PERANCANGAN TOUCHPAD PS/2

BERBASIS MIKROKONTROLER AT89C51

TUGAS AKHIR

Diajukan Sebagai Salah Satu Syarat Untuk Memperoleh

Gelar Sarjana Teknik Elektro

Disusun oleh :

Nama : **Rahan Sukma Yudha**

No.Mahasiswa : **03524038**

JURUSAN TEKNIK ELEKTRO

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

YOGYAKARTA

2007

LEMBAR PENGESAHAN PENGUJI

PERANCANGAN TOUCHPAD PS/2 BERBASIS MIKROKONTROLER AT89C51

TUGAS AKHIR

oleh :

Nama : Rahan Sukma Yudha

No. Mahasiswa : 03524038

Telah Dipertahankan di Depan Sidang Penguji sebagai Salah Satu Syarat untuk Memperoleh Gelar Sarjana Teknik Elektro Fakultas Teknologi Industri Universitas Islam Indonesia

Yogyakarta, Agustus 2007

Tim Penguji,

Tito Yuwono, ST, M.Sc.
Ketua

Yusuf Aziz Amrullah, ST.
Anggota I

Wahyudi Budi Pramono, ST
Anggota II

Mengetahui,
Ketua Jurusan Teknik Elektro
Universitas Islam Indonesia

Tito Yuwono, ST, M.Sc.

HALAMAN PERSEMBAHAN

Tugas akhir ini Ananda persembahkan Kepada :

Ayahanda dan Ibunda tercinta

Atas segalanya yang telah diberikan kepadaku

Teman beserta keluarga besarku yang selalu memberikan perhatian, semangat, motifasi dan do'a untukku

MOTTO

"Sebaik-baik manusia adalah orang yang banyak manfaatnya (kebaikannya) kepada manusia lainnya."

(H.R. Qadla'ie dari Jabir)

"Hari bekerja untuk si pemalas adalah besok, dan hari liburnya adalah hari ini."

(Jhon Wasley)

"Tuhanku (Allah SWT) mendidikku lalu dia mendidik aku sebaik-baiknya."

(H.R. Sam'ani)

KATA PENGANTAR

بسم الله الرحمن الرحيم

Assalamu'alaikum Wr. Wb

Puji syukur kehadirat Allah SWT yang telah melimpahkan rahmat, taufiq dan hidayah-Nya, karena ridho-Nya penyusun dapat menyelesaikan penyusunan tugas akhir (TA) dengan judul **"PerancanganTouchpad PS/2 Berbasisi Mikrokontroler AT89C51"** dan tidak lupa juga dipanjatkan shalawat serta salam pada junjungan Nabi besar Muhammad SAW beserta keluarga dan pengikutnya sampai akhir zaman.

Adapun maksud dari penyusunan tugas akhir ini adalah untuk memenuhi kurikulum S-1 Jurusan Teknik Elektro, Fakultas Teknologi Industri, Universitas Islam Indonesia.disamping itu bertujuan untuk menambah pengetahuan terhadap ilmu yang dipelajari di bangku perkuliahan untuk diterapkan pada aplikasi sesungguhnya.

Dalam menyusun laporan tugas akhir ini tidak terlepas dari berbagai pihak yang memberikan bantuan dan dukungan. Untuk itu pada kesempatan ini penyusun mengucapkan banyak terima kasih yang sebesar-besarnya kepada :

1. Ayahanda Widyatmanto dan Ibunda Siti Murniyati yang senantiasa memberikan dukungan semangat, perhatian, moril, materil dan do'a setiap saat.
2. Bapak Fathul Wahid,ST,M.Sc, selaku Dekan Fakultas Teknik Industri

3. Bapak Tito Yuwono,ST,M.Sc, selaku Ketua Jurusan Teknik Elektro Fakultas Teknologi Industri Universitas Islam Indonesia dan juga selaku Dosen Pembimbing I yang telah memberikan saran-saran, kritik serta bimbingan sehingga penyusun dapat menyelesaikan penyusunan tugas akhir dengan baik..
4. Bapak Yusuf Aziz Amrullah, ST, selaku Dosen Pembimbing II yang telah memberikan saran-saran, kritik serta bimbingan sehingga penyusun dapat menyelesaikan penyusunan tugas akhir dengan baik.
5. Seluruh keluarga besarku yang selalu mendoakan dan memberikan dukungan dalam penyusunan tugas akhir.
6. Dani Raharjo, Juny Suprapto dan Galih Mustiko Aji yang telah berbaik hati meluangkan waktu dan ilmunya.
7. Rekan-rekanku para anoman yang telah banyak memberikan bantuan. Serta seluruh rekan-rekan Teknik Elektro Universitas Islam Indonesia yang tidak bisa disebutkan satu-persatu.
8. Dosen dan karyawan Fakultas Teknologi Industri, Ka.Lab Jurusan Teknik Elektro atas waktu, tempat dan ilmu yang diberikan.
9. Seluruh pihak yang tidak dapat disebutkan satu-persatu, yang telah memberikan *support* dan do'a.

Penyusun telah berupaya yang terbaik dalam menyusun tugas akhir ini, namun penyusun menyadari bahwa penulisan laporan ini tidak luput dari kekurangan, maka kritik dan saran yang konstruktif dari semua pihak sangat

diperlukan untuk penulisan laporan yang selanjutnya dan penyusun terima dengan sepenuh hati sebagai bahan untuk peningkatan kemampuan dan keterampilan penyusun di lain kesempatan.

Akhirnya penyusun berharap semoga laporan ini dapat bermanfaat dan berguna bagi penyusun dan pembaca serta menjadikan amal ibadah yang diterima di sisi-Nya. Amin.

Wassalamu'alaikum Wr. Wb.

