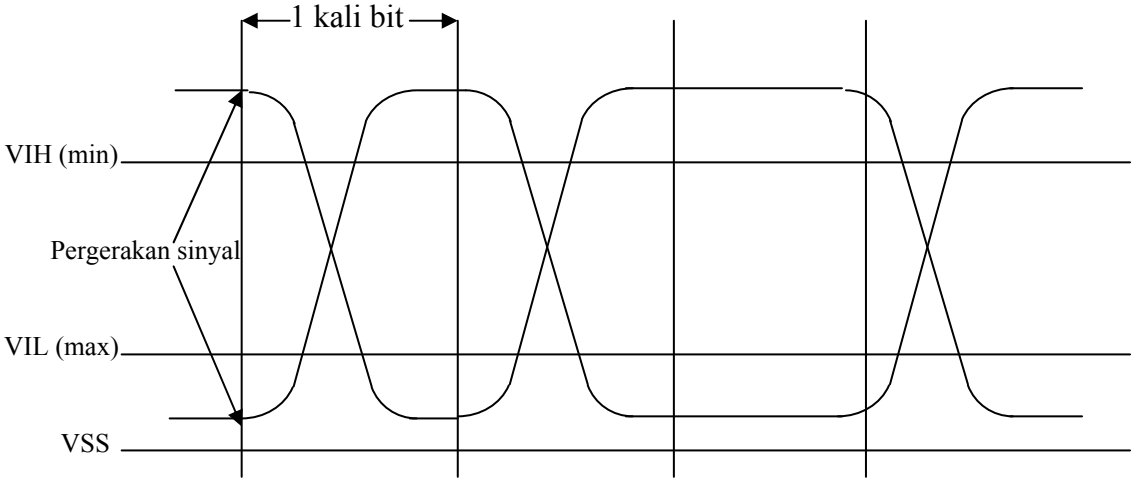
Karakteristik pergerakan sinyal USB dapat dilihat pada gambar dibawah ini :

a. karakteristik sinyal USB *full-speed.*



Gambar 3. Karakteristik sinyal pergerakan USB *full-speed.*

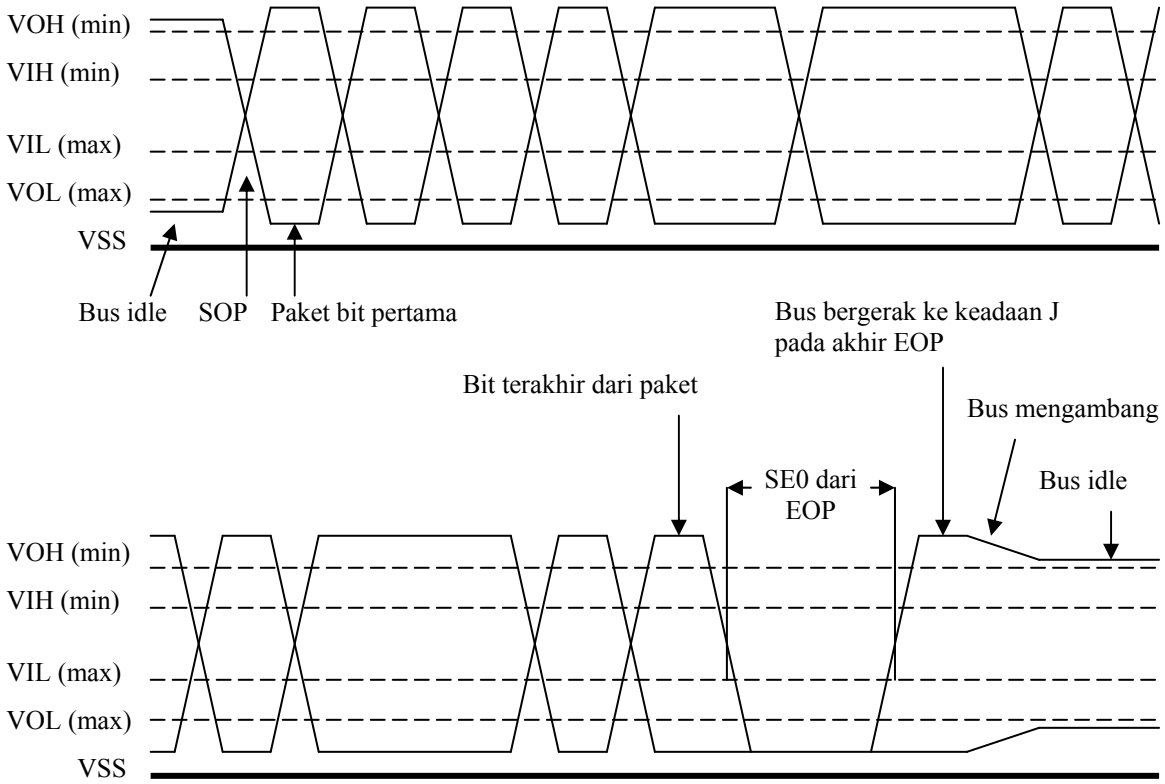
b. karakteristik sinyal USB *low-speed.*



Gambar 4. Karakteristik sinyal pergerakan USB *low-speed.*

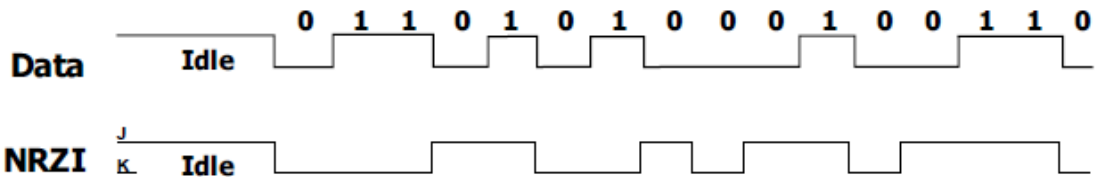
Pengiriman data pada USB terjadi jika adanya diferensial sinyal. Pada *Start of Packet* (SOP) di picu dengan dimulainya menggerakkan D+ dan D- dari keadaan *idle* ke keadaan K.

