2. TEORI PENUNJANG

Pada bab ini akan dijelaskan mengenai teori-teori yang menunjang proses pembuatan tugas akhir, antara lain: teori dasar mengenai USB, pengenalan *chip adapter* untuk *interface* USB PDIUSBD12 dan fitur-fitur yang dimiliki juga mengenai *mouse* dengan *interface* PS/2.

2.1. Universal Serial Bus (USB)

USB merupakan sebuah cara baru untuk menghubungkan peralatan komputer dengan komputer. Keistimewaan dari penggunaan teknik *interface* USB ini adalah semua jenis peralatan komputer dapat dihubungkan ke port USB sehingga memudahkan bagi orang awam untuk membeli dan menghubungkan peralatan komputer. Tidak seperti sekarang dimana ada bermacam-macam port pada *motherboard* komputer yang cukup menyulitkan bagi orang awam untuk menghubungkan peralatan komputer yang dimiliki.

2.1.1. Tujuan

USB diciptakan dengan tujuan membuat arsitektur komputer untuk standar industri yang berfokus pada *Computer Telephony Integration* (CTI), konsumen dan aplikasi yang produktif. Karena itu USB harus memenuhi kriteria-kriteria berikut ini:

- Memudahkan penambahan peralatan pada komputer
- Solusi hemat biaya dengan kecepatan tinggi
- Mendukung data *real time* untuk suara maupun gambar
- Protokol yang dapat mendukung pengiriman data secara *mixed-mode isochronous* dan *asynchronom messaging*
- Peralatan teknologi yang terintegrasi
- Pemahaman berbagai konfigurasi komputer
- *Interfuce* standar yang dapat digunakan dengan mudah untuk produk
- Memungkinkan penambahan alat jenis baru sesuai kemampuan komputer

2.1.2. Keunggulan USB

Spesifikasi USB memiliki keunggulan-keunggulan dibandingkan teknoiogi yang terdahuiu, antara lain:

- Bagi pengguna
  - Bentuk kabel dan *connector* hanya satu macam
  - Detil elektris terisolasi dari pengguna
  - Peralatan mampu mengenali diri sendiri dan pengaturan konfigurasi secara otomatis
  - Peralatanbisadipasangsecaradinamis.
- Mencakup area kerja yang iuas

  Sesuai untuk *bandwidth* alat mulai beberapa kb/s hingga beberapa Mb/s
  - Mendukung pengiriman secara *isochronous* maupun *asynchronous* dengan kabel yang sama
  - Mendukung penggunaan banyak alai sekaligus
  - Mendukung 127 buah alat
  - Mendukung pengiriman berbagai aliran data dan pesan antara *host* dan alat
  - Memungkinkan alat yang memiliki beberapa fungsi sekaligus
  - Beban protokoi lebih rendah sehingga menghasilkan penggunaan *bus* yang tinggi
- *Isochronous bandwidth:*

  Bandwidth yang terjamin sesuai untuk telepon, audio, dll.
  - Beban kerja *isochronous* bisa menggunakan *bus bandwidih* secara maksimal
- Fleksibilitas
  - Mendukung berbagai ukuran paket, yang memungkinkan berbagai pilihan ukuran *buffer*
  - Mendukung berbagai jenis kecepatan data dengan mengakomodasi ukuran *buffer* dan keterlambatan
  - Kendali aliran untuk penanganan *buffer* termasuk dalara protokol
- Ketangguhan:

  Mekanisme penanganan/pemuiihan kesalahan tertanam dalam protokol

- Penghubungan dan pelepasan alat secara dinamis teridentifikasi oleh pengguna secara *real-lime*
- Mendukung identifikasi dari aiat yang cacat

• Sinergi dengan industri PC:
- Protokol mudah diimplementasikan dan diintegrasikan
- Konsisten dengan arsitektur *plug-and-play* PC
- Mempengaruhi *interface* sistem operasi yang ada

• Implementasi murah:
- Jalur rendah biaya pada kecepatan 1.5 Mbps
- Teroptimasi untuk integrasi dalam perangkat keras alat maupun *host*
- Sesuai untuk pengembangan alat hemat biaya

• Pengembangan:
- Arsitektur dapat dikembangkan untuk mendukung penggunaan bebrapa USB *Host Controller* dalam satu sistem

2.1.3.KecepatanUSB

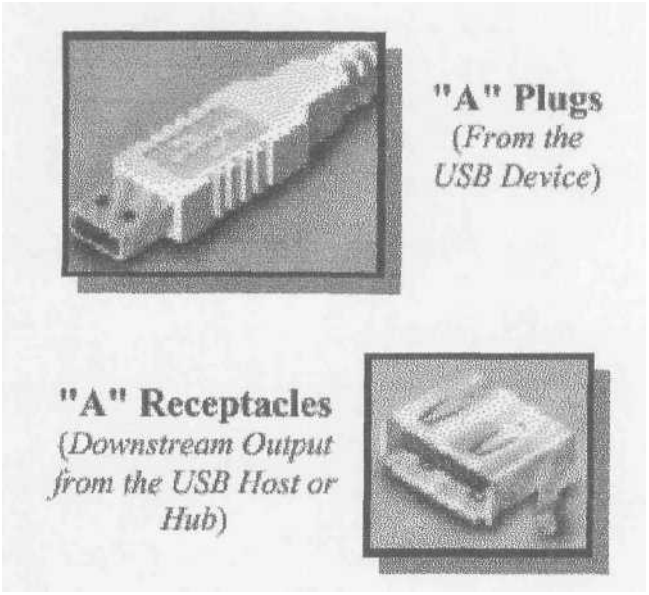
Peralatan USB sendiri terbagi atas 3 kelas berdasarkan kecepatan transfer data. Kelas tersebut yaitu *low speed*dengan kecepatan transfer l,5Mbits/s, *full speed* dengan kecepatan transfer 12Mbits/s (keduanya termasuk dalam spesifikasi USB versi 1.1) dan *high speed* yang kecepatan transfernya mencapai 480Mbits/s yang merupakan spesifikasi terbaru dari USB yaitu USB versi 2.0.

Setiap peralatan USB harus mengidentifikasi kecepatan transfer data yang digunakan apakah *low speed, full speed* atau *high speed.* Caranya dengan menghubungkan jalur D- dengan sebuah resistor*pull-up* ke tegangan 3,3V berarti merupakan peralatan USB *low speed,* bila yang dihubungkan dengan resistor *pull-up* ke tegangan 3,3 V adalah jalur D+ berarti merupakan peralatan USB/w// *speed.* Peralatan USB *high speed* pada inisialisasi awal akan dikenali sebagai peralatan *USBfull speed* hanya saja setelah tnisialisasi terjadi negosiasi kecepatan dengan USB *host,* setelah berhasii dikenaii sebagai peratatan USB *high speed* resistor *pull-up* tidak boleh dihubungkan lagi supaya jalur komunikasinya seimbang.

2.1.4. Konektor, Kabel Dan Koneksi USB

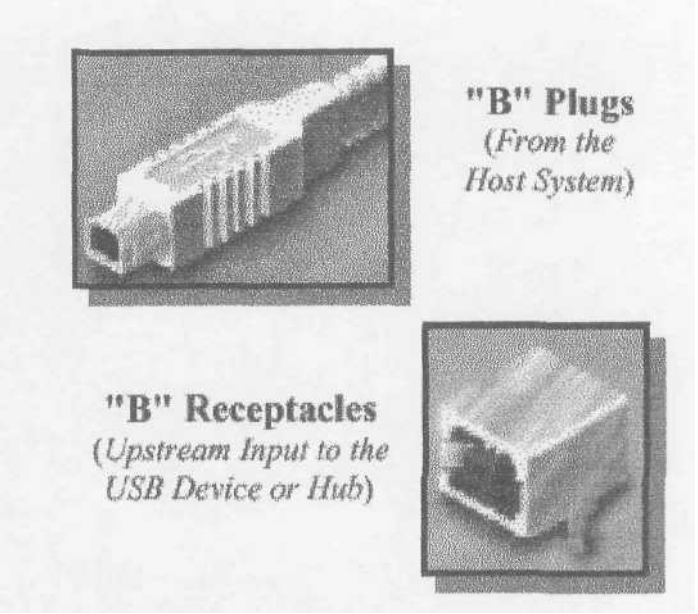
Karena USB ini merupakan sebuah standar baku maka kabel yang dipakai untuk menghubungkan komputer dan peralatannya juga harus memenuhi standar USB tertentu, akibatnya di pasaran hanya tersedia kabel USB yang sudah jadi yaitu kabel yang pada salah satu ujungnya terpasang konektor tipe A dan pada ujung yang lain terpasang konektor tipe B. Meskipun demikian terminal atau port USB yang dipasang pada PCB banyak dijual.

Konektor USB ada 2 macam yaitu konektor tipe A yang dan tipe B, dimana tipe A menghubungkan kabel USB ke port USB pada komputer dan tipe B menghubungkan kabel USB dengan peralatan USB. Hanya saja untuk peralatan sederhana misalnya *keyboard* dan *mouse* biasanya konektor tipe B tidak digunakan dan kabel langsung dihubungkan ke dalam peralatan untuk menghemat biaya dan dengan demikian ukuran peralatan jadi lebih kecil. Bentuk-bentuk konektor USB seperti yang ada pada gainbar 2.1. dan gambar 2.2.



Gambar 2.1. Konektor USB Tipe A

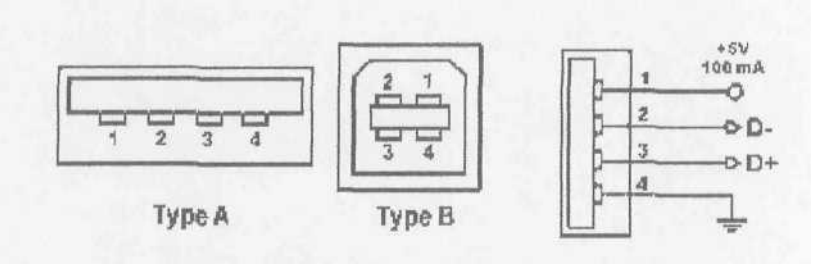
Sumber: USB *Specification* 1.1



Gambar 2.2. Konektor USB Tipe B

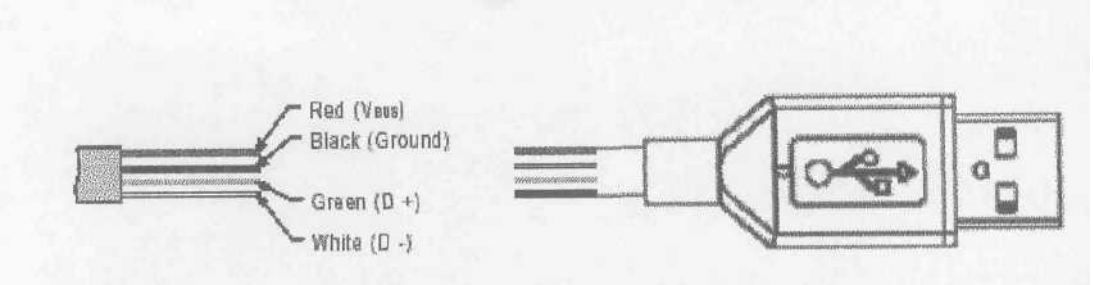
Sumber: USB *Specifwation* 1.1

Kabel USB terdiri atas 4 buah kabel didalaimya, terdiri atas kabel nomor 1 berwarna merah yang terhubung dengan Vbus (sumber tegangan 5 Volt), kabel nomor 2 berwaraa putih merupakan jalur D-, kabel nomor 3 berwama hijau merupakan jalur D+ dan kabel nomor 4 berwarna hitam yang menghubungkan ground. Hubungan antara kabel USB dengan port USB baik tipe A maupun tipe B sesuai dengan gatnbar 2.3. Sedangkan gambar penampang kabel USB ditunjukkan dalam gambar 2.4.