Yogyakarta, Agustus 2007

Penyusun

ABSTRAK

Touchpad adalah sebuah *input device* yang sering digunakan pada komputer *laptop*. *Touchpad* digunakan sebagai penggerak *cursor* dengan memanfaatkan pergerakan jari pemakai, atau dengan kata lain sebagai pengganti fungsi *mouse* pada komputer *desktop* biasa. *Touchpad* PS/2 dapat digunakan pada komputer *desktop* biasa yang memiliki jalur komunikasi PS/2, sehingga tidak hanya laptop memiliki *touchpad*, tetapi komputer *desktop* biasa juga bisa menggunakan *touchpad*. Sistem pada *touchpad* dikendalikan oleh sebuah *chip* mikrokontroler ATMEL AT89C51 yang digunakan sebagai modulator dan sebagai pengolah masukan data-data dan menghasilkan keluaran sesuai dengan kebutuhan. Masukan pada alat ini berupa *optocoupler* yang ditempatkan sepanjang sumbu X sebanyak 17 pasang dan sumbu Y sebanyak 11 pasang serta tiga buah *pushbuttons*. *Optocoupler* berfungsi sebagai pembaca posisi benda secara dua dimensi, sedangkan *pushbuttons* digunakan sebagai pengganti tombol-tombol yang ada pada *mouse* biasa. Cahaya yang dipancarkan oleh IR LED berupa cahaya *infrared* yang telah mengalami modulasi sehingga akan memperjauh jarak pancar dan lebih kebal terhadap gangguan dari luar. *Demodulator* digunakan untuk mendapat kembali data dari sinyal modulasi yang dipancarkan IR LED. *Output* mikrokontroler berupa 3 *byte* data yang berisi perintah-perintah untuk mengendalikan posisi dan keadaan *cursor* pada layar monitor. *Touchpad* terhubung dengan komputer melalui jalur PS/2 *port* yang menggunakan komunikasi dua arah secara *bidirectional synchronous serial*. Fungsi-fungsi khusus juga ditambahkan pada *touchpad* PS/2 ini seperti *single tap* dan *double tap*. *Touchpad* memiliki keunggulan dalam hal penempatan alat sehingga dapat digabungkan langsung dengan *casing* komputer ataupun *monitor*. *Touchpad* PS/2 dapat digunakan untuk menggerakkan *cursor* ke segala arah, click kiri, click kanan dan click kiri seperti pada *mouse* standar biasa.

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1 Latar Belakang

Teknologi yang berkembang dengan sangat pesat, terutama dalam bidang elektronika *digital* dan di bidang IT, telah mampu menghasilkan suatu perangkat komputer *notebook / laptop* dengan ukuran yang jauh lebih kecil dari komputer *desktop* biasa. *Notebook* atau *laptop* memiliki kinerja yang tidak kalah dari komputer *desktop* yang ukurannya jauh lebih besar dan lebih berat.

Salah satu hal yang membedakan antara *laptop* dengan komputer *desktop* adalah pada *laptop* telah disediakan *touchpad* sebagai pengganti dari fungsi *mouse*. *Touchpad* lebih dipilih untuk diimplementasikan pada *laptop* dibandingkan dengan *mouse* karena ukuran bidang kerjanya yang lebih kecil dan ukuran fisik yang juga lebih kecil. Kriteria ukuran inilah yang terus diusahakan oleh para produsen *laptop* agar dalam perkembangannya laptop tetap mempunyai fungsi yang sama dengan komputer *desktop* tetapi dengan ukuran yang jauh lebih kecil. *Touchpad* pada *laptop* lebih mudah digunakan karena dengan hanya menempelkan dan menggerak-gerakkan jari, posisi *cursor* dapat dikendalikan layaknya sebuah *mouse*.

Touchpad mempunyai beberapa keunggulan dibanding dengan mouse. Salah satunya yaitu kemampuan *touchpad* untuk dapat digunakan pada media

yang tegak/*vertical*. Berbeda dengan mouse yang harus menggunakan alas yang *horizontal*. Keunggulan inilah yang membuat *touchpad* dapat dipasangkan sekalipun pada objek tegak seperti tembok, pintu, *dashboard* mobil, bahkan layar monitor pun dapat dipasang *touchpad* sehingga dapat menghemat tempat.

Atas berbagai pertimbangan di atas maka dirancanglah perangkat keras elektronik yang berfungsi sebagai *touchpad* untuk dapat digunakan pada komputer *desktop* biasa melalui jalur komunikasi PS/2 *mouse port*. Perancangan perangkat keras *touchpad* menggunakan mkrokontroler ATMEL AT89C51 sebagai pusat pengendali, LED *infrared* dan *photodiode* sebagai sensor gerak, serta *push button* sebagai tombol yang berfungsi sama seperti tombol pada *mouse* biasa.

1.2 Rumusan Masalah

Dari latar belakang di atas, maka dapat dirumuskan permasalahan sebagai berikut "Bagaimana mendesain dan merealisasikan suatu alat yang dapat berfungsi sebagai *touchpad,* dan dapat digunakan pada komputer *desktop* untuk menggantikan fungsi *mouse* biasa".

1.3 Batasan Masalah

Pembahasan tentang *touchpad* PS/2 ini mempunyai cakupan yang luas. Untuk membatasi masalah agar tidak meluas kepermasalahan lain dan lebih terarah sebagaimana tujuan, penulis membatasi penelitian agar memperoleh suatu solusi yang diinginkan. Batasan-batasan di sini antara lain :

1. Perancangan sistem terdiri dari *sensor/transducer* dan sistem kendali/kontrol.
2. *Sensor* yang digunakan yaitu *infrared detector*, terdiri dari 11 pasang LED *infrared* dan *photodiode* untuk sumbu Y, sedangkan sumbu X terdiri dari 17 pasang.
3. Alat dapat berkomunikasi/berhubungan dengan komputer melalui jalur PS/2 *port*.
4. Pengendalian dilakukan oleh mikrokontroler AT89C51.
5. Fungsi-fungsi yang dapat dilakukan *touchpad* adalah fungsi-fungsi dasar dari sebuah *mouse* standar seperti menggerakkan *cursor* ke segala arah, *click* kiri, *click* tengah dan *click* kanan.