Keadaan SE0 digunakan untuk memicu terjadinya *End-of-Packet* (EOP). EOP akan terjadi dengan menggerakkan D+ dan D- pada keadaan SE0 sebanyak 2 kali bit, diikuti dengan menggerakkan bus ke keadaan J untuk 1 kali bit. Transaksi dari SE0 ke keadaan J mendefinisikan akhir dari paket pada receiver.



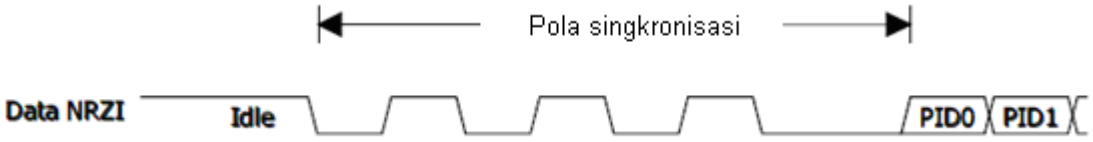
Gambar 5. Level tegangan SOP dan EOP pada perangkat *low* atau *full-speed.*

Pengiriman paket data USB dikodekan dengan NRZI (*Non Return to Zero Invert*). Dimana dalam NRZI, 1 adalah mempresentasikan tidak adanya perubahan level tegangan, dan 0 adalah mempresentasikan perubahan level tegangan. Gambar dibawah ini mengilustrasikan konversi *stream data* menjadi NRZI data.



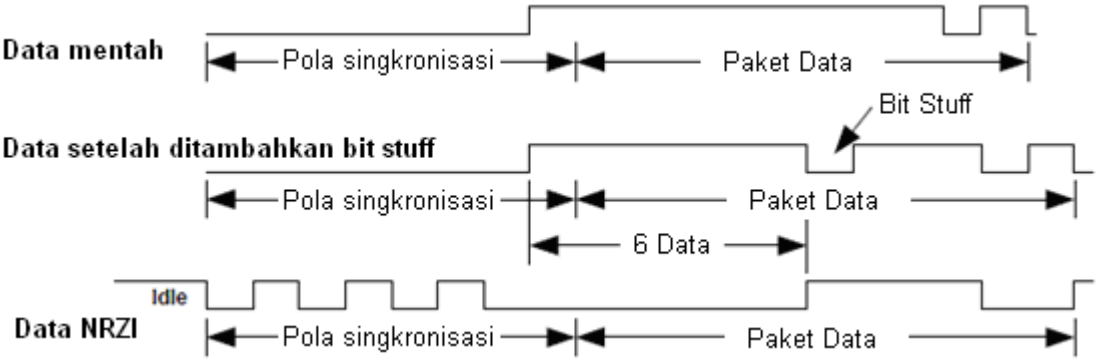
Gambar 6. Pola kode NRZI.

Pola singkronisasi (*Sync Pattern)* digunakan untuk awalan dalam pentransmisian data, dimana digunakan untuk singkronisasi awal pengiriman paket data. Format data yang dikirim adalah 3 keadaan K dan 3 keadaan J dan diikuti 2 keadaan K, sehingga terdiri 8 langkah. Gambar dibawah ini adalah contoh awalan pengiriman paket data.

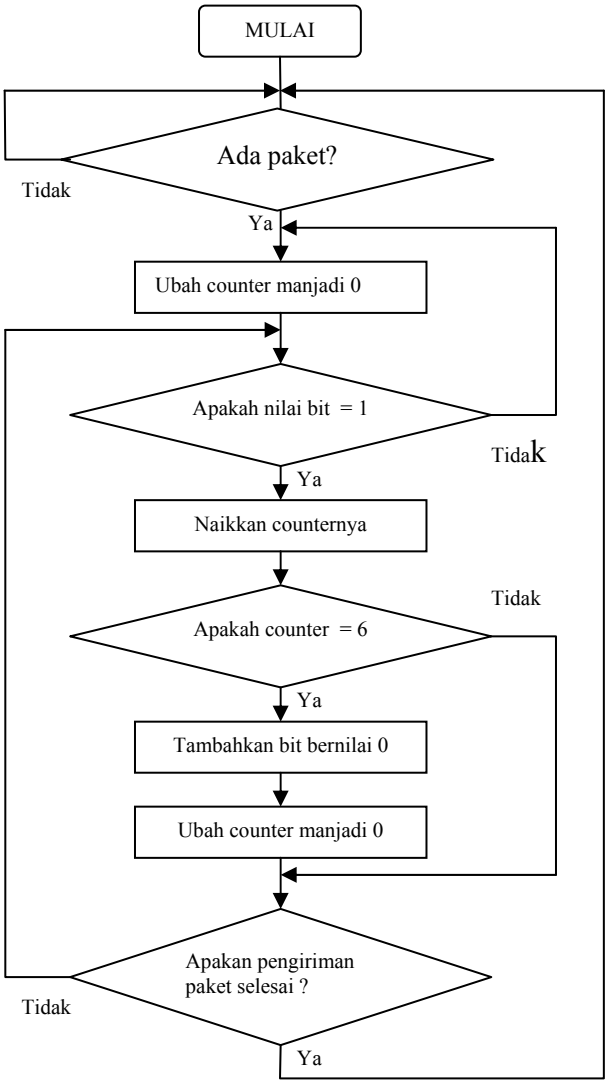


Gambar 7. Pola singkronisasi.

Untuk memastikan transisi sinyal yang sinkron antara host dengan perangkat USB, *bit stuffing* ditambahkan ketika dalam proses pengiriman paket data USB. Level tegangan 0 dimasukkan, setelah 6 data berurutan pertama pada *data stream* sebelum data di encode oleh NRZI. Bit stuffing aktif ketika *data stream* diawali dengan *Sync Pattern*.



Gambar 8. Pola singkronisasi dan penambahan bit stuff.



Gambar 9. Gambar flow chart penambahan bit stuff.