Gambar 2.3. Penampang Konektor USB

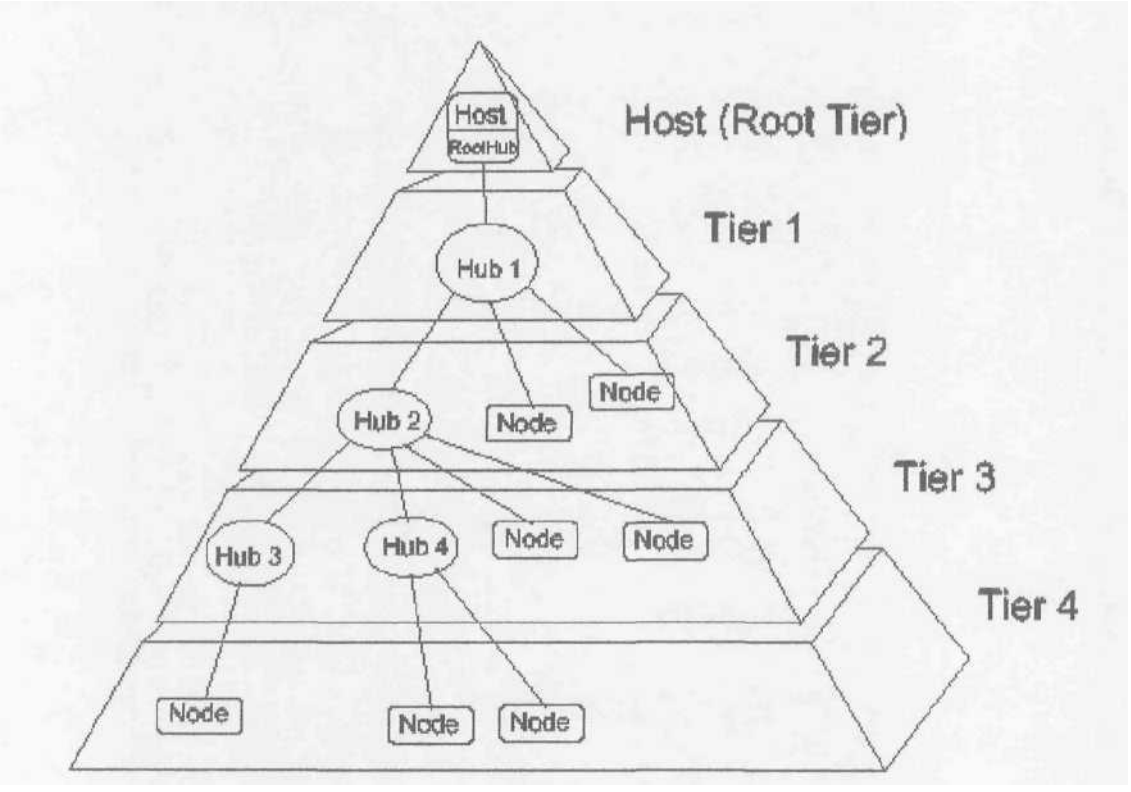
Sumber: STTS USB



Gambar 2.4. Penampang Kabel USB

Sumber: USB *Specification* 1.1

Koneksi USB menggunakan topologi *tiered star*. Sebuah *host* USB dapat dihubungkan dengan peralatan USB baik secara langsung maupun secara bertingkat melalui *hub* USB. Maksimal sebuah *host* USB dapat dihubungkan dengan 127 alat USB. Gambar 2.5. menunjukkan topologi USB.

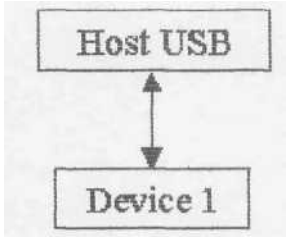


Gambar 2.5. Topologi USB

Sumber: USB *Specification* 1.1

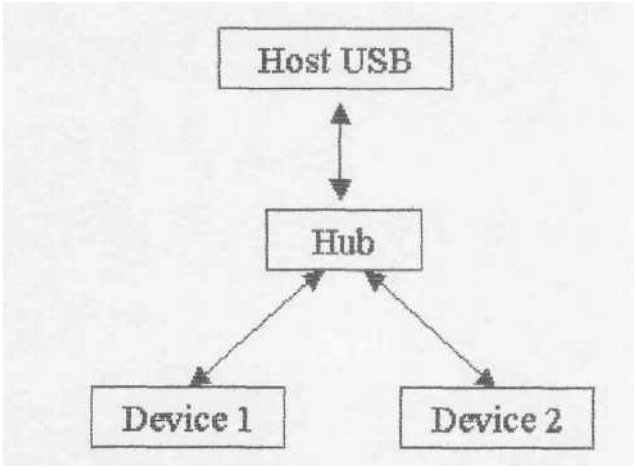
Komputer merupakan sebuah *host* USB. *Motherboard* komputer umumnya memiliki 2 port USB, dimana setiap port USB dapat dihubungkan dengan peralatan USB. Hanya saja untuk menghubungkan lebih dari 1 peralatan USB, port USB harus dihubungkan terlebih dahulu dengan *hub* USB (yang dapat dihubungkan dengan 4 peralatan USB sekaligus ataupun juga dengan USB *hub* yang lain). Tanpa menggunakan *hub* USB berarti hanya ada 2 peralatan USB yang

dapat terhubung dengan komputer secara bersamaan, Setiap port USB dapat menyediakan arus maksimal 500mA (untuk peralatan yang menggunakan *bm powered),* tetapi saat inisialisasi awal setiap peralatan USB hanya boleh menggunakan arus maksimal lOOmA. Biia membutuhkan arus lebih besar dari 500mA, peralatan USB tersebut haras--memiHki--sumber- •daya--sendiri *{self powered).* Hubungan antara *host* USB (komputer) dan peralatan USB tanpa menggunakan *hub* USB ditunjukkan pada gambar 2.6.



Gambar 2.6. Koneksi USB *Host* - USB *Device* Secara Langsung

Pada gambar 2.7. berikut ini menunjukkan cara koneksi peralatan USB dan *host* USB (komputer) melalui *hub* USB.



Gambar 2.7. Koneksi USB *Host*- USB *Device* Melaiui *Huh* USB

2.1.5. SinyaUISB

Sinyal USB merupakan sinyal digital yang dikirimkan melalui D- dan D+. Sinyal mi merupakan sinyal differensial dimana sinyal '0' dinyatakan dengan tegangan D+ < tegangan D- dan sinyal '1' dinyatakan dengan tegangan D+ > tegangan D-. Tidak dinyatakan dengan level tegangan saluran terhadap ground

seperti pada TTL atau RS232. Cara mengirimkan data T adalah dengan memberi D+ 2,8V dan D- tegangan dibawah 0,3V sedangkan cara mengirimkan data 'O' adalah dengan memberi D- 2,8V dan D+ tegangan dibawah 0,3V. Besarnya beda tegangan antara jaiur D- dan D+ atau sebaliknya yang diperiukan oleh penerima data adalah 200mV.

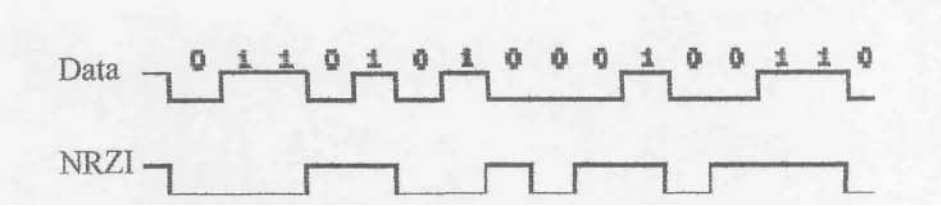
2.1.6. Transfer Pata

Transfer data antara *host* USB dan alat USB berlangsung melalui *endpoint. Endpoint* digunakan untuk mengirimkan atau menerima data dan dapat digunakan untuk komunikasi satu arah ataupun komunikasi dua arah sekaligus. Agar dapat mengirim atau menerima data melalui *endpoint,* kita harus mengetahui format protokol transfer. Pada USB versi 1.1 ada empat jenis transfer, masing-masing mempunyai karakteristik berbeda sesuai dengan kebutuhan. Keempat jenis transfer tersebut adalah:

- *Control Transfer.* digunakan untuk proses konfigurasi saat alat pertama kali dihubungkan dengan komputer.
- *hochronous Tramfer.* digunakan untuk informasi yang peka terhadap waktu. Komunikasi terjadi secara periodik dan kontinu.
- *Interrupt Transfer.* digunakan untuk mengirimk7an data yang berukuran kecii, dan berasai dari *interrupt.*
- *Bulk Transfer.* digunakan untuk data bemkuran besar, komunikasinya tidak periodik dan *seguential.*

2.1.7. Pengkodean Data

Seperti telah disebutkan, USB menggunakan 2 buah jalur untuk mengirimkan data yaitu: D+ dan D-. USB mengirim dan menerima data dengan menggunakan pengkodean NRZI *(Non Retnrrt to Zero Irtvert).* Data dikirimkan mulai dari *Least Significant* Bit (LSB) ke *Most Significant* Bit (MSB).



Gambar 2.8. Contoh Metode NRZI

Sumber: USB Specification 1.1 (telah diolah kembali)

Pengkodean ini akan berabah nilai logikanya bila ada data'O'. Selain menggunakan metode NRZI, USB juga menggunakan tekhik bit *siuffing.* Pada teknik ini, logika '0' disisipkan setelah ada 6 logika '1' secara berarutan. Hal ini bertujuan untuk memastikan cukup transisi sinyal.

Selain itu USB mempunyai *Cyc/ic Redundancy Check* (CRC) untuk memastikan tidak terjadi kesalahan data yang dikirimkan atau diterima. CRC ini ada 2 jenis yaitu CRC 5 bit dan CRC 16 bit. CRC 5 digunakan untuk *address* dan *endpoint* sedangkan CRC 16 untuk data.

2.1.8. Request

Request didefinisikan sebagai permintaan ataupun perintah yang dikirimkan oleh *host* USB kepada alat USB. Jika request tidak dikenali oleh alat, akan dikirim jawaban STALL kepada *host.* Request ini dikirimkan melalui *endpoint control* dengan transfer *control.* Format *request* terdapat pada tabel 2.1. berikut.

Tabel 2.1. Format *Reguest*

Offset	Nama	Ukuran (byte)	Penjelasan
0	BmRequestType	1	D7: Arah pengiriman data 0 = *host* ke alat 1 = alat ke *host* D6…5: Tipe 0 = Standar 1 = *Class* 2 = *Vendor* 3 = *Reserved* D4…0: Penerima 0 = Alat 1 = *Interface* 2 = *Endpoint* 3 = Lainnya 4…31 = *Reserved*
1	Brequest	1	Request
2	Wvalue	2	Nilai bervariasi tergantung *request*

Tabel 2.1. Format *Reguest* (sambungan)

4	Windex	2	Nilai bervariasi tergantung *request*
6	Wlength	2	Jumlah data bila diikuti tahap data

Bagian bmRequestType menunjukkan arah transfer data yang akan berlangsung untuk tahap berikutnya, arahnya ini diabaikan biia panjang data (terdapat pada bagian wlength) yang dikirimkan adalah 0 (menunjukkan tidak ada data yang akan dikirimkan). Untuk *request input,* wLength menunjukkan jumlah data maksimal yang bisa dikerabalikan oleh alat. Sedangkan untuk *request output,* wLength menunjukkan jumlah data yang akan dikirimkan *host.* Data dapat dikirimkan ke *intetface* atau *endpoint* pada alat USB, bagian wlndex menunjukkan nomor *irtterface* atau *endpoint* yang akan dikirimi data. Bagian bRequest menunjukkan nama atau jenis *request* yang diminta.

Tabel 2.2. Kode bRequest Standar

Brequest	**Nilai**
GET STATUS	00H
CLEAR FEATURE	01H
Reserved	02H
SET FEATURE	03H
Reserved	04H
SET ADDRESS	05H
GET DESCRIPTOR	06H
SET DESCRIPTOR	07H
GET CONFIGURATION	08H
SET CONFIGURATION	09H
GET INTERFACE	0AH
SET INTERFACE	0BH
SYNCH FRAME	0CH

- *Device Reguest*

Ada sebelas *device reguest* seperti yang tercantum dalam tabel berikut ini.

Tabel 2.3. *Device Reguest*

BmRequestType	Brequest	Wvalue	WIndex	WLength	Data
00000000B	01H	Pemilih Fitur	0	0	-
10000000B	08H	0	0	1	Nilai Konfigurasi
10000000B	06H	Tipe *Descriptor*	0 atau ID	Panjang	*Descriptor*

Tabel 2.3. *Device Reguesl* (sambungan)

BmRequestType	Brequest	Wvalue	Windex	Wlength	Data
			bahasa	Descriptor	
10000001B	0AH	0	*Interface*	1	*Interface* lain
10000000B	00H	0	0	2	Status Alat
00000000B	05H	Alamat Alat	0	0	-
00000000B	09H	Nilai Konfigurasi	0	0	-
00000000B	07H	Tipe *Descriptor*	0 atau ID Bahasa	Panjang *Descriptor*	*Descriptor*
00000000B	03H	Pemilih Fitur	0	0	-
00000001B	0BH	*Setting* lain	*Interface*	0	-
10000010B	0CH	*0*	*Endpoint*	2	Nomor *frame*

Dari *device request* tersebut diatas, software Tugas Akhir ini hanya mengenali *request. Get Status* (00H), *Set Address* (05H), *Get Descripior* (06H), *Get Conftguration,* (08H) dan *Set Configuration* (09H). Sedangkan *request Clear Feature* (OIH), *Set Feature* (03H), *Set Descriptor* (07H), *Get Interface(OAH), Set Interface* (0BH) dan *Synch Frame* (0CH) tidak dikenali.