1.4 Tujuan Penelitian

Tujuan dari pembuatan tugas akhir ini adalah untuk membuat *touchpad* yang dapat digunakan pada komputer *desktop* melalui jalur PS/2 *mouse port* untuk menggantikan fungsi *mouse* biasa. Dengan demikian diharapkan tercipta *pointer device* yang lebih fleksibel penggunaanya dibanding dengan *mouse*. Sedangkan tujuan sekundernya adalah merealisasikan dan menerapkan pengetahuan yang didapat dan dipelajari selama perkuliahan.

driver yang digunakan, pada model tertentu dapat dilakukan fungsi *click* tombol dengan cara menyentuhkan jari kemudian melepaskan dengan cepat pada bidang *touchpad.*

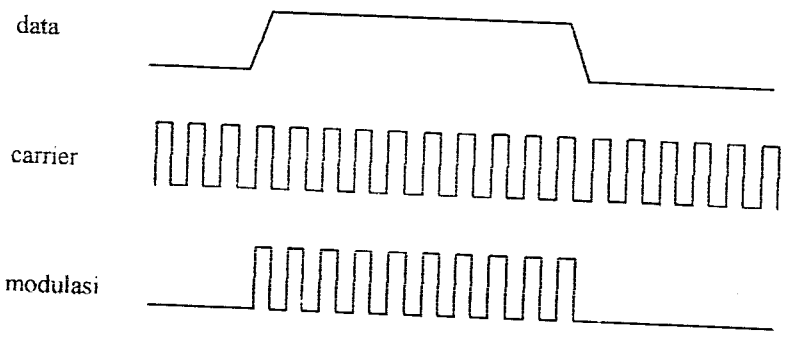
Touchpad mempunyai dua buah metode yang paling sering digunakan untuk mendeteksi jari. Pertama dengan metode *capacitance shunt* yaitu mendeteksi adanya perubahan *capacitance* antara *transmitter* dengan *receiver* yang diletakkan di sisi yang berlawanan pada bidang sentuh. *Transmitter* menghasilkan medan listrik dengan oscilasi 2-300 kHz. Jika sebuah titik *ground* (contohnya jari) yang terletak diantara *transmitter* dan *receiver*, beberapa dari garis medan terpotong sehingga menurunkan nilai *capacitance*. Metode yang kedua dengan cara menggunakan *matrix conductors*. *Conductor* yang jumlahnya banyak dipasang secara berjajar dalam dua buah lapisan yang dipisahkan oleh udara. *Conductor* pada lapisan pertama diposisikan tegak lurus dengan *conductor* pada lapisan kedua. Sebuah sinyal berfrekuensi tinggi di alirkan secara sekuensial berpasangan diantara matriks dua dimensi yang terbentuk dari barisan *conductors*. Besarnya arus yang mengalir pada titik-titik *nodes* sebanding dengan besarnya *capacitance*. Ketika jari (*ground*) diletakkan diantara titik tersebut maka beberapa dari garis medan listrik berubah yang mengakibatkan perubahan *capacitance* pada titik tersebut.

2.2.1.1 *Sensor Infrared*

Komponen yang dikhususkan untuk menerima cahaya inframerah, memiliki desain lubang tempat menerima cahaya yang khusus, sehingga dapat mengurangi interferensi dari cahaya non inframerah. Oleh sebab itu *sensor* inframerah yang baik, biasanya jendela (pelapis dari silikon) berwarna biru tua keungu-unguan. *Sensor* ini biasa digunakan untuk aplikasi inframerah di luar rumah (*outdoor*).

2.2.1.2 *Transmitter Infrared*

Inframerah dapat digunakan baik untuk memancarkan data maupun sinyal suara. Keduanya membutuhkan sinyal *carrier* untuk membawa sinyal data maupun sinyal suara tersebut sampai pada *receiver*. Sinyal data akan dimodulasikan dengan sinyal *carrier* sehingga akan menghasilkan modulasi FM. Teknik ini digunakan karena lebih kebal terhadap perubahan amplitudo apabila sinyal mengalami gangguan di udara.



Gambar 2.2 Modulasi Sinyal

2.2.1.3 *Receiver Infrared*

Komponen yang dapat menerima cahaya disebut juga photodetektor, merupakan komponen yang peka terhadap cahaya yang dapat berupa *dioda* (*photodiode*) atau *transistor* (*phototransistor*). Komponen ini mengubah energi cahaya dalam hal ini energi cahaya inframerah menjadi pulsa listrik. Komponen harus mampu mengumpulkan sinyal inframerah sebanyak-banyaknya sehingga sinyal pulsa listrik yang dihasilkan kualitasnya cukup baik. Semakin besar intensitas inframerah yang diterima, maka sinyal pulsa listrik yang dihasilkan akan baik. Jika sinyal inframerah yang diterima intensitasnya lemah, maka receiver inframerah tersebut harus mempunyai pengumpul cahaya (*light collector*) yang cukup baik dan sinyal pulsa listrik yang dihasilkan harus dikuatkan.

Konfigurasi photodetektor yang umum dipakai adalah teknik yang dikenal dengan "*Reserved Bias*" atau "*Photoconductive Mode*". Pada mode *reserved bias*, photodetektor diberi tegangan eksternal mulai dari beberapa volt sampai 50 volt (tergantung karakteristiknya). Ketika photodetektor mendapatkan cahaya (dalam hal ini cahaya inframerah), terdapat arus bocor yang relatif kecil. Besar kecilnya arus bocor ini tergantung dari intensitas cahaya inframerah yang mengenai photodetektor tersebut.