1.4 Protokol pada Universal Serial Bus

Setiap transaksi USB mengandung

a. *Token Packet* (Header yang mendefinisikan apa yang mengikuti)
b. *Optional Data Packet* (Mengandung *data payload*)
c. *Status Packet* (Digunakan untuk status paket atau status kesalahan)

USB berpusat di host, host menginisialisasi semua transaksi. Paket pertama adalah sebuah token yang diaktifkan oleh host untuk mendeskripsikan apa yang mengikuti, dan menentukan transaksi apakah membaca atau menulis ke perangkat. Paket berikutnya adalah paket pembawa *data payload* yang diikuti paket *handshaking*, melaporkan data atau *token* telah diterima dengan sukses atau tidak.

1. **Urutan Pengiriman Data**

   Data bit pada USB dikirim pertama adalah bit *Least-significant bit* (LSb) kemudian *most-significant bit* (MSb).

2. **SYNC Field**

   Semua paket Harus dimulai oleh *sync field*. *Sync field* tersebut panjangnya 8 bits, dimana digunakan untuk singkronisasi clock dari *receiver* dan *transmiter*. 2 bit terakhir mengindikasikan bahwa field PID dimulai.

3. **PID**

   PID digunakan untuk mengidentifikasi tipe paket yang sedang dikirim.

GROUP	Nilai PID	Paket identifikasi
Token	0001	OUT Token
	1001	IN Token
	0101	SOF Token
	1101	Setup Token

Data	0011	DATA0
	1011	DATA1
	0111	DATA2
	1111	MDATA
Handshake	0010	ACK Handshake
	1010	NAK Handshake
	1110	STALL Handshake
	0110	NYET (Belum merespon)
Special	1100	PREamble
	1100	ERR
	1000	Split
	0100	Ping

Tabel 6. Daftar jenis PID.

Tertera paket PID tersebut sebanyak 4 bit, untuk meyakinkan diterima dengan benar, 4 bit tersebut di komplemenkan dan diulangi, sehingga membentuk 8 bit PID. Formatnya adalah sebagai berikut.

PID0	PID1	PID2	PID3	nPID0	nPID1	nPID2	nPID3

Tabel 7. Pola lengkap PID.

4. **ADDR**

   Fied Address sangatlah spesifik, dengan panjang 7 bit sehingga memperbolehkan terhubung dengan 127 perangkat. *Address* 0 adalah tidak benar, sehingga jika ada perangkat yang belum terdefinisikan ke host, harus mengirimkan *Address* 0

5. **ENDP**

   Field endpoint terdiri dari 4 bit, yang memungkinkan mempunyai *endpoint* sebanyak 16. perangkat USB *low-speed* hanya mempunyai 2 endpoint, tambahan Address dan *default pipe*. (maksimal 4 endpoint)**.**

6. **CRC**

   Cyclic Redundancy Checks adalah sistem pengecekan paket *payload*. Semua paket token mempunyai 5 bit CRC dan paket data mempunyai 16 bit CRC

7. **EOP**

   End of paket. Disignal dengan *Single Ended Zero* (SE0) kira kira 2 waktu pensinyalan yang diikuti kondisi J untuk 1 kali bit.

USB mempunyai 4 perbedaan tipe paket. Token paket mengindikasikan tipe transaksi yang mengikuti, data paket mengandung data *payload*, paket *handshake* digunakan untuk menjawab (*acknowledging*) data atau melaporkan kesalahan, dan *Start of Frame* mengindikasikan dimulainya frame baru.

1. **Paket Token**

   Paket token terdiri dari :

   - **In**, menginformasikan perangkat USB bahwa host akan membaca informasi
   - **Out**, menginformasikan perangkat USB bahwa host akan mengirimkan informasi
   - **Setup**, digunakan untuk memulai *control transfer*

Sync	PID	ADDR	ENDP	CRC5	EOP

Tabel 8. Pola Paket Token..

2. **Paket Data**

   Paket data terdiri atas 2 jenis, dimana masing masing dapat mengirimkan data 0 sampai 1023 byte.

   - Data0
   - Data1

Sync	PID	Data	CRC16	EOP

Tabel 9. Pola lengkap Paket Data.

3. Paket Handshake

Terdapat 3 jenis tipe paket *handshake* :

- **ACK**, *Acknowledment* dimana paket telah sukses diterima
- **NAK**, *Not Acknowled* melaporkan bahwa perangkat tidak dapat mengirimkan atau menrima data. Juga digunakan untuk transaksi interupsi untuk menginformasikan host bahwa tidak ada data untuk dikirim.
- **STALL**, perangkat menemukan keadaan dimana memerlukan intervensi dari host.

Sync	PID	EOP

Tabel 10. Pola paket *handshake*.

4. Start of Frame

Paket SOF mengandung nomer frame 11 bit yang dikirim oleh host setiap kurang lebih 1ms ±500ns

Sync	PID	Nomer Frame	CRC5	EOP

Tabel 11. Pola *Start of Frame*.

USB mempunyai 4 jenis tipe transfer / endpoint, yaitu:

- *Control Transfer*
- *Interrupt Transfer*
- *Isochronous Transfer*
- *Bulk Transfer*

1. Control Transfer

Control transfer pada umumnya digunakan untuk perintah (*command*) dan status operasi. Panjang transfer paket control pada perangkat *low-speed* harus 8 byte, pada perangkat *high-speed* memperbolehkan dengan ukuran 8, 16, 32, 64 byte, dan perangkat *full-speed* harus mempunyai ukuran 64 byte