Tabel 2.4. menunjukkan nilai dan tipe *descriptor* apa yang diminta melalui *request (get descriptor* dan *set descriptor).*

Tabel 2.4. Nilai Dan Tipe *Descriptor*

Tipe *Descriptor*	Nilai
Device	1
Configuration	2
String	3
Interface	4
Endpoint	5

Berikut ini penjelasan masing-masing *requesf.*

- *Clear Feature* (OIH) dan *Set Feature* (03H) digunakan untuk mematikan atau mengaktifkan fitur *Device Remote Wakeup.*
- *Get Configuration* (08H) digunakan untuk meminta nilai konfigurasi alat. Jika alat mengirimkan nilai no! berarti belum ada konfigurasi. sedangkan nilai selain nol menunjukkan aiat dalam kondisi terkonfigurasi.

  Get Descriptor (06H) digunakan untuk meminta *descriptor* sesuai dengan tipe *descriptor* yang tertera. Permintaan akan *device descriptor* akan dibalas alat dengan mengirimkan *device descriptor.* Sedangkan permintaan akan *configuration descriptor* akan dibalas alat dengan mengirimkan *configuration*

descriptor, *inlerface descriplor, class-specific descriplor* (jika ada), dan *endpoint descripior.* Hal ini dilakukan karena *interface* dan *endpoinl descriptor* tidak bisa diminta langsung dengan *Get Descriptor. Descriptor* yang dikirimkan alat tidak boleh melebihi panjang wLength yang telah ditentukan *host.*

- *Get Interface* (OAH) digunakan untuk meminta *setting* lain yang dipakai *interface.* Bila wValue tidak berisi 0 atau wLength tidak berisi 1 maka alat tidak dikenaii dan alat akan menanggapi dengan *request error.*
- *Get Status* (00H) digunakan untuk meminta status alat. Alat akan mengirimkan status sepanjang dua byte yang hanya digunakan 2 bit pertama (bit lain *reserved).* Nilai T pada Dl menunjukkan kemampuan alat dalam *remote wakeup* yaitu untuk membangunkan *host* dari kondisi *suspend.* Bit *remote wakeup* ini bisa diubah oleh *Set Featitre* atau *Clear Feature* yang memilih fitur *Device Remote Wakevp.* Nilai ' 1' pada DO menunjukkan bahwa alat mempunyai daya sendiri, jika bernilai '0' berarti alat bersifat *bus-powered.*
- *Sei Address* (05H) digunakan untuk memberi alat sebuah alamat baru.
- *Set Configuralion* (09H) digunakan untuk menentukan konfigurasi yang akan digunakan aiat.
- *Set Descriptor* (07H) digunakan untuk memperbaharui *descriptor* yang ada atau menambah *descriptor* baru.
- *Set Interface* (OBH) digunakan untuk memilih *setting* lain dari *interface* yang digunakan.
- *Synch Frame* (OCH) digunakan untuk melakukan *synchronisation frame* pada proses transfer *isochronous.*

• *Class-Specific Request*

Request ini ditujukan kepada *interface* (infbrmasi *class-speciflc* terdapat pada level *interface).* Ada enam *class-specific request* seperti yang tercantum dalam tabel-tabel berikut ini.

Tabel 2.5. Kode bRequest HID

BRequest	Nilai
GET REPORT	01H
GET IDLE	02H
GET PROTOCOL	03H
Reserved	04H – 08H
SET REPORT	09H
SET IDLE	0AH
SET PROTOCOL	0BH

Tabel 2.6. *Class-Specific Request*

BmRequestType	Brequest	WValue	Windex	WLength	Data
10100001B	01H	Tipe *Report* dan *Report* ID	Nomor *Interface*	Panjang *Report*	*Report*
10100001B	02H	0 dan *Report* ID	Nomor *Interface*	1	*Idle Rate*
10100001B	03H	0	Nomor *Interface*	1	0 = *Boot* 1 = *Report*
00100001B	09H	Tipe *Report* dan *Report* ID	Nomor *Interface*	Panjang *Report*	*Report*
00100001B	0AH	Durasi dan *Report* ID	Nomor *Interface*	0	-
00100001B	0BH	0 = *Boot* 1 = *Report*	Nomor *Interface*	0	-

Tabel 2.7. Tipe *Report*

Tipe *Report*	Nilai
Input	01H
Output	02H
Feature	03H
Reserved	04H - FFH

Dari enam *class-specific request* tersebut, *soflware* Tugas Akhir ini hanya mengenali *Get Report* (OIH) dan *Set Report* (09H). *Get Idle* (02H), *Get Protocol* (03H), *Set Jdle* (0AH), *dan Set Protocol* (0BH) tidak akan dikenaii. *Report* TD umunya jarang digunakan sehingga bernilai 'O'. Berikut ini penjelasan masing-masing *request:*

Get Report (OIH) digunakan untuk meminta laporan yang dijabarkan pada *report descriptor,* misalnya laporan mengenai status penekanan tombol *keyboard. Get Report* bersifat wajib bagi semua alat.

- *Get Idle* (02H) digunakan untuk meminta *idle rate* untuk *inpnt report* tertentu.

- *Get Proiocol* (03H) digunakan untuk meminta status protokol yang sedang digunakan/aktif. *Gei Proiocol* bersifat wajib bagi alat *bool* (berada dalam subkelas *hoot),*
- *Set Report* (09H) digunakan untuk memberi laporan yang dijabarkan pada *report descriptor,* misalnya laporan mengenai status LED *keyboard.* Jika alat tidak memiliki jalur *interrupt out endpoint,* maka perintah mi akan dikirimkan melalui *control endpoint.*
- *Set Idle* (OAH) digunakan untuk menghentikan *report* tertentu pada jalur *interrupt irt* sampai ada kejadian baru atau selama waktu tertentu berlalu. Hal ini akan membatasi frekuensi laporan pada *interrupt in endpoint.* Selama isi laporan tidak berubah, maka *endpoint* akan mengirimkan NAK setiap ada *polling.* Jika tidak ada perabahan, *polling* akan selalu menerima NAK selama durasi tertentu. Jika durasi diatur lebih kecil daripada *polling rate* maka laporan tetap dikirimkan sesuai *polling rate.* Misalkan durasi diatur selama 10 milidetik dan *polling rate* diatur pada 4 milidetik. Jika tidak ada perubahan pada laporan, maka polling rate akan mendapat NAK sebanyak dua kali (pada detik keempat dan kedelapan) sebelum alat mengeluarkan laporan (pada detik kesepuluh).
- *Set Protocol* (OBH) digunakan untuk menentukan status protokol yang akan digunakan/aktif antara protokoi *hoot* dan *report. Set Protocol* bersifat wajib bagi alat *boot* (berada dalam subkelas *boot).*

2.1.9. Descriptor

Descriptor adalah jawaban untuk *request* yang dikirimkan *host* yang berisi identitas alat yang termasuk nama dan konfigurasi alat. Secara umum terdapat *device descriptor, conftguration descriptor, interface descriptor, endpoint descriptor,* dan *string descriptor.* Semua *descriptor* diawali dengan satu byte yang menunjukkan panjang total dari *descriptor* tersebut dan diikuti tipe dari *descriptor. String descripior* berisi identitas bahasa yang digunakan dan data yang bisa ditampilkan serta dibaca secara langsung. *String descriptor* ini bersifat tidak mutlak untuk digunakan oleh sebab itu *software* pada Tugas Akhir ini tidak menggunakan *descriptor* ini. Tugas Akhir ini berhubungan dengan *mouse* (yang

termasuk kelas *Human Interface Device* (HID)) maka ada tambahan HID *descriptor* setelah *interface descriptor.* Jadi *descriptor* yang digunakan adalah; *device - configuration - interface* - HID - *endpoint.* Berikut ini penjelasan masing-masing *descriptor* dan nilai yang berhubungan dengan Tugas Akhir ini.

- *Device Descriptor*

Descriptor ini menjelaskan tentang *device* secara umum. Setiap *device* hanya memiliki satu *device descriptor.*

Tabel 2.8. *Device Descriptor*

Offset	Nama	Ukuran (byte)	Penjelasan	Contoh (*Hex*)
0	Blength	1	Besar *descriptor* ini dalam byte	12
1	BdescriptorType	1	Tipe *device descriptor*	01
2	BcdUSB	2	Versi spesifikasi USB dalam BCD	0110
4	BDeviceClass	1	Bernilai antara 01 – FEH. Jika bernilai FFH berarti *vendor-specific*. Jika bernilai 00 berarti informasi kelas berada pada tingkat *interface*.	00
5	BdeviceSubClass	1	Bernilai 00 jika bDeviceClass bernilai 00	00
6	BdeviceProtocol	1	Berisi protokol yang digunakan pada basis *device*. Bernilai 00 jika tidak ada.	00
7	bMaxPacketSize0	1	Besar paket maksimum pada *endpoint* 0	10
8	IdVendor	2	*Vendor* ID	0458
10	IdProduct	2	*Product* ID	0001
12	BcdDevice	2	Versi *device* dalam BCD	0100
14	Imanufacturer	1	Indeks *manufacturer* pada *string descriptor*. Bernilai 00 jika tidak ada.	00
15	Iproduct	1	Indeks produk pada *string descriptor*. Bernilai 00 jika tidak ada.	00
16	IserialNumber	1	Indeks nomor serial pada *string descriptor*. Bernilai 00 jika tidak ada.	00
17	BnumConfigurations	1	Jumlah konfigurasi yang mungkin	01

- *Configuration Descriptor*

Configuration descriptor ini menjelaskan tentang konfigurasi dari alat. Satu *device* dapat memiliki beberapa *configuration descriptor* naraun pada umumnya hanya memiliki satu *configuration descriptor.*

Tabel 2.9. *Configuration Descriptor*

Offset	Nama	Ukuran (byte)	Penjelasan	Contoh (*Hex*)
0	Blength	1	Besar *descriptor* ini dalam byte	09
1	BdescriptorType	1	Tipe *configuration descriptor*	02
2	WtotalLength	2	Jumlah data yang akan dikirimkan (*configuration*, *interface*, HID, dan *endpoint descriptor*)	0022
4	BnumInterface	1	Jumlah *interface* yang dikenali	01
5	BconfigurationValue	1	Nilai yang dipakai *host* untuk memilih konfigurasi ini	01
6	Iconfiguration	1	Indeks konfigurasi pada *string descriptor*. Bernilai 00 jika tidak ada.	00
7	BmAttributes	1	D7 *Bus Powered* D6 *Self Powered* D5 *Remote Wakeup* D4..0 harus bernilai 0	80
8	BmaxPower	1	Konsumsi arus maksimum dalam satuan 2 mA (1EH = 30 = 30 x 2 mA = 60mA)	1E

D7 bmAttributes pada USB 1.0 memang digunakan untuk menunjukkan *bus power* tetapi pada USB 1.1 harus bernilai 1 sedangkan indikator *bus power* ditunjukkan dari nilai bMaxPower selain 0.

- *Interface Descriptor*

Descriptor ini dikirimkan setelah *configuration descnptor.* Setiap konfigurasi dapat memiliki lebih dari satu *interface* sehingga *inlerface descriptor* tersebut harus dikirimkan sesuai dengan *configuralion descripior.*

Tabel 2.10. *Interface Descriptor*

Offset	Nama	Ukuran (byte)	Penjelasan	Contoh (*Hex*)
0	Blength	1	Besar *descriptor* ini dalam byte	09
1	BdescriptorType	1	Tipe *interface descriptor*	04
2	BinterfaceNumber	1	Nomor *interface*. Bersifat *zero-based*.	00

Tabel 2.10. *Inferfdce Descripior* (sambungan)

Offset	Nama	Ukuran (byte)	Penjelasan	Contoh *(Hex)*
3	BaltemateSetting	1	Nilai untuk memilih *setting interface* lain	00
4	BnumEndpoints	|	Jumlah *endpoinl* yang digunakan. Jika 0 berarti hanya menggunakan *endpoini* 0.	01
5	BinterfaceClass	1	Tidak boleh bernilai 0. Jika bernilai FF berarti *vendor-specific,* Kelas HID bernilai 3.	03
6	BinterfaceSubClass	1	Bernilai 0 jika blnterfaceClass bernilai 0. PadaHID: 0 tidak ada *subclass* 1 *boot interface subclass.*	00
7	BintertaceProtocol	1	Jika bernilai 0 berarti tidak ada protokol yang bersifat *class-specific.* Jika bernilai FF berarti menggunakan protokol yang *vendor-specific.* Pada HID: 0 tidak ada protokol 1 *keyboard* *2 moiise.* Jika blnterfaceSubClass bernilai 0 maka blnterfaceProtocol harus bemilai 0.	00
8	linterfkce	1	Indeks *interface* pada *string descriptor.* Bernilai 00 jika tidak ada	00

- HID *Descriptor*

Descriptor ini merupakan *descriptor class-specific* dan dikirimkan setelah *interface descriptor. VHD class descriptor* terdiri dari dua *descriptor, report descriptor danphysical descriptor. Report descriptor* berisi format data yang akan dikirimkan. Sedangkan *physical descriptor* menyatakan bagian tubuh manusia yang digunakan untuk menjalankan sebuah alat. *Physical descriptor* bersifat tidak mutlak dan tidak digunakan pada Tugas Akhir ini.