Sebuah *photodiode*, biasanya mempunyai kerakter yang lebih baik daripada *phototransistor* dalam hal respons terhadap cahaya inframerah. *Photodiode biasanya* mempunyai respons 100x lebih cepat daripada *phototransistor*. Oleh sebab itu, para *designer* cenderung menggunakan

photodiode daripada *phototransistor*. Tetapi, sebuah *phototransistor* tetap punya keunggulan yaitu mempunyai kemampuan untuk menguatkan arus bocor menjadi ratusan kali jika dibandingkan dengan *photodiode*.

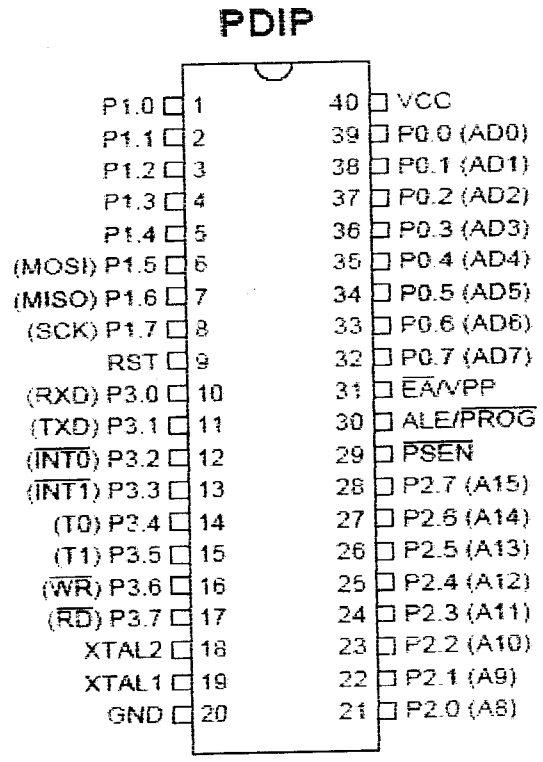
Sebuah photodetektor biasanya dilapisi dengan plastik transparan yang juga berfungsi sebagai lensa *fresnel*. Lensa ini merupakan lensa cembung yang punya sifat mengumpulkan cahaya. Lensa tersebut juga merupakan filter cahaya, lebih dikenal sebagai "*optical filter*" yang hanya melewatkan cahaya inframerah saja. Walaupun demikian, cahaya yang tampak oleh mata tetap masih bisa mengganggu kerja dari photodetektor karena tidak semua cahaya tampak bisa di *filter* dengan baik. Oleh karena itu, sebuah penerima inframerah harus punya *filter* kedua, yaitu rangkaian *filter* yang berfungsi untuk memfilter sinyal 30KHz sampai 40KHz saja.

Faktor lain yang juga berpengaruh pada kemampuan penerimaan inframerah adalah "*active area*" dan "*respond time*". Semakin besar area terima suatu photodetektor, akan semakin besar pula intensitas cahaya yang dapat dikumpulkan, sehingga arus bocor yang diharapkan dalam teknik *reserved bias* semakin besar. Selain itu, semakin besar area penerimaan maka sudut penerimaan juga semakin besar. Kelemahan area penerimaan yang semakin basar adalah *noise* yang dihasilkan juga semakin besar, begitu halnya dengan respons terhadap frekuensi. Semakin besar area penerimaan, respons terhadap frekuensi akan semakin menurun dan sebaliknya.

Respond time dari suatu photodetektor inframerah mempunyai waktu respons yang biasanya dalam satuan *nanoseconds*. *Respond time*

2.3 Mikrokontroler AT89C51

AT89C51 adalah mikrokontroler keluaran Atmel dengan 4K *byte Flash* PEROM (*Programmable and Eraseable Read Only Memory*). AT89C51 mempuyai memori dengan teknologi *nonvolatile memory*, isi memori tersebut dapat diisi ulang ataupun dihapus berkali-kali. Memori ini biasa digunakan untuk menyimpan instruksi (perintah) berstandar MCS-51 *code* sehingga memungkinkan mikrokontroler ini untuk bekerja dalam mode *single chip operation* (mode operasi keping tunggal) yang tidak memerlukan *external memory* (memori luar) untuk menyimpan *source code* tersebut. Gambar 2.4 merupakan konfigurasi pin mikrokontroler AT89C51.