2. Interrupt Transfer

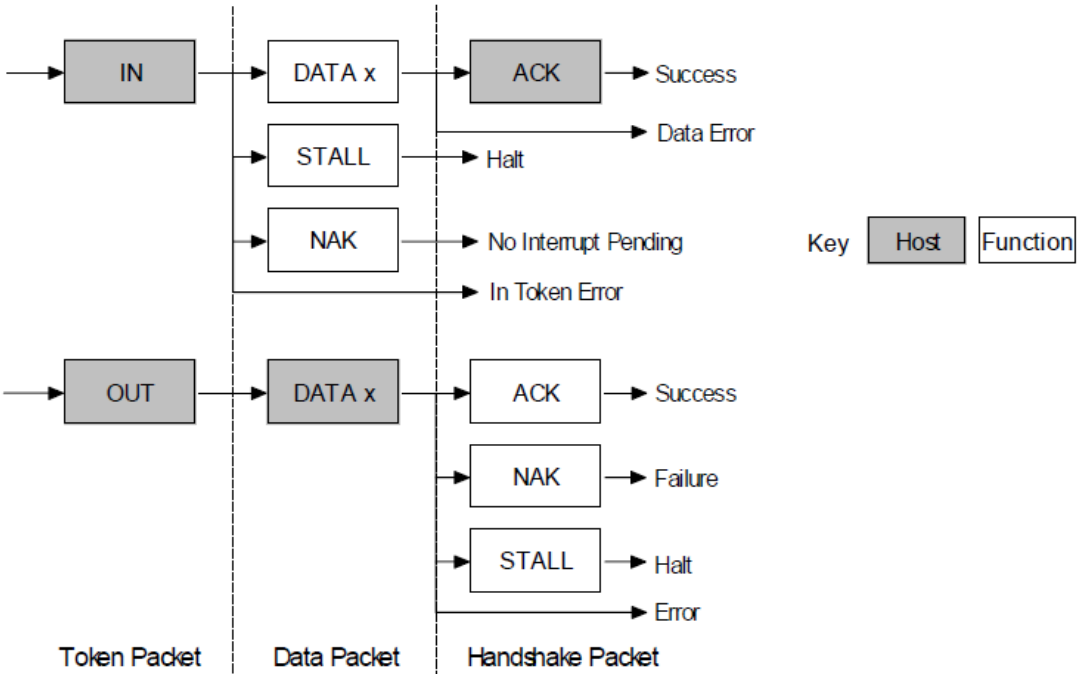
Jika perangkat USB memerlukan perhatian dari host, maka perangkat USB harus menunggu sampai host melakukan *polling*, sebelum dapat melaporkan bahwa memerlukan perhatian khusus.

Keunggulan *Interrupt Transfer*

- Menjamin pengiriman data
- *Stream Pipe* yang *undirectional*
- Deteksi error

Interrupt Transfer pada umumnya non-periodik. Sebuah permintaan interrupt adalah antrian oleh perangkat sampai host melakukan *polling* ke perangkat USB untuk menanyakan data

- Maksimal ukuran data *payload* perangkat *low-speed* adalah 8 byte.
- Maksimal ukuran data *payload* perangkat *full-speed* adalah 64 byte.
- Maksimal ukuran data *payload* perangkat *high-speed* adalah 1024 byte.



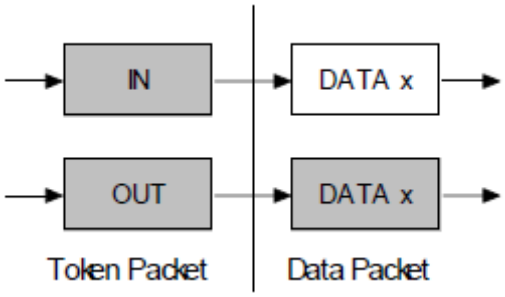
Gambar 10. Transaksi *Interrupt* IN dan OUT.

3. Isochronous Transfer

Isochronous transfer terjadi secara berkesinambungan dan periodik. Pada dasarnya mengandung informasi waktu yang sangat sensitif, sebagai contoh pada *audio* dan *video stream*. Jika terjadi *delay* atau pengulangan dari data pada *audio stream*, maka akan terjadi adanya data yang tidak sempurna sehingga terjadi adanya suara yang kurang enak didengar.

- Jaminan untuk akses *bandwidth* USB
- *Stream Pipe* yang *undirectional*
- Kecepatan pengiriman
- Deteksi kesalahan melalui CRC, tapi tidak menjamin pengiriman data
- Hanya untuk perangkat *full* dan *high speed*

Maksimal ukuran data *payload* di spesifikasikan pada *endpoint descriptor* dari *isochronous endpoint*.



Gambar 11. Transaksi *isochronous*.

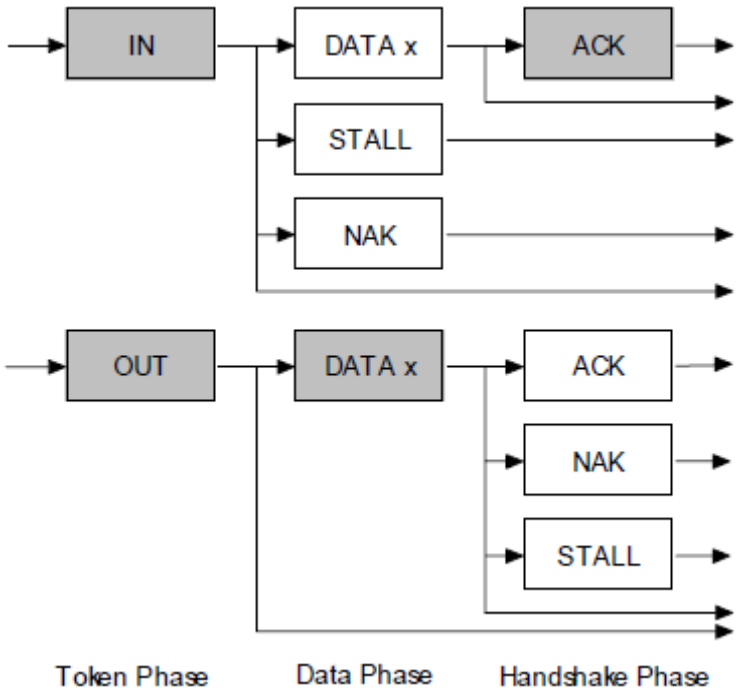
Diagram diatas menjelaskan bahwa transaksi IN/OUT isochronous tidak mempunyai kondisi *handshaking* dan tidak dapat melaporkan terjadinya kesalahan.