Tabel 2.11. HID *Descriptor*

Offset	Nama	Ukuran (byte)	Penjelasan	Contoh *(Hex)*
0	Blength	1	Besar *descriptor* ini dalam byte	09
1	BdescriptorType	1	Tipe HID *descnptor*	21
2	BcdfflD		Versi spesifikasi HID daiam	0111

Tabel 2.11. HID *Descriptor* (sambungan)

Offset	Nama	Ukuran (byte)	Penjelasan	Contoh (*Hex*)
			BCD	
4	BcountryCode	1	Negara target perangkat keras. Bernilai 0 jika alat tidak terlokalisasi	00
5	BnumDescriptor	1	Jumlah HID *class descriptor* yang mengikuti	01
6	BdescriptorType	1	Tipe *report descriptor*	22
7	WdescriptorLength	2	Besar *report descriptor*	0032

Jika ada dua HDD *class descriptor* yang dikirim maka bDescriptorType dan wDescriptorLength untuk HID *class descriptor* kedua harus disertakan di beiakang bDescriptorType dan wDescriptorLength yang pertama.

- *Endpoint Descriptor*

Descriptor ini dikirimkan setelah HE) *descriptor. Endpoint* 0 tidak boleh disertakan. Jika tidak menggunakan *endpoint* lain selain *endpoint* 0 maka tidak ada *endpoint descriptor.*

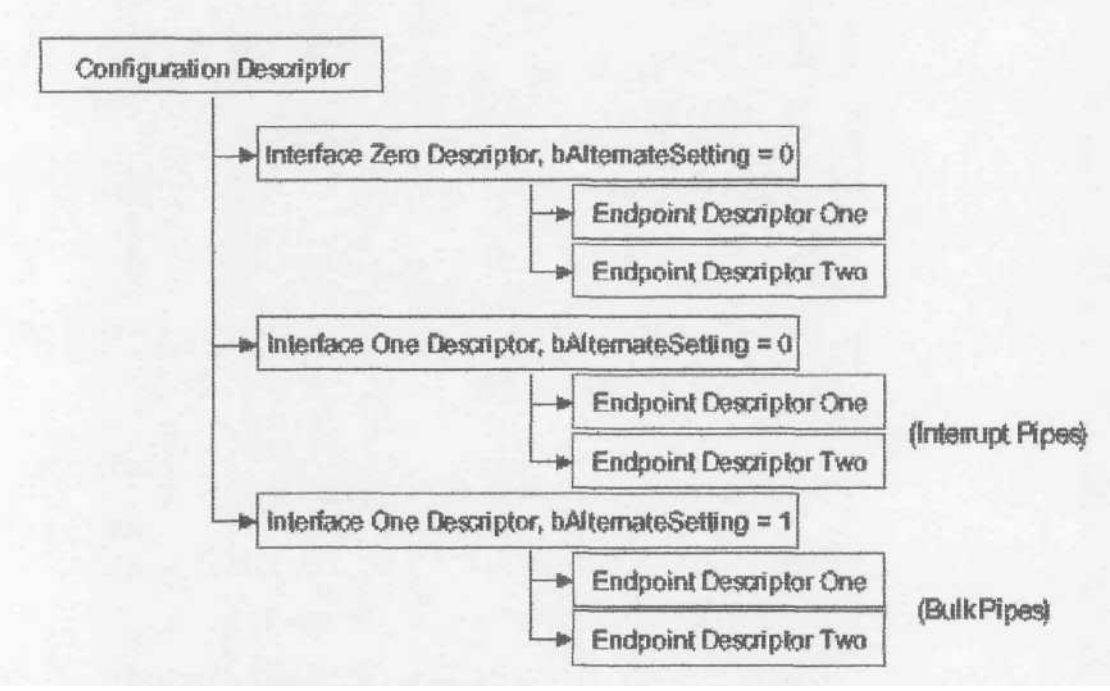
Tabel 2.12. *Endpoint Descriptor*

Offset	Nama	Ukuran (byte)	Penjelasan	Contoh (*Hex*)
0	Blength	1	Besar *descriptor* ini dalam byte	07
1	BdescriptorType	1	Tipe *endpoint descriptor*	05
2	BendpointAddress	1	D3..0 nomor *endpoint* D6..4 harus bernilai 0 D7 arah (diabaikan untuk *control endpoint*), 0 = OUT *endpoint*, 1 = IN *endpoint*	81
3	BmAttributes	1	D1..0 tipe transfer: 00 *Control* 01 *Isochronous* 10 *Bulk* 11 *Interrupt* Bit lain harus bernilai 0.	03
4	WmaxPacketSize	2	Besar paket maksimum yang bisa ditransaksikan melalui *endpoint* ini. Untuk *interrupt endpoint*, jumlah data yang lebih kecil dari ukuran maksimum bisa dikirimkan tetapi akan menghentikan transfer dan membutuhkan intervensi untuk memulai kembali	0008

Tabel 2.12. *Endpoini Descriplor* (sambungan)

Ojfset	Nama	Ukuran (byte)	Penjelasan	Contoh *(Hex)*
6	Binterval	1	Interval untuk *polling* data dalam satuan milidetik. Diabaikan untuk *bulk* dan *control endpoint.* Untuk *isochronous endpoint* harus berniiai 1. Untuk *interrupt endpnint* bisa bernilai antara 1 hingga FF.	10

Interface, HID, dan *endpoint descriptor* tidak pemah diminta secara terpisah maka harus dikirimkan mengikuti *configuration descriptor.* Gambar berikut menunjukkan kemungkinan format pengiriman jika terdapat lebih dari satu konfigurasi, *interface,* dan *cndpoint.*



Gambar 2.9. Diagram *Configuration Descriptor* Banyak Konfigurasi

- *Report Descriptor*

Descriptor ini berisi format laporan yang akan dikirimkan. Untuk *rnouse, descriptor* ini meliputi: jumlah dan besar laporan, nilai maksimum dan minimum (ditekan dan dilepas) dan tombol yang digunakan. Berikut ini *repori descriplor* pada Tugas Akhir ini berikut penjelasannya.

Tabel 2.13. *Report Descriptor* Untuk *Mouse*

Nama	*Hex*	Biner	Penjelasan
Usage Page (Generic Desktop)	05 01	000001 01 0000 0001	000001B menunjukkan kode *Usage Page*, 01B menunjukkan ukuran, dan 0000 0001B menunjukkan kode *Generic Desktop*
Usage (*Mouse*)	09 02	000010 01 0000 0010	000010B menunjukkan kode *Usage*, 01B menunjukkan ukuran, dan 0000 0010B menunjukkan kode *mouse*
Collection (Application)	A1 01	101000 01 0000 0001	101000B menunjukkan kode *Collection*, 01B menunjukkan ukuran, dan 0000 0001B menunjukkan aplikasi
Usage (*Pointer*)	09 01	000010 01 0000 0001	000010B menunjukkan kode *Usage*, 01B menunjukkan ukuran, dan 0000 0001 menunjukkan kode *Pointer*
Collection (Application)	A1 00	101000 01 0000 0000	101000B menunjukkan kode *Collection*, 01B menunjukkan ukuran, dan 0000 0000B menunjukkan aplikasi
Usage Page (Buttons)	05 09	000001 01 0000 1001	000001B menunjukkan kode *Usage Page*, 01B menunjukkan ukuran, dan 0000 1001B menunjukkan kode *button*
Usage Minimum (01)	19 01	000110 01 0000 0001	000110B menunjukkan kode *Usage Minimum*, 01B menunjukkan ukuran, 0000 0000B menunjukkan penggunaan terkecil
Usage Maximum (03)	29 02	001010 01 0000 0010	001010B menunjukkan kode *Usage Maximum*, 01B menunjukkan ukuran, 0000 0010B menunjukkan penggunaan terbesar
Logical Minimum (0)	15 00	000101 01 0000 0000	000101B menunjukkan kode *Logical Minimum*, 01B menunjukkan ukuran, dan 0000 0000B menunjukkan nilai logika terendah yang akan dilaporkan
Logical Maximum (1)	25 01	001001 01 0000 0001	001001B menunjukkan kode *Logical Maximum*, 01B menunjukkan ukuran, dan 0000 0001B menunjukkan nilai logika terbesar yang akan dilaporkan
Report Count (3)	95 03	100101 01 0000 0011	100101B menunjukkan kode *Report Count*, 01B menunjukkan ukuran, 0000 0011B menunjukkan bahwa ada 3 buah laporan masing-masing berukuran 1 bit (*report size*)
Report Size (1)	75 01	011101 01 0000 0001	011101B menunjukkan kode *Report Size*, 01B menunjukkan ukuran, dan 0000 0001B menunjukkan bahwa ukuran 1 laporan = 1 bit

Tabel 2.15. *Report Descriptor* untuk *Mouse* (sambungan)

Nama	*Hex*	Biner	Penjelasan
Input (Data, Var, Abs)	81 02	100000 01 00000 010	100000B menunjukkan kode *Input* (data dari alat), 01B menunjukkan ukuran, tiga bit terakhir (010B) menunjukkan sifat absolut/0 (nilai pasti, bukan relatif terhadap nilai sebelumnya), variabel/1, dan data/0 (bisa berubah, bukan konstanta tetap)
Report Count (1)	95 01	100101 01 0000 0001	100101B menunjukkan kode *Report Count*, 01B menunjukkan ukuran, 0000 0001B menunjukkan bahwa ada 1 buah laporan yang berukuran 5 bit (*report size*)
Report Size (5)	75 05	011101 01 0000 0101	011101B menunjukkan kode *Report Size*, 01B menunjukkan ukuran, dan 0000 0101B menunjukkan bahwa ukuran 1 laporan = 5 bit
Input (Constant)	81 03	100000 01 0000 0011	100000B menunjukkan kode *Input* (data dari alat), 01B menunjukkan ukuran, tiga bit terakhir (001B) menunjukkan sifat absolut/0, variabel/0, dan konstanta/1 (tidak bisa berubah)
Usage Page (Generic Desktop)	05 01	000001 01 0000 0001	000001B menunjukkan kode *Usage Page*, 01B menunjukkan ukuran, dan 0000 0001 menunjukkan kode *Generic Desktop*
Usage (X)	09 30	100100 01 0011 0000	100100B menunjukkan kode *Usage*, 01B menunjukkan ukuran, 0011 0000B menunjukkan penggunaan terkecil
Usage (Y)	09 31	100100 01 0011 0001	100100B menunjukkan kode *Usage*, 01B menunjukkan ukuran, 0011 0001B menunjukkan penggunaan terbesar
Logical Minimum (-127)	15 81	000101 01 1000 0001	000101B menunjukkan kode *Logical Minimum*, 01B menunjukkan ukuran, dan 1000 0001B menunjukkan logika terendah yang akan dilaporkan
Logical Maximum (127)	25 7F	001001 01 0111 1111	001001B menunjukkan kode *Logical Maximum*, 01B menunjukkan ukuran, dan 0111 1111B menunjukkan logika tertinggi yang akan dilaporkan
Report Size (8)	75 08	011101 01 0000 1000	011101B menunjukkan kode *Report Size*, 01B menunjukkan ukuran, dan 0000 1000B menunjukkan bahwa ukuran 1 laporan = 8 bit
Report Count (2)	95 02	100101 01 0000 0010	100101B menunjukkan kode *Report Count*, 01B menunjukkan ukuran, 0000 0010B menunjukkan bahwa ada 2 buah laporan yang berukuran 8 bit (*report size*)

Tabel 2.13. *Report Descriptor* untuk *Mouse* (sambungan)

Nama	*Hex*	Biner	Penjelasan
Input (Data, Var, Relative)	81 06	100000 01 0000 0110	100000B menunjukkan kode *Input*, 01B menunjukkan ukuran, tiga bit terakhir (110B) menunjukkan sifat data/0 (data dapat berubah bukan konstanta), variabel/1(berarti variabel bukan array) , dan relatif/1 (nilai relatif terhadap nilai sebelumnya)
End Collection	C0	110000 00	110000B menunjukkan kode untuk mengakhiri *Collection* dan 00B menunjukkan ukuran
End Collection	C0	110000 00	110000B menunjukkan kode untuk mengakhiri *Collection* dan 00B menunjukkan ukuran

2.1.10. Proses Enumerasi

Definisi proses enumerasi *(bm enumeration)* menurut USB *Specijication* 1.1 adalah kegiatan pengidentifikasian dan pemberian alamat tertentu pada alat yang dihubungkan ke *bus.* Pada protokol USB, sebuah alat yang dihubungkan ke komputer akan menerima serangkaian *request,* dari *deacriptor* yang dikirimkan sebagai balasan *reguest* kepada *host* dapat diketahui identitas dan kemampuan alat tersebut. Proses enumerasi secara umum adalah sebagai berikut:

- Alat dihubungkan ke komputer, USB *host controller* memulai enumerasi dengan melakukan *bus reset* dan memberikan alamat *default* 0 pada alat.
- *Host* meminta *device descriptor* panjang data yang diperlukan *host* hanya delapan byte pertama yang berisi kapasitas *endpoint* 0 dan panjang *device descriptor* sesungguhnya.
- *Host* melakukan *hus reset* lagi.
- *Host* memberikan perintah *set address* yang bertujuan memindahkan jalur komunikasi dengan alat ke alamat baru.
- *Host* meminta *device descriptor* iengkap.
- Setelah itu *host* minta *configuration descriptor. Request* ini dilakukan dalam dua tahap, yang pertama *host* hanya meminta sembilan byte pertama yang berisi jumlah total *configuratwn descriptor.* Dan pada kedua *host* meminta keseluruhan *configuration descriptor.*
- Sesudah itu *host* akan meminta *string descriptor* jika ada.