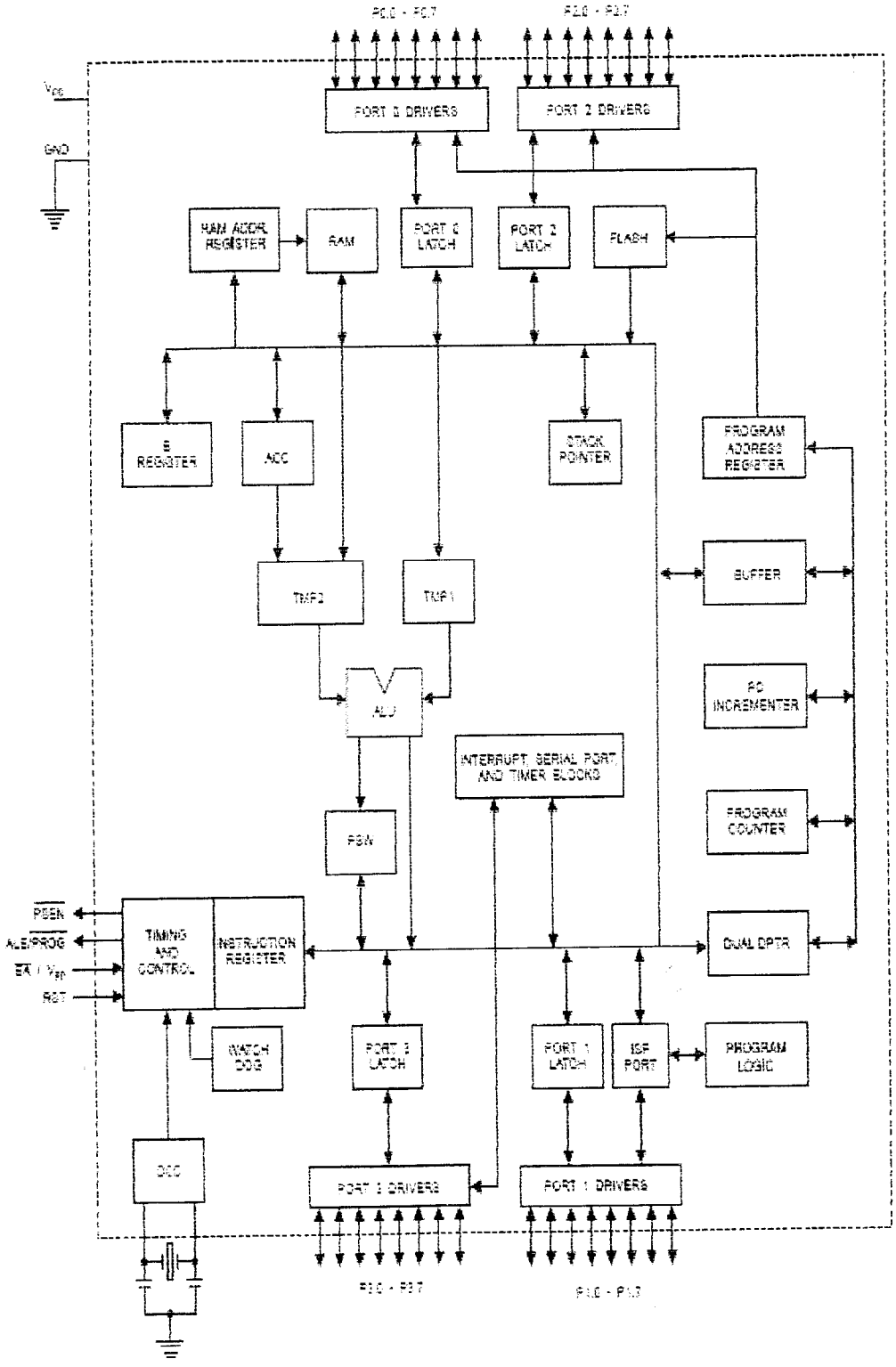
Gambar 2.4 Konfigurasi Pin AT89C51

Mikrokontroler AT89C51 terdiri dari beberapa bagian yang berfungsi untuk mendukung pengendaliannya, bagian-bagiannya adalah sebagai berikut :

1. Kapasitas memori internal 4 *Kbyte* (*Flash 4 Kbytes*)
2. 8 x 128 *byte* RAM (*Random Acces Memory*)
3. 32 jalur I/O yang dapat diprogram
4. 2 buah (6-bit pewaktu/pencacah (*timer/counter*)
5. *Serial Port Full Dupleks*
6. *Chip oscillator* dan *clock*
7. 6 buah sumber interupsi
8. Kompatibel dengan semua produk MCS 51

Diagram blok arsitektur AT89C51 dapat dilihat pada gambar 2.5. Pada diagram blok tersebut dapat disimpulkan bahwa untuk aplikasi yang tidak membutuhkan adanya RAM dan atau ROM dengan skala besar, maka AT89C51 telah dapat dipergunakan dalam konfigurasi *single chip*.

Fasilitas *Parallel Port* yang dimiliki dapat dipergunakan untuk mengendalikan peralatan eksternal atau memasukkan data yang diperlukan. *Port Serial* dapat dipergunakan untuk mengakses sistem komunikasi data dengan dunia luar. *Timer/counter* yang ada dapat dipergunakan untuk mencacah pulsa, menghitung lama pulsa atau sebagai pewaktu umum. Sedangkan sistem *interrupt* membuat AT89C51 dapat dipakai pada aplikasi-aplikasi yang mendekati sistem dengan proses *real-time*.



Gambar 2. 5 Diagram Blok Arsitektur AT89C51

Deskripsi Pin AT89C51 :

1. Vcc (pin 40)

   Suplai tegangan

2. *Ground* (pin 20)

   Pentanahan

3. *Port* 0 (pin 32 – 39)

   Port 0 dapat berfungsi sebagai I/O biasa, dan dapat menerima kode *byte* pada saat *Flash Programming*. Sebagai *port output*, *port* ini dapat memberikan *output sink* ke delapan buah TTL *input*. Selain itu, *port* juga dapat difungsikan sebagai *input* dengan memberikan logika 1 pada *port* tersebut. *Port* 0 yang merupakan saluran I/O 8 bit *open collector* dapat juga digunakan sebagai multipleks bus alamat rendah dan bus data selama adanya akses ke memori program eksternal. Saat proses pemrograman dan verifikasi, *Port* 0 digunakan sebagai saluran data. *Pull up* eksternal diperlukan selama proses verifikasi.

4. *Port* 1 (pin 1 – 8)

   Port 1 berfungsi sebagai I/O biasa atau menerima *low order address bytes* selama *Flash Programming*. *Port* ini memiliki *internal pull up* dan dapat berfungsi sebagai *input* dengan memberikan logika 1. Sehingga *output port* ini dapat memberikan *output sink* ke empat buah input TTL. *Port* ini juga dapat digunakan sebagai saluran alamat saat pemrograman

dan verifikasi. *Port* 1 dipetakan pada alamat 90H dan dapat berfungsi sebagai berikut.