4. Bulk Transfer

Bulk transfer dapat digunakan untuk pentransferan data yang banyak. Sebagai contoh pada printer atau scanner. Bulk transfer menghasilkan sistem koreksi kesalahan pada field CRC16 pada data *payload* dan deteksi kesalahan akan dikirim ulang untuk meyakinkan data diterima dengan benar tanpa kesalahan.

Bulk transfer tidak mementingkan kecepatan transfer, karena hanya mengutamakan pentransferan data terkirim dengan benar.

- Digunakan untuk transfer data yang besar atau banyak.
- Deteksi kesalahan melalaui CRC16 dan menjamin data terkirim.
- Tidak ada jaminan *bandwidth* dan kecepatan yang minimal
- *Stream Pipe* yang *undirectional*.
- Hanya pada perangkat *full* dan *high speed*.



Gambar 12. Transaksi *Bulk*.

USB Descriptors

Semua perangkat USB mempunyai deskripsi, yang mana mendeskripsikan informasi ke *host*, sebagai contoh termasuk perangkat apa, siapa pembuatnya, versi USB yg didukung, berapa banyak perangkat tersebut dapat dikonfigurasikan, jumlah *endpoint* dan lain sebagainya.

USB Descriptor diantaranya adalah sebagai berikut :

- *Device Descriptors.*
- *Configuration Descriptors.*
- *Interface Descriptor.*
- *Endpoint Descriptors.*
- *String Descriptors.*

Perangkat USB hanya dapat mempunya 1 *device descriptor*. *Device descriptor* berisi informasi, sebagai contoh USB *revision*, Produk dan vendor ID digunakan untuk memprioritaskan driver dan jumlah configurasi yang dimiliki. Jumlah konfigurasi mengindikasikan berapa banyak percabangan konfigurasi *descriptor* yang mengikuti.

Semua *descriptor* mempunya standar format. Byte pertama berisi panjang *descriptor*, byte kedua berisi tipe *descriptor*. Jika panjang dari *descriptor* lebih kecil dari apa yang telah didefinisikan, kemudian *host* akan mengabaikannya. Jika ukuran lebih besar dari yang diharapkan, host akan mengabaikan ekstra byte dan memulai mencari *descriptor* berikutnya.

Offset	Field	Ukuran	Nilai	Keterangan
0	bLength	1	Angka	Ukuran descriptor dalam byte
1	bDescriptorType	1	Konstan	Tipe Descriptor
2	bcdUSB	2	BCD	USB Start dari parameter descriptor

Tabel 12. Data pada *USB Descriptors*.

1. **Device Descriptor**

   Device Descriptor pada USB mempresentasikan seluruh perangkat. Perangkat USB hanya dapat mempunyai 1 *device descriptor*, seperti informasi penting tentang perangkat, sebagai contoh versi USB yg didukung, Ukuran maksimal paket, Vendor dan Produk ID, dan jumlah yg mungkin dapat dikonfigurasikan oleh perangkat.

Offset	Field	Ukuran	Nilai	Keterangan
0	bLength	1	Angka	Ukuran descriptor dalam byte (18 byte)
1	bDescriptorType	1	Konstan	Tipe Descriptor (0x01)
2	bcdUSB	2	BCD	Nomer spesifik dimana perangkat berjalan
4	bDeviceClass	1	Class	Class Code Jika sama dengan 0, masing masing spesifik interface milij dari class code Jika sama dengan 0xFF, Class Code adalah spesifik vendor. Lainnya adalah Class Code yang benar
5	bDeviceSubClass	1	SubClass	SubClass Code (Dikeluarkan oleh USB.org)

6	bDeviceProtocol	1	Protocol	Protocol Code (Dikeluarkan oleh USB.org)
7	bMaxPacketSize	1	Angka	Ukuran maksimal paket untuk Zero Endpoint, ukuran yang benar adalah 8, 16, 32, 64
8	idVendor	2	ID	Vendor ID (Dikeluarkan oleh USB.org)
10	idProduct	2	ID	Product ID (Dikeluarkan oleh pembuat perangkat)
12	bcdDevice	2	BCD	Angka Device Release
14	iManufacturer	1	Index	Indek dari Manufacturer String Descriptor
15	iProduct	1	Index	Indek dari Product String Descriptor
16	iSerialNumber	1	Index	Indek dari Serial Number String Descriptor
17	bNumConfiguration	1	Nilai angka	Jumlah Konfigurasi yang memungkinkan

Tabel 13. Data pada paket Device Descriptors.

- Field **bcdUSB** melaporkan versi tertinggi dari perangkat USB yg mendukung. Nilainya adalah *binary coded decimal* dengan format 0xJJMN, dimana JJ adalah nomer versi mayor, M adalah nomer versi minor dan N adalah nomer sub versi minor. Sebagai contoh USB 2.0 akan melaporkan 0x0200, USB 1.1 adalah 0x0110 dan USB 1 adalah 0x0100
- **bDeviceClass**, **bDeviceSubClass** dan **bDeviceProtocol** digunakan oleh sistem operasi untuk mencari *class driver* untuk perangkat USB. Pada umumnya hanya hanya bDeviceClass yg digunakan.
- **bMaxPacketSize** melaporkan maksimal ukuran paket untuk *endpoint zero*. Semua perangkat harus mendukung *endpoint zero*.
- **idVendor** dan **idProduct** digunakan sistem operasi untuk mencari driver untuk perangkat USB. Vendor ID ditentukan oleh USB-IF.
- **bcdDevice** mempunyai format yang sama dengan bcdUSB dan digunakan untuk menghasilkan nomer versi perangkat. Nilai ini ditentukan oleh pembuat perangkat.
- **string descriptors** mengandung informasi lengkap tentang pembuat perangkat, yaitu produk dan serial number. Disini tidak ada keharusan untuk mempunyai string descriptors. Jika tidak ada string descriptor, maka harus di set dengan indek 0.
- **bNumConfigurations** mendefinisikan jumlah konfigurasi perangkat yg didukung.