- Lalu sistem operasi komputer akan meminta *driver* dan melakukan instalasi alat tersebut.
- Komputer meminta semua *descriptor* lagj setelah itu melakukan *Sel Conflguration* dan menentukan konfigurasi alat.

Jika pada proses enumerasi ini aiat mengirimkan balasan yang salah, *host* akan mengulangi proses enumerasi dari awal sebanyak dua kali lagi (proses enumerasi maksimal tiga kali) sebelum menghentikan proses enumerasi dan alat *di-suspend.*

2.2. PD1USBD12

PDIUSBD12 ini merupakan sebuah *chip* USB *interface* buatan Philips Semiconductors. *Chip* ini dapat berkomunikasi dengan mikrokontroler dengan kecepatan tinggi melalui hubungan paralel. *Chip* ini sesuai dengan spesifikasi USB 2.0 dan dapat digunakan untuk sebagian besar jenis peralatan yang ada misalnya peralatan *printing,* peralatan komunikasi, *Human Interf'ace Device,* peralatan penyimpan data dan peralatan yang termasuk dalam *Imaging Class.*

2.2.1. Fitur Dari PDIUSBD12

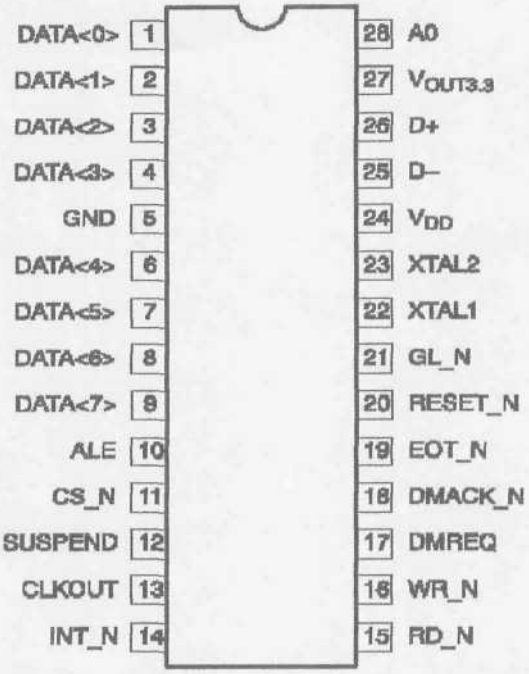
Chip ini memiliki berbagai jenis fitur, diantaranya:

- Memenuhi spesifikasi USB 2.0
- Mempunyai *Serial Inlerface Engine* (SIE), memori *Firsl In First Out* (FTFO), *transceiver* dan juga pengatur tegangan yang terintegrasi.
- Dapat digunakan untuk hampir semua spesifikasi jenis-jenis peralatan USB
- Dapat diakses dengan mikrokontroler atau mikroprosesor eksternal secara paralel dengan kecepatan tinggi (2Mbps)
- Memiliki memori FIFO sebesar 320 byte.
- *Main Endpoint* memiliki *buffer* ganda sehingga meningkatkan kecepatan dan memudahkan transfer data.
- Dapat menggunakan *bwpower*
- Koneksi ke USB *bus* yang dikontrol dengan program (SoftConnect™)
- Indikator LED digunakan untuk menyatakan hubungan komunikasi yang terjadi (GoodLink™)

- Frekuensi *clock out* dapat diatur melalui program
- Rangkaian internal power-on reset
- Dapat beroperasi dengan 2 daya kerja yaitu 3,3 ± 0,3V atau 5 ± 0,5V

2.2.2. **Pin** Pada PDIUSBD12

PDIUSBD12 merapakan IC 28 pin dengan posisi pin seperti terdapat pada gambar berikut ini.



Gambar 2.10. Pin Pada PDIUSBD12

Sumber : *datasheet* PDIUSBD12

Pada tabel berikut dijelaskan tentang pin-pin pada PDIUSBD12.

Tabel 2.14. Penjelasan Pin PDIUSBD12

Simbol	Pin	Tipe	Penjelasan
Data<0>	1	IO2	Bit 0 data dua arah.
Data<1>	2	IO2	Bit 1 data dua arah.
Data<2>	3	IO2	Bit 2 data dua arah.
Data<3>	4	IO2	Bit 3 data dua arah.
Data<4>	6	IO2	Bit 4 data dua arah.
Data<5>	7	IO2	Bit 5 data dua arah.
Data<6>	8	IO2	Bit 6 data dua arah.
Data<7>	9	IO2	Bit 7 data dua arah.
RD_N	15	I	*Read Strobe* (*Active Low*).
WR_N	16	I	*Write Strobe* (*Active Low*).

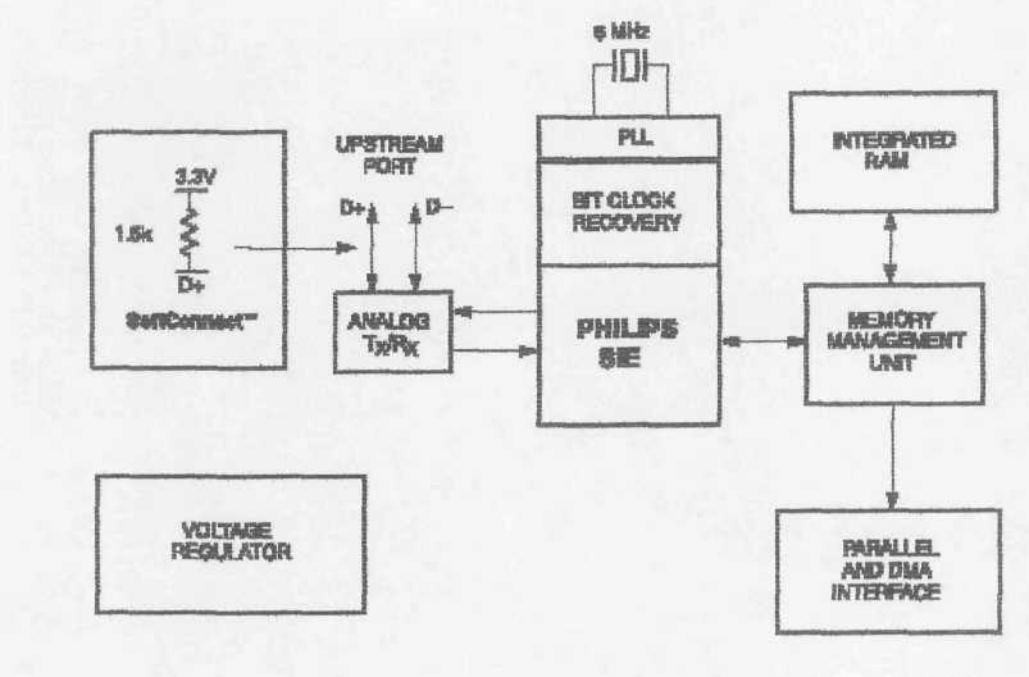
Tabel 2.14. Penjelasan Pin (sambungan)

Simbol	Pin	Tipe	Penjelasan
A0	28	I	Bit A0 = 1 menyatakan fase perintah, A0 = 0 menyatakan fase data. Bernilai *don't care* untuk jalur data dan alamat yang di-*multiplex* dan dihubungkan ke V_{CC}.
GND	5	P	*Ground*.
VDD/VCC	24	P	Tegangan *input* (4,0 – 5,5 V). Untuk menjalankan IC pada tegangan 3,3 V, maka VCC dan VOUT3.3 harus dihubungkan dengan tegangan 3,3 V.
VOUT3.3	27	P	Tegangan *output* 3,3 V.
ALE	10	I	Digunakan untuk mengakhiri *latch* alamat pada jalur data dan alamat yang di-*multiplex*. Untuk jalur data dan alamat yang terpisah, ALE dihubungkan dengan *Ground*.
CS_N	11	I	*Chip Select* (*Active Low*).
SUSPEND	12	I,OD4	*Chip* dalam kondisi *suspend*.
CLKOUT	13	O2	frekuensi *clock* yang keluarkan diatur melalui program.
INT_N	14	OD4	*Interrupt* (*Active Low*).
DMREQ	17	O4	DMA *Request*.
DMACK_N	18	I	DMA *Acknowledge* (*Active Low*).
EOT_N	19	I	Mengakhiri transfer DMA (*Active Low*). Sekaligus menjadi untuk Vbus *sensing*. EOT_N hanya dapat bekerja jika digunakan bersamaan dengan DMACK_N dan salah satu dari RD_N atau WR_N.
RESET_N	20	I	*Reset* (*Active Low* dan *Asynchronous*).Rangkaian *power-on reset* terdapat pada IC, pin ini dihubungkan dengan V_{CC}.
GL_N	21	OD8	Indikator LED GoodLink (*Active Low*).
XTAL1	22	I	Dihubungkan dengan kristal 6 MHz.
XTAL2	23	O	Dihubungkan dengan kristal 6 MHz. Jika menggunakan *clock* eksternal selain crystal, dihubungkan ke XTAL1 dan pin XTAL2 tidak dihubungkan kemana-mana.
D-	25	A	Jalur data USB D-.
D+	26	A	Jalur data USB D+.

Pada bagian tipe, huruf T menyatakan *input* dan huruf [1]O' menyatakan *output.* Angka di belakangnya menyatakan besamya arus (dalam mA). sedangkan huruf 'P' menyatakan*power* dan huruf 'A' menyatakan analog.

2.2.3. Blok Diagram

Bagian **ini** menggambarkan konsep umum **cara** kerja PDIUSBD12, gambaraya seperti terlihat pada gambar 2.11.



Gambar 2.11. Blok Diagram Rangkaian *Internal* PDIUSBD12

Sumber: *dptasheet* PDIUSBD12

Bagian-bagian yang ada yaitu:

- *Transceiver* Analog

  Bagian yang berhubungan langsung dengan kabel USB (melalui resistor)

- Pengatuf Tegangan

  Pengatur tegangan 3,3V yang terintegrasi memberi daya pada *transceiver* anaiog. Dapat juga digunakan untuk menghubungkan resistor *pull-up* eksternal 1,5KQ.

- PLL(Phase Locked Loop)

  Adanya rangkaian pengganda *(multiplier)* yang terintegrasi sehingga cukup menggunakan kristaJ 6MHz yang hemat biaya tetapi dapat menghasilkan frekuensi *clock out* hingga 48MHz.

- Philips SEE (Serial *Interface* Engine)

  Bagian ini yang mengimplementasikan protokol *layer* USB termasuk *synchronization pattern recognition,* konversi paralel/serial, bit-*stifffmg/de-stiiffing,* memeriksa dan menghasilkan CRC, verifikasi dan menghasilkan *packet identifier* (PID), pengenalan alamat, evaluasi dan menghasilkan handshake.

- Rangkaian SoftConnect

  Untuk menghubungkan *chip* dengan jalur USB dilakukan dengan menghubungkan jalur D+ denganVcc melalui resistor l,5kQ, pada *chip* ini ada resistor l,5kD yang terintegrasi dan dapat dihubungkan ke Vcc dengan mengirimkan perintah melalui program dari mikrokontroler.

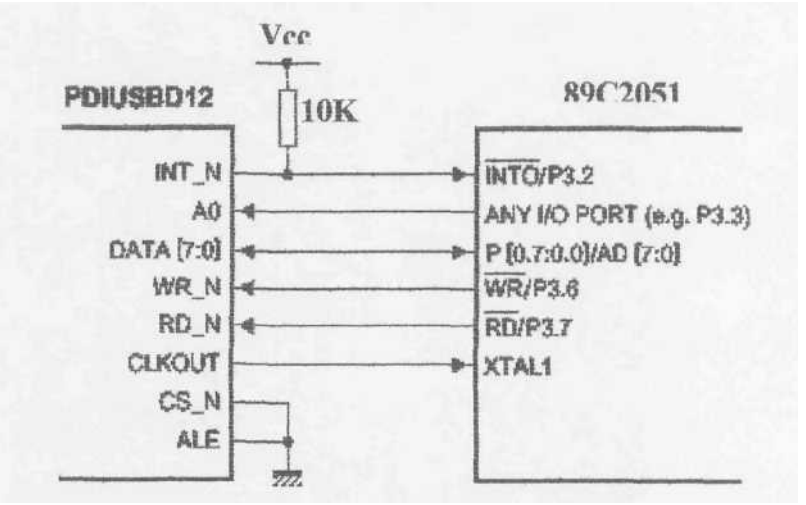
- Memory Management Unit dan Integrated RAM

  Bagian ini menjembatani perbedaan kecepatan transfer data antara USB dan mikrokontroler sehingga mikrodapat mengakses data menurut kemampuan kecepatannya tanpa kehilangan data dari USB.