Tabel 2.1 Fungsi Alternatif Port 1 AT89C51

Pin *Port*	**Fungsi Alternatif**
P1.0	TI2 (*Timer/Counter 2 eksternal input*)
P1.1	TO2 (*Timer Counter 2 eksternal output*)
P1.2	T2EA (*Timer/Counter 2 Capsture Reload Trigger*)
P1.3	SSI (*Slave Port Slave Input*)
P1.4	SSO (*Slave Port Slave Output*)
P1.5	MOSI (*Master Data Output, slave data input pin* untuk SPI)
P1.6	MISO (*Master Data Input, slave data output pin* untuk SPI)
P1.7	SCK (*Master Clock Input, slave data input pin* untuk SPI)

5. *Port* 2 (Pin 21 – 28)

Port 2 berfungsi sebagai I/O biasa atau *high order address*, saat mengakses memori secara 16 bit (Movx @DPTR). Sebagai *output*, *port* ini dapat memberikan *output sink* pada ke empat buah input TTL, sedangkan untuk memfungsikan sebagai *port input* dilakukan dengan memberikan logika 1. *Port* ini memiliki *internal pull up.*

6. *Port* 3 (pin 10 – 17)

Memiliki sifat yang sama dengan *Port* 1 dan *Port* 2 yaitu sebagai *port* I/O 8 bit dengan *internal pull up* yang memiliki fungsi pengganti. Bila fungsi pengganti tidak dipakai maka dapat digunakan sebagai *port* paralel 8 bit serba guna. Selain itu, sebagian *Port* 3 dapat berfungsi

sebagai sinyal kontrol saat proses pemrograman dan verifikasi. Adapun fungsi dari *Port* 3 adalah sebagai berikut.

Tabel 2.2 Fungsi Alternatif *Port* 3 AT89C51

Pin *Port*	Fungsi Alternatif
P3.0	RDX (*Port* untuk masukan serial)
P3.1	TDX (*Port* untuk keluaran serial)
P3.2	INT0 (untuk melayani interupsi eksternal 0)
P3.3	INT1 (untuk melayani interupsi eksternal 1)
P3.4	T0 (untuk masukan *eksternal timer 1*)
P3.5	T1 (untuk masukan *eksternal timer 0*)
P3.6	WR (*Eksternal Data Memory Write Strobe*)
P3.7	RD (*Eksternal Data Memory Read Strobe*)

7. *Reset* (Pin 9)

   Reset input. Reset akan aktif dengan memberikan *input high* selama 2 *cycle.* Pulsa transisi dari rendah ke tinggi akan me*reset* mikrokontroler.

8. ALE (Pin 30)

   Pin ini berfungsi sebagai *Address Latch Enable* (ALE) yang me-*latch low byte address* pada saat mengakses memori eksternal. ALE hanya akan aktif saat mengakses memori eksternal (movx dan movc).

9. PSEN (Pin 29)

   Program Store Enable (PSEN), pin ini berfungsi pada saat mengakses program yang terletak pada memori eksternal.

10. EA (Pin 31)

    External Access Enable (EA) merupakan sinyal kontrol untuk pembacaan memori program. Pada kondisi *low* pin ini akan berfungsi

sebagai EA yaitu mikrokontroler akan menjalankan program yang ada pada memori eksternal setelah sistem di*reset*.

Sedangkan jika berkondisi *high*, pin akan berfungsi menjalankan program yang ada pada memori internal pada saat *Flash Programming*, pin ini akan mendapat tegangan 12 volt.

11. XTAL1 (Pin 19)

Input inverting osilator.

12. XTAL2 (Pin18)

Output inverting osilator.

2.3.1 Struktur Memori

AT89C51 mempunyai struktur memori yang terdiri atas :

a. *RAM Internal*, memori sebesar 128 *byte* yang biasnya digunakan untuk menyimpan variabel atau data yang bersifat sementara.

b. *Special Function Register* (*Register* Fungsi Khusus), memori yang berisi *register-register* yang mempunyai fungsi-fungsi khusus yang disediakan oleh mikrokontroler tersebut, seperti *timer*, *serial* dan lain-lain.

c. *Flash PEROM*, memori yang digunakan untuk menyimpan instruksi-instruksi MCS-51.

AT89C51 mempunyai struktur memori yang terpisah antara *RAM Internal* dan *Flash PEROM*-nya. *RAM Internal* dialamati oleh *RAM Address Register* (*Register* Alamat RAM) sedangkan *Flash PEROM* yang menyimpan perintah-

perintah MCS-51 dialamati oleh *Program Address Register* (*Register* Alamat *Program*). Dengan adanya struktur memori yang terpisah tersebut, walaupun *RAM Internal* dan *Flash PEROM* mempunyai alamat awal yang sama, yaitu alamat 00H, namun secara fisiknya kedua memori tersebut tidak saling berhubungan.

2.3.1.1 *RAM Internal*

RAM Internal terdiri atas :

a. *Register Banks*

AT89C51 mempunyai delapan buah *register* yang terdiri atas R0 hingga R7. Kedelapan buah register ini selalu terletak pada alamat 00H hingga 07H pada setiap kali sistem di*reset*. Namun, posisi R0 hingga R7 dapat dipindah ke *Bank* 1 (08H hingga 0FH), *Bank* 2 (10H hingga 17H) atau *Bank* 3 (18H hingga 1FH) dengan mengatur bit RS0 dan RS1.

b. *Bit Addressable RAM*

RAM pada alamat 20H hingga 2FH dapat diakses secara pengalamatan bit (*bit addressable*) sehingga hanya dengan sebuah instruksi saja bit dalam area ini dapat di*set*, *clear*, *AND* dan *OR*. Sebagai contoh, pada saat terjadi instruksi Setb 67H, hal ini sama dengan men*set* bit MSB dari alamat 2C.