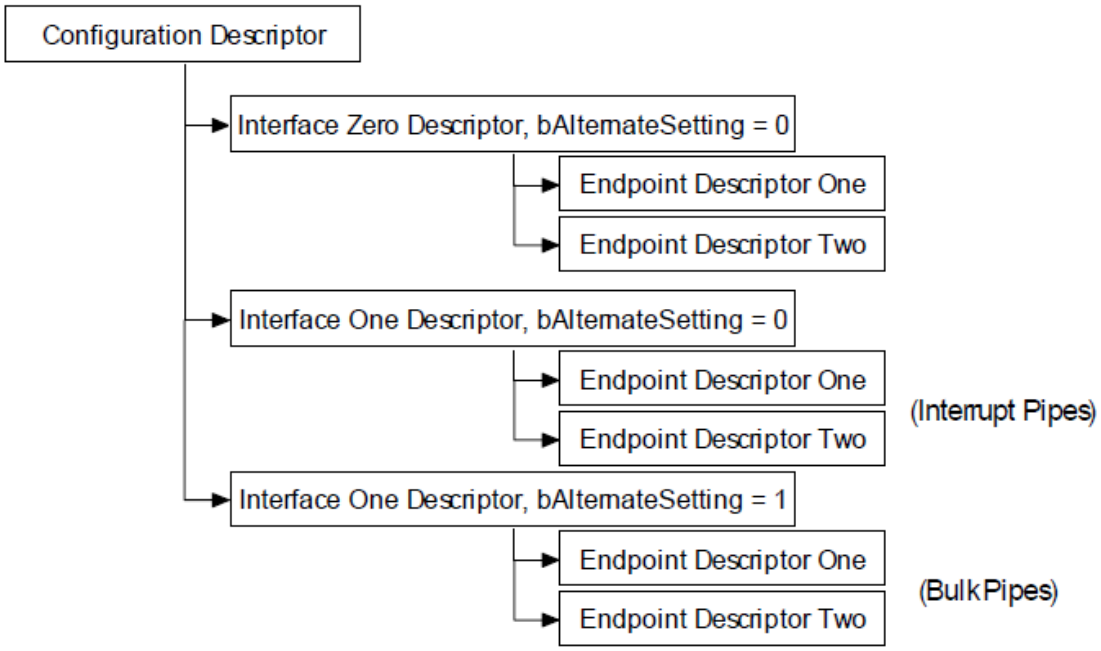
2. Configuration Descriptors

Suatu perangkat USB dapat mempunyai beberapa perbedaan konfigurasi, walaupun mayoritas perangkat USB hanya mempunyai satu konfigurasi. Konfigurasi descriptor berisi informasi bagaimana perangkat mendapat suplay tegangan, jumlah maksimal daya yg dibutuhkan dan jumlah *interface* yang dimiliki. Olehkarena itu sangatlah memungkinkan mempunyai 2 konfigurasi.

Pertama, semua konfigurasi telah diuji oleh *host*, host akan mengirim perintah *SetConfiguration* dengan nilai bukan 0 dimana sesuai dengan bConfigurationValue.

Offset	Field	Ukuran	Nilai	Keterangan
0	bLength	1	Nilai	Ukuran descriptor dalam byte
1	bDEscriptorType	1	Konstan	Configuration Descriptor (0x02)
2	wTotalLength	2	Nilai	Total panjang data yang dihasilkan
4	bNumInterfaces	1	Nilai	Jumlah interface
5	bConfigurationValue	1	Nilai	Argumen nilai untuk memilih konfigurasi
6	iConfiguration	1	Indek	Indek dari string descriptor yang mendeskripsikan konfigurasi ini
7	bmAttributes	1	Bitmap	D7 bus powered D6 Self powered D5 Remote wakeup D4..0 Reserved (0)
8	bMaxPower	1	mA	Maksimal daya yg dikonsumsi

Tabel 14. Data pada paket *Configuration Descriptors*.



Gambar 13. *Configuration Descriptor*.

- **bNumInterface**, spesifikasi jumlah *interface* yg ditampilkan untuk konfigurasi ini
- **bConfigurationValue**, digunakan untuk permintaan *SetConfiguration* untuk memilih konfigurasi
- **iConfiguration**, adalah indek ke *string descriptor* yg mendeskripsikan *dari user interface* sehingga bisa dibaca.
- **bmAttributes**, adalah berisi parameter daya dari konfigurasi. Jika perangkat *self powered*, maka akan men-set D6. bit D7 telah digunakan pada USB 1.0 untuk mengindikasikan perangkat telah mendapat daya
- **bMaxPower**, mendefinisikan nilai maksimal daya yg digunakan perangkat. Kira kira adalah 2mA dan maksimal adalah 500mA.

3. Interface Descriptor

Interface descriptor bisa dilihat sama seperti *header* atau pengelompokan pada endpoint.

Offset	Field	Ukuran	Nilai	Keterangan
0	bLength	1	Nilai	Ukuran descriptor dalam byte
1	bDEscriptorType	1	Konstan	Configuration Descriptor (0x04)
2	bInterfaceNumber	1	Nilai	Jumlah interface
3	bAlternateSetting	1	Nilai	Nilai digunakan untuk memilih alternatif seting
4	bNumEndpoint	1	Nilai	Jumlah endpoint yang digunakan untuk interface ini
5	bInterfaceClass	1	Class	Class Code (ditentukan oleh USB.org)
6	bInterfaceSubClass	1	SubClass	SubClass Code (ditentukan oleh USB.org)
7	bInterfaceProtocol	1	Protocol	Protocol Code
8	iInterface	1	Indek	Indek dari string descriptor untuk mendeskripsikan interface ini

Tabel 15. Data pada paket *Interface Descriptor*.