- *Interface* paralel dan *interface* DMA

  Interface paralel standar ini didefinisikan sebagai *interface* yang mudah dan cepat serta dihubungkan langsung dengan bermacam-macam mikrokontroler. *Interface* DMA dimaksudkan untuk transfer data dari endpoint 2 ke memori.

Contoh hubungan paralel PDIUSBD12 dengan mikrokontroler 89C2051 dapat dilihat pada gambar 2.12. berikut



Gambar 2.12. Hubungan PDIUSBD12 Dengan 89C2051

Sumber: *Datasheet* PDIUSBD12

2.2.4. Endpoint

Endpoint pada PDIUSBD12 ini dapat digunakan dalam 4 mode, yaitu: mode 0, mode l,mode 2 dan mode3. Mode 0 ini merupakan mode non-

isochronous transfer, mode 1 untuk transfer isochronous *output*, mode 2 untuk transfer isochronous *input* dan mode 3 untuk transfer isochronous *input* dan *output.* Untuk lebih jelasnya dapat dilihat pada tabel 2.15. berikut:

Tabel 2.15. Konfigurasi *Endpoint*

Sumber: *Datasheet* PDIUSBD12

Endpoint number	Endpoint index	Transfer type	Direction[1]	Max. Packet size (bytes)
Mode 0 (Non-ISO mode)				
0	0	Control	OUT	16
	1		IN	16
1	2	Generic[2]	OUT	16
	3		IN	16
2	4	Generic[2][3]	OUT	64[4]
	5		IN	64[4]
Mode 1 (ISO-OUT mode)				
0	0	Control	OUT	16
	1		IN	16
1	2	Generic[2]	OUT	16
	3		IN	16
2	4	Isochronous[3]	OUT	128[4]
Mode 2 (ISO-IN mode)				
0	0	Control	OUT	16
	1		IN	16
1	2	Generic[2]	OUT	16
	3		IN	16
2	5	Isochronous[3]	IN	128[4]
Mode 3 (ISO-I/O mode)				
0	0	Control	OUT	16
	1		IN	16
1	2	Generic[2]	OUT	16
	3		IN	16
2	4	Isochronous[3]	OUT	64[4]
	5		IN	64[4]

[1] IN: input for the USB host; OUT: output from the USB host.
[2] Generic endpoints can be used either as Bulk or Interrupt endpoint.
[3] The main endpoint (endpoint number 2) is double-buffered to ease synchronization with the real-time applications and to increase throughput. This endpoint supports DMA access.
[4] Denotes double buffering. The size shown is for a single buffer.

Endpoint 2 merupakan endpoint utama yang digunakan untuk transfer data dalamjumlah besar, endpoint ini dilengkapi fitur-fitur:

- Bufferganda

- Dapat beroperasi pada mode DMA.
- Dapat dikonfigurasi untuk transfer data secara isochronous ataupun non-isochronous (mode buik dan interrupt)

2.2.5. Perintah Yang Dipakai

Perintah yang ada terbagi atas 3 jenis yaitu: perintah untuk inisialisasi *chip*, perintah untuk transfer data antara *host* dan mikrokontroler serta perintah umum lain dapat digunakan. Perintah untuk inisialisasi adalah perintah yang digunakan saat proses enumerasi alat USB oleh *host* USB, antara lain perintah untuk mengaktifkan *endpoint* juga perintah untuk melakukan proses *set address.* Perintah untuk transfer data digunakan untuk mengatur transfer data antara *endpoint* USB dan mikrokontroler. Sebagian besar transfer data diawali dengan *interi-upt* pada mikrokontroler oleh USB *chip.* Selain itu tnasih ada perintah lain yang digunakan secara umum.

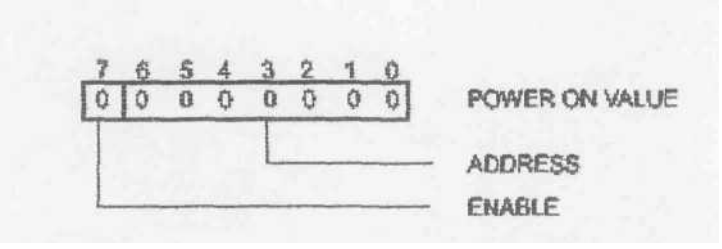
Berikut ini akan dibahas mengenai perintah yang digunakan:

o *Sel Address/Enable*

Perintah ini digunakan untuk merujuk PDIUSBD12 ke alamat yang diberikan oleh komputer.

Kode : DOh

Transaksi : menulis 1 byte.



Gambar 2.13. Alokasi Bit *Set Address/Enable*

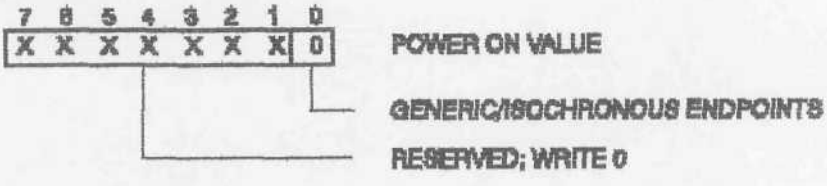
Sumber: *Daiasheei* PDIUSBD12

o *Set Endpoint Enable*

Perintah ini digunakan untuk mengaktifkan *endpoint generic* atau *isochronous.*

Kode : D8h

Transaksi : menulis 1 byte.



Gambar 2.14. Alokasi Bit *Set Endpoint Enable*

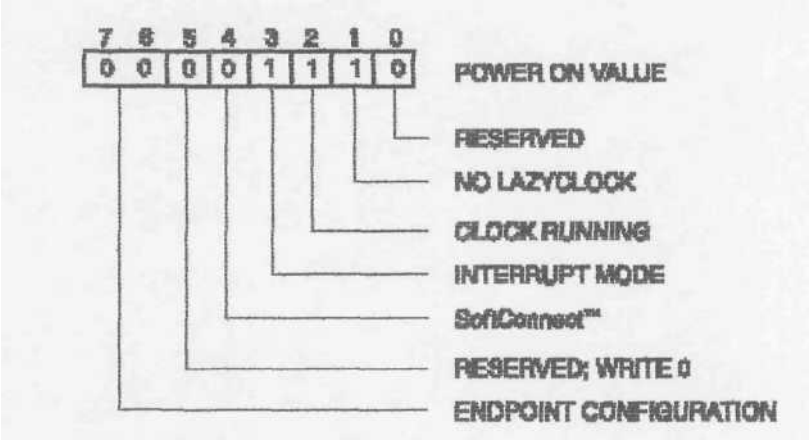
Sumber: *Datasheet* PDIUSBD12

o *Set Mode*

Perintah ini mengatur konfigurasi PDTUSBD12 yaitu: *endpoint*, SoftConnect, dan frekuensi *clock out.*

Kode : F3h

Transaksi : menulis 2 byte.



Gambar 2.15. Alokasi Bit *Set ModeByte* 1

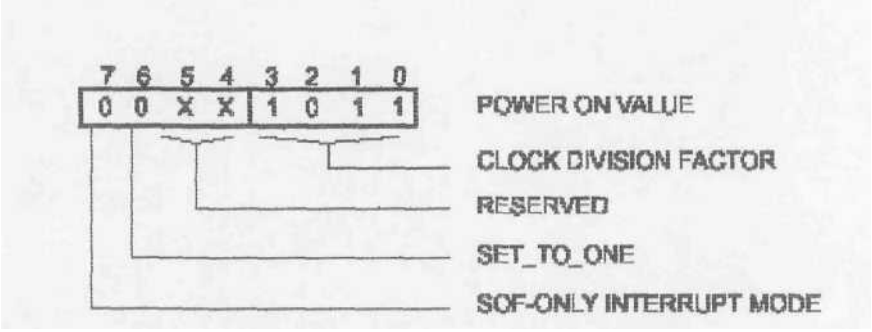
Sumber: *Dalasheel* PDIUSBD12

Tabel 2.16. Alokasi Bit *Set Mode* Byte 1

Bit	Simbol	Penjelasan
7 dan 6	Konfigurasi *Endpoint*	Nilai '00' = mode non-ISO Nilai '01' = mode ISO-OUT Nilai '10' = mode ISO-IN Nilai '11' = mode ISO-I/O
4	SoftConnect	Di-*set* '1' untuk menghubungkan resistor *pull-up* ke Vbus. Nilai tidak akan diubah oleh *bus reset*.

Tabel 2.16. Alokasi Bit *Set Mode* Byte 1 (sambungan)

3	Mode *Interrupt*	Nilai '1'menyatakan kesalahan dan 'NAK' menghasilkan *interrupt*. Nilai '0' menyatakan status saja 'OK' dilaporkan. Nilai tidak akan diubah oleh *bus reset*.
2	Mode *Clock*	Nilai '1' menunjukkan dalam status *suspend*, *clock* internal dan PLL akan tetap berjalan. Nilai '0' menunjukkan bahwa *clock* internal, osilator kristal dan PLL akan dihentikan jika tidak dibutuhkan (terutama pada saat *Suspend*). Nilai tidak akan diubah oleh *bus reset*.
1	LazyClock	Nilai '1' menunjukkan bahwa CLKOUT tidak akan berubah ke LazyClock. Nilai '0' menunjukkan bahwa CLKOUT berubah ke LazyClock1 milidetik setelah pin *Suspend* bernilai *high*. frekuensi LazyClock 30 kHz ± 40 %. Nilai tidak akan diubah oleh *bus reset*.



Gambar 2.16. Alokasi Bit *Set Mode* Byte 2

Sumber: *Datasheet* PDIUSBD12

Tabel 2.17. Alokasi Bit *Set Mocle* Byte 2

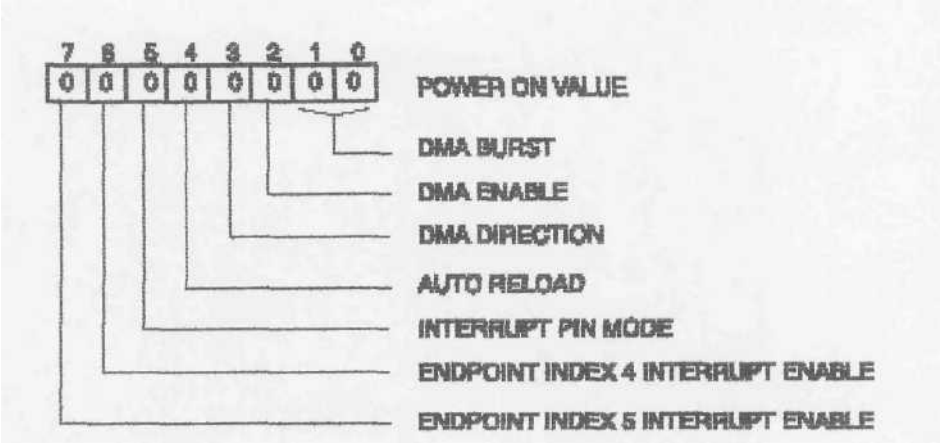
Bit	Simbol	Penjelasan
7	Mode *Interrupt* SOF	Nilai '1' menyatakan bahwa *interrupt* hanya dihasilkan oleh *Start Of Frame* (SOF) nilai Mode Pin *Interrupt* pada bit 5 Set DMA diabaikan.
6	*Set-To-One*	Harus diisi '1'. Bernilai awal '0' saat dinyalakan.
3 s/d 0	Faktor Pembagi *Clock*	Nilai ini menentukan frekuensi CLKOUT dengan rumus 48 MHz/(N+1) dimana N merupakan Faktor Pembagi *Clock*. Nilai awal 11 akan menghasilkan frekuensi 4 MHz. Nilai tidak akan diubah oleh *bus reset*.

o *Set DMA*

Perintah ini digunakan untuk mengatur dan membaca konfigufasi DMA. Tetapi pada Tugas Akhir ini, mode DMA tidak digunakan, dan perintah ini ditujukan untuk mengatur *interrupt.*

Kode : FBh

Transaksi : menulis 1 byte (untuk mode DMA ada transaksi baca 1 byte)