Dengan adanya sistem *bit addressable RAM*, proses yang seharusnya dijalankan dengan tiga *cycle* dapat digantikan dengan sebuah instruksi yang hanya membutuhkan satu *cycle* saja.

c. RAM Keperluan Umum

RAM Keperluan Umum dimulai dari alamat 30H hingga 7FH dan dapat langsung diakses dengan pengalamatan langsung maupun tak langsung.

2.3.1.2 *Register* Fungsi Khusus

AT89C51 mempunyai 21 *Special Function Register* (*Register* Fungsi Khusus) yang terletak pada antara alamat 80H hingga FFH. Beberapa dari *register-register* ini juga mampu dialamati secara pengalamatan bit sehingga dapat dioperasikan seperti RAM yang lokasinya dapat dialamati dengan pengalamatan bit.

a. Accumulator

Register ini terletak pada alamat E0H. Hampir semua operasi aritmatik dan operasi logika selalu menggunakan *register* ini. Untuk proses pengambilan dan pengiriman data dari memori eksternal juga diperlukan *register* ini.

b. Port

AT89C51 mempunyai empat buah *port*, yaitu *Port* 0, *Port* 1, *Port* 2 dan *Port* 3 yang terletak pada alamat 80H, 90H, A0H dan B0H. Namun, jika digunakan *eksternal memory* ataupun fungsi-fungsi

spesial, seperti *External Interrupt*, *Serial* maupun *External Timer*, *Port* 0, *Port* 2 dan *Port* 3 tidak dapat digunakan sebagai *port* dengan fungsi umum. Untuk itu disediakan *Port* 1 yang dikhususkan untuk *port* dengan fungsi umum. Semua *port* ini dapat diakses dengan pengalamatan secara bit sehingga dapat dilakukan perubahan *output* pada tiap-tiap *pin* dari *port* ini tanpa mempengaruhi *pin-pin* yang lainnya.

c. *Program Status Word*

Program Status Word atau PSW terletak pada alamat D0H yang terdiri atas beberapa bit sebagai berikut :

PSW.7	PSW.6	PSW.5	PSW.4	PSW.3	PSW.2	PSW.1	PSW.0
CY	AC	F0	RS1	RS0	OV	-	P

Gambar 2.6 PSW

1. *Flag Carry*

Flag Carry (terletak pada alamat D7H) mempunyai fungsi sebagai pendeteksi terjadinya kelebihan pada operasi penjumlahan atau terjadi pinjam (*borrow*) pada operasi pengurangan. Selain pada operasi aritmatik, *carry* juga berfungsi sebagai tempat penyimpanan sebesar 1 bit pada operasi *boolean*

2. *Flag Auxiliary Carry*

Bit ini menunjukkan adanya *carry* dari bit ketiga menuju bit keempat pada operasi aritmatika atau dari 4 bit terendah ke 4 bit

g. *Register Timer*

AT89C51 mempunyai dua buah 16 bit *Timer/Counter*, yaitu *Timer* 0 dan *Timer* 1. *Timer* 0 terletak di alamat 8AH untuk TL0 dan 8CH untuk TH0 dan *Timer* 1 terletak di alamat 8BH untuk TL1 dan 8DH untuk TH1.

h. *Register Port Serial*

AT89C51 mempunyai sebuah *on chip serial port* (*Port Serial* di dalam keping) yang dapat digunakan untuk berkomunikasi dengan peralatan lain yang menggunakan *port* juga seperti *modem*, *shift register* dan lain-lain.

Buffer (penyangga) untuk proses pengiriman maupun pengambilan data terletak pada *Register* SBUF, yaitu pada alamat 99H. Sedangkan untuk mengatur mode serial dapat dilakukan dengan mengubah isi dari SCON yang terletak pada alamat 98H.

i. *Register Interupt*

AT89C51 mempunyai lima buah interupsi dengan dua *level* prioritas interupsi. Interupsi akan selalu nonaktif setiap kali sistem di*reset*. *Register-register* yang berhubungan dengan *interrupt* adalah *Interrupt Enable Register* (IE) atau *Register* Pengaktif Interupsi pada alamat A8H untuk mengatur keaktifan tiap-tiap *interrupt* dan *Interrupt Priority Register* (IP) atau *Register* Prioritas Interupsi pada alamat 0B8H.

j. *Register Control Power*

Register ini terdiri atas SMOD yang digunakan untuk melipat dua *baud rate* dari *port serial,* dua buah bit untuk *flag* fungsi umum pada bit ketiga dan bit kedua, *Power Down* (PD) *bit* dan *Idle* (IDL) *bit.*

Pada mode *Idle* hubungan antara CPU dan *internal clock* terputus, namun kondisi *port* tetap pada kondisi terakhir, ALE dan PSEN menjadi *high, timer* masih tetap bekerja. Mode *Idle* berakhir pada saat terjadi *interupt, reset* atau kondisi-kondisi lain yang me*reset* IDL *bit.* Pada mode *Power Down oscillator* dan semua fungsi berhenti, RAM tetap pada kondisi terakhir, begitu pula dengan *port* dan ALE maupun PSEN akan berkondisi 0. Mode *Power Down* akan berakhir pada saat terjadi *reset.*

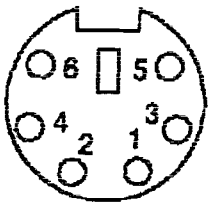
2.3.1.3 *Flash* PEROM

AT89C51 mempunyai 4 Kb *Flash* PEROM (*Programmable and Eraseable Read Only Memory*), yaitu ROM yang dapat ditulis ulang atau dihapus menggunakan sebuah perangkat *programmer. Flash* PEROM dalam AT89C51 menggunakan *Atmel's High-Density Non Volatile Technology* yang mempunyai kemampuan untuk ditulis ulang hingga 1000 kali dan berisikan standard MCS51

Program yang ada pada *Flash* PEROM akan dijalankan jika pada saat sistem di*reset, pin* EA/VP berlogika satu sehingga mikrokontroler aktif berdasarkan program yang ada pada *Flash* PEROM-nya

2.4 PS/2 *Mouse Port*

PS/2 *mouse port* pada dasarnya adalah sama dengan *serial port* jika dilihat dari fungsinya untuk mengirimkan dan menerima data secara *serial*. Jenis komunikasi *serial* yang digunakan pada PS/2 *mouse port* adalah secara *synchronous*, yaitu dengan menggunakan bantuan pulsa *clock*. Pada proses pengiriman dan penerimaan data, sinyal *clock* diperlukan oleh peralatan penerima data untuk mengetahui adanya pengiriman setiap bit data.