- **bInterfaceNumber**, mengindikasikan indek dari *interface descriptor*. Nilai ini dasarnya harus 0 dan akan naik untuk masing masing *interface descriptor.*
- **bAlternativeSetting** dapat digunakan untuk spesifik *alternatif interface*. *Alternatif interface* ini dapat di pilih dengan mengeset *interface request*.
- **bNumEndpoint** mengindikasikan jumlah *endpoint* yang digunakan untuk *interface*. Nilai ini harus tidak menggunakan *endpoint zero* dan digunakan untuk mengindikasikan jumlah *endpoint descriptor* yang mengikuti.
- **bInterfaceClass**, **bInterfaceSubClass** dan **InterfaceProtocol** dapat digunakan untuk menspesifikasikan class yang didukung, sebagai contoh HID, *communication*, *mass storage*, dan lain sebagainya.
- **iInterface** memperbolehkan untuk menyipan informasi *text* dari *interface*.

4. Endpoint Descriptors

Endpoint Descriptor digunakan untuk mendeskripsikan *endpoint* selain *endpoint zero. Endpoint zero* selalu diasumsikan untuk mengontrol *endpoint* dan dikonfigurasikan sebelum *endpoint* lain. *Host* akan menggunakan informasi ini untuk mengembalikan dari *descriptor* ini untuk memperkirakan persyaratan *bandwidth* yang digunakan.

Offset	Field	Ukuran	Nilai	Keterangan
0	bLength	1	Jumlah	Ukuran Descriptor dalam byte
1	bDescriptorType	1	Konstan	Endpoint Descriptor (0x05)
2	bEndpointAddress	1	Endpoint	Endpoint Address 0..3b jumlah endpoint 4..6b Dicadangkan (0) 7b Direction (diabaikan untuk control endpoint), yaitu: 0 = Out Endpoint 1 = In Endpoint

3	bmAttributes	1	Bitmap	Bit 0..1 Transfer Type 00=Control 01=isochronous 10=Bulk 11=Interrupt Bit 2..7 dicadangkan. Jika isochronous endpoint Bit 3..2 Tipe Singkronisasi (Iso mode) 00=Tidak singkron 01=Asingkron 10=Adaptive 11=Singkron Bit 5..4 Digunakan untuk Tipe (Iso mode) 00=Data endpoint 01=Feedback endpoint 10=Explicit Feedback data Endpoint 11=Dicadangkan
4	wMaxPacketSize	2	Jumlah	Maksimal ukuran paket endpoint untuk mengirim atau menerima
6	bInterval	1	Jumlah	Interval untuk polling endpoint data transfer. Nilai dalam jumlah frame. Diabaikan untuk Bulk dan Control Endpoint. Iso harus sama dengan 1 dan nilai field mungkin dalam jarak 1 sampai 255 untuk Interrupt endpoint.

Tabel 16. Data pada paket *Endpoint Descriptors*.

- **bEndpointAddress**, mengindikasikan endpoint apa yang mendeskripsikan
- **mbAttributes** berisi tipe transfer. Ini dapat juga sebagai Control, *Interrupt*, *Isochronous* atau *Bulk transfer*. Jika *Isochronous endpoint* di spesifikasikan, tambahan atribut dapat dipilih, sebagai contoh singkronisasi
- **wMaxPacketSize**, mengindikasikan maksimal ukuran *payload* untuk *endpoint* ini.
- **bInterval,** digunakan untuk nilai *polling* interval transfer tertentu. Satuannya di implementasikan dalam frame. 1 ms untuk *low* atau *full speed,* dan 125µs untuk perangkat *high-speed*

5. String Descriptors

String descriptor menghasilkan informasi yang dapat kita baca dan sifatnya opsional. Jika tidak digunakan, semua string indek harus di set menjadi 0 yang mengindikasikan bahwa tidak ada *string descriptor*.

String tersebut disandikan dengan format *unicode* sehingga dapat didukung banyak bahasa. String indek 0 harus mengembalikan daftar bahasa yang didukung. Daftar USB language ID dapat ditemukan di www.usb.org

Offset	Field	Ukuran	Nilai	Keterangan
0	bLength	1	Jumlah	Ukuran descriptor dalam byte
1	bDescriptionType	1	Konstan	String Descriptor (0x03)
2	wLANGID[0]	2	Jumlah	Mendukung kode bahasa 0 Contoh: 0x0409 English – United States
3	wLANGID[1]	2	Jumlah	Mendukung kode bahasa 1 Contoh: 0x0C09 English – Australia
4	wLANGID[2]	2	Jumlah	Mendukung kode bahasa X Contoh: 0x0407 German – Standar

Tabel 17. Data pada paket *String Descriptors*.

String Descriptor diatas memperlihatkan *String Descriptor 0*. *Host* harus membaca *descriptor* ini untuk memperkirakan bahasa apa yang tersedia. Jika bahasa mendukung, akan direferensikan dengan mengirimkan Language ID pada *wIndex field* dari permintaan *GetDescriptor*(string)

Setup Packet

Setiap perangkat USB harus merespon *Setup Packet* pada *standar transfer*. *Setup Packet* tersebut digunakan untuk mendeteksi dan mengkonfigurasi perangkat dan membawa fungsi umum, sebagai contoh seting alamat dari perangkat USB, meminta *device descriptor* atau mengecek *status endpoint*.

Semua perangkat USB, *host* mengharapkan semua perintah permintaan harus diproses dalam waktu kurang dari 5 detik.

- Permintaan perangkat standard tanpa proses data harus selesai dalam 50ms
- Permintaan perangkat standard dengan proses data harus dimulai untuk menghasilkan data 500ms setelah permintaan dijalankan.
- Perintah *SetAddress* harus memproses perintah dan mengembalikan status dalam 50ms, perangkat mempunyai 2ms untuk mengubah alamat sebelum permintaan berikut dikirim

Masing masing permintaan dimulai dengan Setup Packet dengan panjang 8 byte dengan format berikut.