Gambar 2.17. Alokasi Bit *Set* DMA

Sumber: *Datasheet* PDIUSBD12

Tabel 2.18. Alokasi Bit *Set* DMA

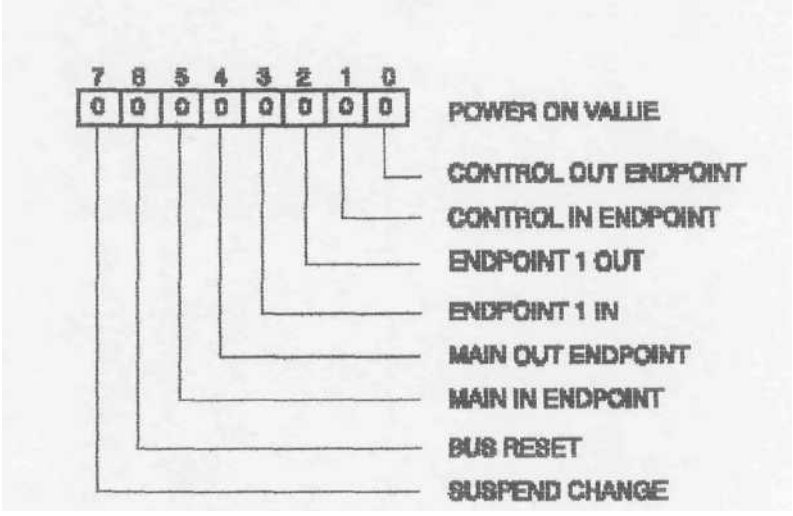
Bit	Simbol	Penjelasan
7	*Endpoint Index* 5 *Interrupt Enable*	Jika bernilai '1' maka *interrupt* akan terjadi setiap kali ada validasi *buffer*.
6	*Endpoint Index* 4 *Interrupt Enable*	Jika bernilai '1' maka *interrupt* akan terjadi setiap kali *buffer* berisi paket yang sah.
5	Mode Pin *Interrupt*	Nilai '0' menyatakan bahwa interrupt akan berasal dari bit-bit yang ada pada *interrupt register*. Nilai '1' menyatakan bahwa *interrupt* akan terjadi setiap ada *Start Of Frame* (SOF) pada *bus*.
4	*Auto Reload*	Jika bernilai '1' maka operasi DMA akan dimulai lagi dari awal secara otomatis.
3	Arah DMA	*Bit* ini menunjukkan arah arus data DMA. Nilai '1' menunjukkan arah dari memori ke PDIUSBD12 (DMA *Write*), nilai '0' menunjukkan arah dari PDIUSBD12 ke memori (DMA *Read*) melalui deklarasi pin DMREQ. *Buffer main endpoint* harus terisi penuh (untuk pembacaan DMA) atau kosong (untuk penulisan DMA) sebelum DMREQ dinyatakan. Pada mode DMA *single cycle*, DMREQ dimatikan pada penerimaan DMACK_N. Pada mode DMA *burst*, DMREQ akan dimatikan setelah sejumlah *burst* habis. Proses ini berlanjut sampai EOT_N dinyatakan bersama dengan DMACK_N dan RD_N atau WR_N, yang akan mengisi nilai '0' pada bit ini dan mengakhiri operasi DMA. Operasi DMA bisa langsung diakhiri dengan mengisi bit ini dengan '0'
1 dan 0	DMA *Burst*	Memilih panjang *burst* untuk operasi DMA: 00 DMA *single cycle* 01 DMA *burst* (4-*cycle*) 10 DMA *burst* (8-*cycle*) 11 DMA *burst* (16-*cycle*)

o *Read Inlerrupl Regisler*

Perintah ini menunjukkan sumber *interrupl. Interrupt* dari *endpoinl* dapat dibersibkan dengan membaca transaksi terakhirnya. *Interrupt* lainnya dibersihkan setelah membaca perintah ini.

Kode : F4h

Transaksi : baca 2 byte.

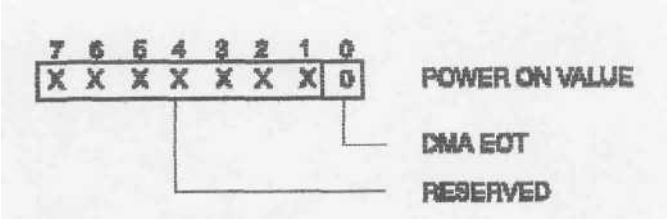


Garabar 2.18. Alokasi Bit *Interrupt Register* Byte 1

Sumber: *Datasheet* PDIUSBD12

Tabel 2.19.Alokasi Bit *Interrupt Register* Byte 1

Bit	Simbol	Penjelasan
7	*Bus Reset*	Ketika PDIUSBD12 tidak menerima 3 SOF, maka PDIUSBD12 akan memasuki status *suspend* dan bit ini bernilai *high*. Perubahan ke status *suspend* atau status *awake* akan mengubah bit ini menjadi *high* dan menghasilkan *interrupt*.
6	*Suspend Change*	Setelah *bus reset* sebuah *interrupt* akan dihasilkan dan mengisi bit ini dengan '1'. *Bus reset* sama dengan *reset* secara perangkat keras melalui pin RESET_N dengan perbedaan bahwa *bus reset* menghasilkan pemberitahuan *interrupt* dan alat akan berada pada alamat mula-mula yaitu alamat 0.



Gambar 2.19. Alokasi Bit *Interrupt Register* Byte 2

Sumber: *Datasheet* PDIUSBD12

Tabel 2.20. Alokasi Bit *Interrupt Register* Byte 2

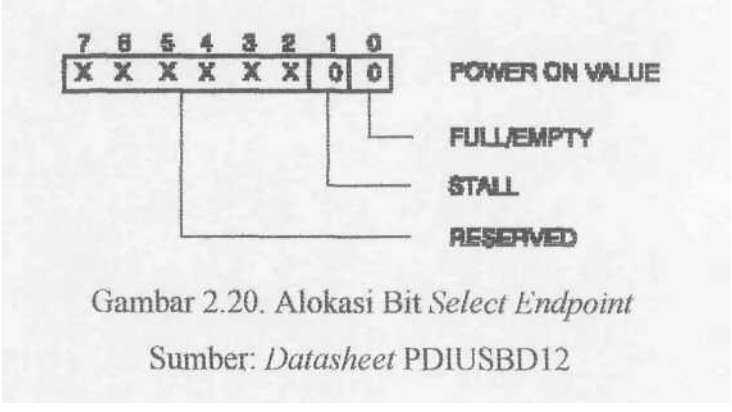
Bit	Simbol	Penjelasan
0	DMA EOT	Bit ini menyatakan-bahwa operasi DMA telah lengkap.

o *Select Endpoint*

Perintah ini mengarahkan penunjuk internal pada awal *buff'er* yang dipilih. Bisa juga diikuti dengan pembacaan data yang berisi status *endpoint* dan status *bitffer.*

Kode : OOh sampai 05h

Transaksi : baca 1 byte (tidak mutlak).



Gambar 2.20. Alokasi Bit *Select Endpoint*

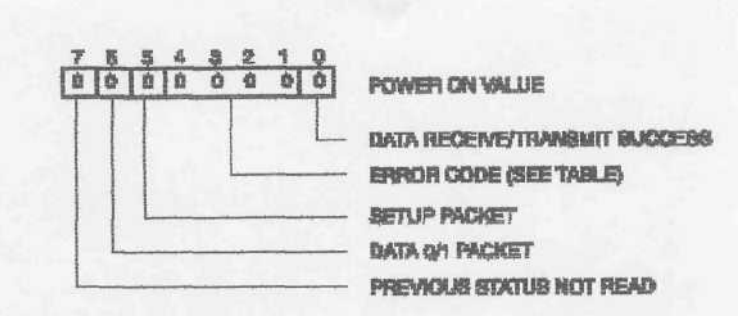
Sumber: *Datasheet* PDIUSBD12

o *Read Last Transaction Status Register*

Perintah ini diikuti data yang berisi transaksi terakhir dari *endpoint* yang dipilih. Selain itu perintah ini juga membersihkan *interrupt flag.*

Kode : 40h sampai 45h

Transaksi : baca 1 byte.



Gambar 2.21. Alokasi Bit *ReadLast Transaction Status Register*

Sumber: *Dalasheet* PDIUSBD12

Tabei 2.21. Alokasi Bit *Read Last Transaction Status Register*

Bit	Simbol	Penjelasan
7	Status Sebelumnya Belum Terbaca	Nilai '1' menunjukkan bahwa kejadian kedua terjadi sebelum status sebelumnya dibaca.
6	Paket DATA 0/1	Nilai '1' menunjukkan bahwa paket terakhir yang dikirim/diterima memiliki PID DATA1.
5	Paket SETUP	Nilai '1' menunjukkan bahwa paket terakhir yang diterima memiliki *token* SETUP (nilai bit ini selalu '0' untuk *buffer* IN).
4 s/d 1	Kode Kesalahan	Lihat tabel 3.4.7.2. Kode Kesalahan.
0	Pengiriman/ Penerimaan Data	Nilai '1' menunjukkan bahwa data telah berhasil dikirim/diterima.

Tabel 2.22. Kode Kesalahan

Kode (biner)	Penjelasan
0000	Tidak ada kesalahan.
0001	Kesalahan PID; bit 7 s/d 4 bukan kebalikan dari bit 3 s/d 0.
0010	PID tidak diketahui; urutannya benar tetapi PID tersebut tidak ada.
0011	Paket yang tidak diharapkan; paket bukan tipe yang diharapkan (*token*, data, atau *acknowledge*), atau *token* SETUP pada *endpoint* selain *control endpoint*.
0100	Kesalahan CRC *token*.
0101	Kesalahan CRC data.
0110	Waktu melebihi batas (*time out*).
0111	Tidak pernah terjadi.
1000	Akhir paket yang tidak diharapkan.
1001	Mengirimkan atau menerima NAK.
1010	Mengirimkan STALL, *token* diterima tetapi *endpoint* dalam status *stall*.
1011	Kesalahan batas (*overflow*), paket yang diterima melebihi ruang *buffer* yang tersedia.
1101	Kesalahan bit*stuff*.
1111	PID DATA salah; PID DATA yang diterima bukan yang diharapkan.

o *Read Buffer*

Perintah ini diikuti oleh beberapa proses pembacaan data yang mengisi *huffer endpoint.* Setiap seiesai pembacaan. penunjuk *buffer* internai akan ditambah 1. Penunjuk *buffer* tidak ditaruh di posisi teratas pada *buffer* oleh perintah ini. Hal ini berarti bahwa pembacaan atau penulisan *buffer* bisa disela oleh perintah lain kecuali perintah *Select Endpoint.* Urutan data pada buffer diatur seperti berikut:

byte 0: bersifat *reserved*, bisa berisi bermacam-macam nilai

- byte 1: jumlah data (byte)
- byte 2: byte data 1
- byte3: bytedata2
- dst.

Kode :FOh

Transaksi : baca maksimal 130 byte.

o *Write Buffer*

Perintah ini diikuti oleh beberapa proses penulisan data yang akan mengisi *huffer endpoim,* Urutan data sama dengan urutan data pada perintah *Read Riiffer,* Byte pertama harus selalu 0, Byte kedua berisi jumlah data yang akan dikirim. Dan byte-byte berikutnya merupakan data yang akan dikirimkan. Untuk pengiriman *non-isochronous (hulk* atau *interrupt*\ butYer harus diisi secara penuh sebelum data dikirim ke *host* dan perpindahan ke *buffer* berikutnya terjadi.

Penulisan atau pembacaan melebihi kapasitas *bvffer* atau penulisan ke *buffer* OUT atau pembacaan dari *buffer* IN akan mengakibatkan proses yang salah.

Kode : FOh

Transaksi : menulis maksimal 130 byte.

o *Clear Buffer*

Setelah paket diterima secara lengkap, *biiffer full flag* intemal akan diaktifkan, Semua paket berikutnya akan ditolak dengan mengirimkan NAK. Setelah *micrvcontrolter* membaca data yang diterima, maka buffer harus

dibersihkan dengan perintah ini. Jika *buffer* telah bersih, maka paket berikutaya bisa diterima,

Kode : F2h

Transaksi : tidak ada

o *Validate Buffer*

Setelah *microcontroller* mengisi data pada *buffer* IN, maka *microcontroller* harus mengaktifkan *buffer full flag* dengan perintah ini. Hal ini menyatakan bahwa data dalam *bujfer* sudah benar dan bisa dikirim ke *host* saat *token* IN selanjutnya diterima.

Kode : FAh

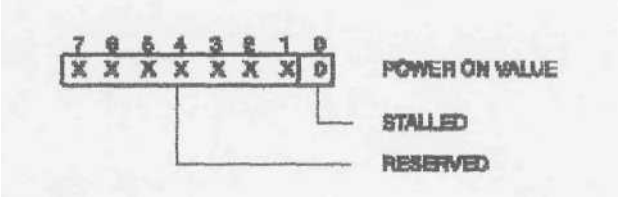
Transaksi : tidak ada

o *Set Endpoint Status*

Pengiriman perintah *Set Endpoint Status* akan melakukan inisialisasi *endpoint* yang dimaksud dan jika *endpoint* tersebut dalam keadaan *stall*, maka perintah ini akan menghilangkannya. Perintah ini akan menghapus *bujfer* dan jika berada pada *buffer* OUT maka *bujfer* akan menunggu DATA 0 PID, jika berada pada *hujfer* TN maka *huffer* akan menulis DATA 0 PTD.