Offset	Field	Ukuran	Nilai	Keterangan
0	bmRequestType	1	Bit-Map	**D7 Arah transfer Data** 0=Host ke perangkat 1=Perangkat ke Host **D6..5 Tipe** 0=Standard 1=Class 2=Vendor 3=Dicadangkan **D4..0 Penerima** 0=Perangkat 1=Interface 2=Endpoint 3=Lainnya 4..31 = Dicadangkan

1	bRequest	1	Nilai	Permintaan
2	wValue	2	Nilai	Nilai
4	wIndex	2	Indek atau offset	Indek
6	wLength	2	Jumlah	Jumlah byte yang di transfer jika itu adalah data phase

Tabel 18. Data pada paket *packet Setup*.

Field **bmRequestType** akan memperkirakan arah dari sebuah permintaan, tipe permintaan dan menentukan penerima data.
Field **bRequest** memperkirakan jenis permintaan yang sedang dibuat.

Standard Device Request

USB mempunya 8 buah standar permintaan, yaitu :

bmRequestType	bRequest	wValue	wIndex	wLength	Data
1000 0000b	GET_STATUS (0x00)	0	0	2	Status Perangkat
0000 0000b	CLEAR_FEATURE (0x01)	Fitur pemilih	0	0	-
0000 0000b	SET_FEATURE (0x03)	Fitur pemilih	0	0	-
0000 0000b	SET_ADDRESS (0x05)	Alamat Perangkat	0	0	-
1000 0000b	GET_DESCRIPTOR (0x06)	Descriptor Type&Index	0 atau Language ID	Panjang Descriptor	Descriptor
0000 0000b	SET_DESCRIPTOR (0x07)	Descriptor Type&Index	0 atau Language ID	Panjang Descriptor	Descriptor
1000 0000b	Get_CONFIGURATION (0x08)	0	0	1	Nilai konfigurasi

0000 0000b	SET_CONFIGURATION (0x09)	Nilai konfigurasi	0	0	-

Tabel 19. Daftar *standard Device Request*.

Standard Interface Request

bmRequestType	bRequest	wValue	wIndex	wLength	Data
1000 0001b	GET_STATUS (0x00)	0	Interface	2	Status Interface
0000 0001b	CLEAR_FEATURE (0x01)	Fitur pemilih	Interface	0	-
0000 0001b	SET_FEATURE (0x03)	Fitur pemilih	Interface	0	-
1000 0001b	GET_INTERFACE (0x0A)	0	Interface	0	Alternatife interface
0000 0001b	SET_INTERFACE (0x11)	Alternatif Seting	Interface	0	-

Tabel 20. Daftar *standard Interface Request*.

Standard Enpoint Request

bmRequestType	bRequest	wValue	wIndex	wLength	Data
1000 0010b	GET_STATUS (0x00)	0	Endpoint	2	Status Endpoint
0000 0010b	CLEAR_FEATURE (0x01)	Fitur pemilih	Endpoint	0	-
0000 0010b	SET_FEATURE (0x03)	Fitur pemilih	Endpoint	0	-
1000 0010b	SYNCH_FRAME (0x12)	0	Endpoint	2	Jumlah Frame

Tabel 21. Daftar *standard Endpoint Request*.

1.5 Cara Berkomunikasi Dengan Antar Muka Universal Serial Bus

Dalam model komunikasi yang akan dijelaskan berikut ini adalah dengan mengunakan standar transfer *setup-packet*. Dimana *setup-packet* digunakan untuk mengkonfigurasi dan membawa fungsi umum.

Tabel dibawah ini adalah struktur dari *setup-packet* dimana panjangnya 8 byte.

Offset	Field	Ukuran	Nilai	Keterangan
0	bmRequestType	1	Bit-Map	**D7 Arah transfer Data** 0=Host ke perangkat 1=Perangkat ke Host **D6..5 Tipe** 0=Standard 1=Class 2=Vendor 3=Dicadangkan **D4..0 Penerima** 0=Perangkat 1=Interface 2=Endpoint 3=Lainnya 4..31 = Dicadangkan
1	bRequest	1	Nilai	Permintaan
2	wValue	2	Nilai	Nilai
4	wIndex	2	Indek atau offset	Indek
6	wLength	2	Jumlah	Jumlah byte yang di transfer jika itu adalah data phase

Table 22. Data pada paket *packet Setup.*

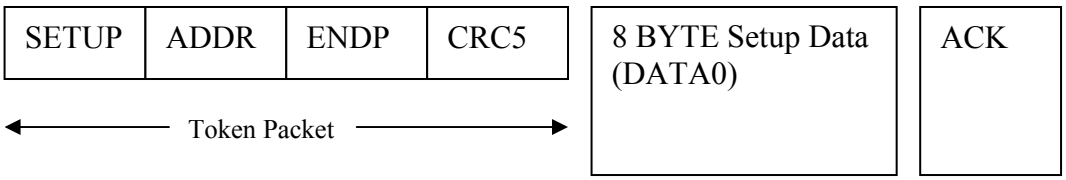
Dengan memberikan nilai 2 (10 biner) pada tipe *bmRequestType* maka pengiriman data setup-packet adalah dari *vendor* itu sendiri. Tabel berikut merupakan contoh pengiriman data ke perangkat USB.

Offset	Field	Ukuran	Nilai	Keterangan
0	bmRequestType 4..0: Recipient 6..5: Type 7: Direction	1 byte	C2h xxx00010 x10xxxxx 1xxxxxxx	 Endpoint Vendor Device-to-Host
1	bRequest	1 byte	05h	
2	wValue	2 byte	8080h	
4	wIndex	2 byte	0180h	
6	wLength	2 byte	0001h	

Raw data = 00h.

Table 23. Contoh pengiriman data menggunakan *setup-packet.*

Dikarenakan nilai *bmRequestType* dengan tipe *vendor*, maka nilai *bRequest*, *wValue, wIndex, wLength tergantung* oleh *vendor*. Raw data berisi nilai (data) yang dikirim dari perangkat ke *host* atau *host* ke perangkat.



Token Packet Setup = 1101b + komplemen = 11010010b

ADDR = xxxxxxxb

ENDP = 0001b

CRC5 = xxxxxb