Kode : 40h - 45h

Transaksi : menulis 1 byte



Gambar 2.22. Alokasi Bit *Set Endpoint Status*

Sumber: *Datasheet* PDIUSBD12

o *Acknowledge Setup*

Penerimaan paket SETUP menghapus *buffer* IN dan mematikan perintah *Validate Buffer* dan *Cleur Buffer* untuk *endpomt* IN dan OUT. Untuk mengaktifkan kembali perintah-perintah ini diperlukan perintah *Acknowledge*

Seiup. Hal ini memastikan paket SETUP terakhir tetap berada dalam *buffer* dan tidak ada paket yang bisa dikirimkan kembali ke *hosi.* hingga *microcontroller* telah menyatakan secara ekspiisit telah adanya paket SETUP.

Kode :Flh

Transaksi : tidak ada

2.3. **Mouse**

Mouse merupakan sebuah *pointing device* yang menakjubkan, prinsip kerjanya sederhana namun sangat efektif dan mempermudah penggunaan komputer. *Mouse* pertama kali dipopulerkan pada tahun 1984 bersamaan dengan komputer Apple Macintosh dan sejak saat itu *mouse* menjadi perlengkapan yang memudahkan pemakaian komputer.

2.3.1. Tujuan Pembuatan *Mouse* Dan Jenis-Jenis *Mouse.*

Tujuan utama pembuatan *mouse* adalah untuk menerjemahkan pergerakan tangan kita sehingga dapat dimengerti oleh komputer. Dewasa ini teknologi telah semakin canggih sebingga ada banyak cara untuk mencapai tujuan diciptakannya *mouse.* Sekarang ada bermacam-macam *mouse* berdasarkan koneksinya, antara lain *mouse* dengan *interface* seriai, *moitse* dengan *interface* PS/2, *mouse* dengan *interface* USB, dan *mouse* dengan *interface wireless. Mouse* dengan *interface wireless* ini dapat dibedakan lagi *mouse* dengan *interface wireless* yang menggunakan *Radio Frequency* dan *mouse* dengan *interface wireless* yang menggunakan *Infra Red.*

Mouse dengan *interface* serial menggunakan koneksi 4 pin yaitu Vcc, Gnd, Rx dan Tx. *Mouse* dengan *interface* PS/2 juga menggunakan koneksi 4 pin hanya saja berapa Vcc, Gnd, *Clock* dan jalur data. *Mouse* dengan *interface* USB menggunakan koneksi Vcc, Gnd, D- dan D+. *Mouse* dengan *interface wireless* ada yang mengirimkan data melalui gelombang radio, ada juga yang menggunakan gelombang infra merah. Keuntungan dari *rnouse* dengan *interface wireless* adalah praktis (karena tidak menggunakan kabel yang kadang mengganggu) dan juga jarak antara *mouse* dengan *interface wireless* dan

komputer lebih jauh daripada *mouse* dengan *inierface* selain *wireless* yang jaraknya terbatas kabel.

Berdasarkan cara mengetahui pergeserannya *mou.se* dibedakan atas *mouse* konvensionai dan *mouse optical. Mouse* konvensional terdiri atas komponen-komponen sebagai berikut yaitu: sebuah bola, dua buah *shaft* yang terhubung dengan piringan berlubang dan masing-masing terletak saling tegak lurus untuk mendeteksi pergerakan berdasarkan snmbu X dan sumbu Y, juga ada 4 pasang sensor infra-merah masing-masing 2 pasang untuk setiap piringan dan ada sebuah *chip processor. Processor* digunakan untuk memproses pulsa infra-merah (yang diterima dari piringan-piringan yang memberikan pulsa sesuai pergeseran pada sumbu X dan sumbu Y) menjadi data biner yang ditransimisikan ke komputer. *Monse optical* yang pertama kali diperkenalkan pada tahun 1999 memiliki sebuah kamera kecil untuk menangkap pantulan cahaya LED dan sebuah *Digital Signal Processor* (DSP). Garabar yang ditangkap oleh kamera diproses oleh DSP yang dapat memproses 18 *Milliom Imtruction per Second* (MIPS) dan dapat diketahui pola dari gambar yang masuk, kemudian *mouse* mengirimkan data ke komputer berdasarkan *output* dari DSP. Keuntungan *mouse* jenis *opiical* ini antara lain: *mouse* tertutup (bagian dalam *mouse* tidak berdebu dan tidak mengganggu proses petnbacaan gerakan *mouse),* pergerakan *mouse* diproses secara cepat sehingga respon gerakan pada komputer juga lebih halus dan tidak membutuhkan permukaan khusus misalnya *mouse pad* dan karena tidak ada bagian yang bergerak seperti pada *mouse* konvensional, kemungkinan terjadi kerusakan lebih kecil.

2.3.2. *Mouse* Yang Menggunakan *Interface* PS/2

Mouse yang menggunakan *intetface* PS/2 memiliki 4 mode kerja, yang masing-masing berbeda dalam hal cara *monse* mengirimkan data. Yang pertama adalah mode *reset* (saat dimana *mouse* pertama kali menerima tegangan ataupun saat *mouse* menerima perintah *reset),* pada mode ini data tidak dikirimkan sampai *mouse* menerima perintah untuk mengaktifkan transmisi data. Berikutnya adalah mode *stream,* ini adalah mode yang umumnya digunakan sebagian besar *software* yang memerlukan *mome.* Pada mode ini data dikirimkan saat terjadi perubahan

kondisi tombol atau pergeseran *mouse.* Ada juga mode *remoie* dimana data hanya dikirimkan saat komputer meminta data, sebelum diminta *mouse* hanya mencatat jumiah pergeseran yang terjadi sesudah komputer terakhir meminta data.yang terakhir adaiah mode *wrap* dimana *mouse* mengirimkan ulang semua data yang diterima kecuali data untuk *reset* dan data untuk *reset* mode *wrap* (kembali pada kondisi sebelum mode *wrap).*

Seperti telah disebutkan diatas, sebelum digunakan *mouse* yang menggunakan *interface* PS/2 harus diinisialisasi terlebih dahulu. Setelah proses inisialisasi baruiah *mouse* mulai mengirimkan data pergeseran atau perubahan kondisi yang terjadi pada tombolnya. Untuk melakukan inisialisasi mula-mula *mouse* diberi perintah untuk melakukan *reset (mouse)* akan menjalankan proses *self-test.* Selanjutnya *mouse* berada pada kondisi asal yaitu: *sample rate* 100 *samples/sec,* resolusi 4 *counts/mm, scaling* 1:1 dan transmisi data pada kondisi tidak aktif. Setelah itu *mouse* harus diberi perintah untuk memulai transmisi data (0F4h) sehingga saat terjadi pergeseran atau ada perubahan kondisi pada tombol *mouse* mengirimkan data. Pada tabel 2.23. terdapat perintah-perintah yang dapat dikirimkan ke *mouse* dan kegunaan dari setiap perintah.

Tabel 2.23. Perintah Yang Dapat Dikirimkan Pada *Mouse* PS/2

Command	Hex Value
Reset *Mouse* (*Mouse* Returns AA, 00 after self-test)	FF
Resend Message	FE
Set to Default Values	F6
Enable *Mouse* Streaming Mode (*Mouse* starts sending data packets at default rate)	F4
Disable Streaming Mode	F5
Set Sampling rate (XX is the number of packets per second)	F3, XX
Read Device Type	F2
Set Remote Mode	EE
Set Wrap Mode *Mouse* returns data sent by system	EC
Read Remote Data *Mouse* sends 1 data packet	EB
Set Stream Mode	EA
Status Request *Mouse* returns 3-bytes with current settings	E9

Tabel 2.23. Perintah Yang Dapat Dikirimkan Pada *Mouse* PS/2 (sambungan)

Set Resolution XX is 0, 1, 2, 3	E8, XX
Set Scaling 2 to 1	E7
Reset Scaling	E6

Pada tabel 2.24. berikut adalah respon yang dikirimkan *mouse* terhadap perintah yang diterima.

Tabel 2.24. Respon *Mouse* Terhadap Perintah

Message Sent	**Hex Value**
Resend Message	FE
2 Bad messages in a row	FC
Mouse Acknowledge Command *(Sent by mouse after each command byte)*	FA
Mouse passed self-test	AA

Untuk melakukan proses pengiriman data pada *mouse* harus diperhatikan urutan langkah-langkah seperti dibawah ini:

- Jalur *clock* diberi logika *law* seiama lOOus
- Jalur data diberi logika *low*
- Jalur *clock* dikembalikan ke logika *high*
- Tunggu *mouse* membuat jalur *clock* berlogika *low*
- Kirimkan data bit 1
- Tunggu *mouse* membuat jalur *clock* berlogika *high*
- Tunggu *mouse* membuat jalur *clock* berlogika *low*
- Ulangi langkah 5-7 untuk mengirimkan data 8 bit dan parity bit
- Jalur data diberi logika *high*
- Tunggu *mouse* membuat jaiur data berlogika *low*
- Tunggu *mouse* raembuat j alur *clock* berlogika *low*
- Tunggu *mouse* meimbuat jaiur *clock* dan beriogika *high*

2.3.3. Data Yang Dikirimkan *Mouse* Yang Menggunakan *Interface* PS/2

Mouse yang menggunakan *interface* PS/2 akan mengirimkan data sebanyak 3 byte setiap kali terjadi pergeseran atau perubahan kondisi pada tombol (ditekan/diiepas). Data tersebut dikirimkan secara serial, dimana setiap byte data terdiri dari 11 bit. Setiap pengiriman terdiri dari *start* bit, 8bit data, *panty* bit dan *stop* bit. *Start* bit bernilai logika *low* sedangkan *stop* bit bemilai logika *high. Parity* bit menggunakan sistem *parity* ganjil *(oddparity).* Byte pertama dari 3 byte data yang dikirimkan *mouse* menyatakan kondisi button kiri pada bit ke-0, kondisi button kanan pada bit ke-1, kondisi button tengah pada bit ke-2 (pada *mouse 2* tombol bit ini berlogika 0), bernilai 1 pada bit ke-3, arah pergerakan sumbu X pada bit ke-4, arah pergerakan sumbu Y pada bit ke-5, kondisi X *overflow* pada bit ke-6 dan kondisi Y *overflow* pada bit ke-7. Byte kedua berisi nilai pergerakan pada sumbu X (berisi data Olh atau OFFh) dan byte ketiga berisi nilai pergerakan pada sumbu Y (beisi data Olh atau OFFh). Tabel 2.25. menyatakan data yang ditransmisikan oleh *mouse* yang menggunakan *interface* PS/2 secara lengkap.

Tabel 2.25. Transmisi Data Dari *Mouse* PS/2

	Bit [0]	Bit [1]	Bit [2]	Bit [3]	Bit [4]	Bit [5]	Bit [6]	Bit [7]	Bit [8]	Bit [9]	Bit [10]
Packet [0]	Start Bit	Y Data *Overflow*	X Data *Overflow*	Y Data Sign Bit	X Data Sign Bit	1	*Middle Button* Bit	*Right Button* Bit	*Left Button* Bit	Parity Bit	Stop Bit
Packet [1]	Start Bit	X Position Bit [7] MSB	X Position Bit [6]	X Position Bit [5]	X Position Bit [4]	X Position Bit [3]	X Position Bit [2]	X Position Bit [1]	X Position Bit [0] LSB	Parity Bit	Stop Bit
Packet [2]	Start Bit	Y Position Bit [7] MSB	Y Position Bit [6]	Y Position Bit [5]	Y Position Bit [4]	Y Position Bit [3]	Y Position Bit [2]	Y Position Bit [1]	Y Position Bit [0] LSB	Parity Bit	Stop Bit

Tabel 2.26. berikut akan menjelaskan isi dari 3 byte data yang dikirimkan berdasarkan pergeseran *mouse* ataupun perubahan kondisi tombol pada *mouse* yang menggunakan *interface* PS/2.

Tabel 2.26. Data Yang Dikirimkan *Mouse* PS/2 Sesuai Aktivitasnya.

Perubahan Pada *Mouse*	byte1, byte2, byte3
1 gerakan ke atas	08,00,01
1 gerakan ke bawah	28,00,FF
1 gerakan ke kanan	08,01,00
1 gerakan ke kiri	18,FF,00
Button kiri ditekan	09,00,00
Button kiri dilepas	08,00,00
Button tengah ditekan	0C,00,00
Button tengah dilepas	08,00,00
Button kanan ditekan	0A,00,00
Button kanan dilepas	08,00,00