Ket :
1. Data
2. NC
3. Gnd
4. +5 VDc
5. Clock
6. NC

Gambar 2.7 Susunan *Pin* PS/2 *Port*

Sinyal *clock* tersulut pada saat pengiriman bit yang pertama dan setiap perubahan bit data. Peralatan atau komponen penerima akan mengetahui adanya pengiriman bit yang pertama ataupun perubahan bit data dengan mendeteksi sinyal *clock*.

Setiap terjadi perubahan keadaan pada alat (pergeseran atau penekanan tombol), *device* akan mengirimkan informasi ke PS/2 *mouse port* berupa 3 *byte* data. Isi paket data yang dikirimkan adalah dijelaskan pada diagram dibawah ini.

	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Byte 1	Y overflow	X overflow	Y sign bit	X sign bit	Always 1	Middle Btn	Right Btn	Left Btn
Byte 2	X Movement							
Byte 3	Y Movement							

Gambar 2.8 *Packet Data Format*

Byte pertama berisi informasi *status left button*, *status middle button*, *status right button*, arah perubahan gerak dibandingkan dengan posisi sebelumnya, dan penunjuk X,Y *overflows* jika *counter* melebihi 255. *Byte* ke 2 berisi besarnya perubahan pergerakan pada sumbu X sedangkan *byte* ke 3 berisi informasi besarnya perubahan pergerakan pada sumbu Y. Besarnya pergerakan pada sumbu X danY tergantung pada besarnya nilai *counter* yang dihasilkan.

2.4.1 Komunikasi Data

PS/2 *port* menggunakan *bidirectional synchronous serial protocol*. *Device* selalu menghasilkan sinyal *clock*. Jika komputer akan mengirimkan data, maka pertama kali yang harus dilakukan untuk memulai komunikasi adalah dengan membuat jalur *clock* menjadi rendah (*low*). Kemudian komputer akan mengirimkan bit yang pertama dari *byte* data yang akan dikirimkan. Bit pertama berupa *start bit* yang berlogika rendah (*low*). Pada saat komputer mengirimkan *start bit*, jalur *clock* diubah lagi menjadi *high*. Proses ini dinamakan dengan "*request-to-send*" yang akan ditanggapi oleh *device* dengan memulai menghasilkan pulsa *clock*.

Kondisi *bus* PS/2 *port*:

Data = high, Clock = high : *Idle state.*

Data = high, Clock = low : *Communication Inhibited.*

Data = low, Clock = high :*Host Request-to-Send*

Semua data dikirimkan dalam 1 *byte* per sekali pengiriman dan setiap *byte* data terdiri dari :

a. 1 *start bit*. Selalu 0.

b. 8 *data bits*, *least significant bit* (LSB) dikirim pertama.

c. 1 *parity bit* (*odd parity*).

d. 1 *stop bit*. Selalu bernilai 1.

e. 1 *acknowledge bit* (hanya pada *host-to-device communication*)

Bit parity akan *set* (*high*) jika bit yang bernilai 1 berjumlah genap dan *bit parity* akan bernilai 0 jika bit yang bernilai 1 berjumlah ganjil. *Bit parity* berfungsi untuk mendeteksi *error* pada saat proses pengiriman data.

Data yang dikirimkan dari *device* ke *host* akan dibaca oleh *host* ketika terjadi perubahan pulsa *clock* dari tinggi ke rendah dan sebaliknya, data yang dikirimkan dari *host* ke *device* akan dibaca oleh *device* ketika terjadi perubahan pulsa *clock* dari rendah ke tinggi.

2.4.1.1 Komunikasi : *Device-to-Host*

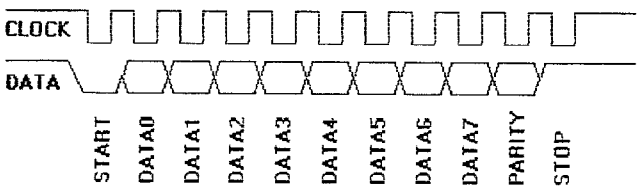
Pada saat *device* akan mengirimkan data, terlebih dahulu *device* akan memastikan jalur *clock* harus berada pada kondisi logika tinggi (*high*). Jika jalur *clock* tidak dalam kondisi *high*, kemungkinan *host* sedang melakukan komunikasi, sehingga *device* harus menunggu mengirimkan datanya sampai *host* melepaskan jalur *clock* menjadi *high* kembali. Jalur *clock* harus

dipastikan dalam kondisi *high* selama minimal 50 *microseconds* sebelum *device* dapat memulai mengirimkan datanya.

Jumlah bit yang dikirimkan dalam satu paket *frame* ada 11 bit, yaitu :

a. 1 *start bit*. Selalu 0.
b. 8 *data bits*, *least significant bit* (LSB) dikirim pertama.
c. 1 *parity bit* (*odd parity*).
d. 1 *stop bit*. Selalu bernilai 1.

Device mengirimkan bit pada jalur data saat jalur *clock high*, dan akan dibaca oleh *host* pada saat jalur *clock low*.



Gambar 2.9 *Timing Clock* Pengiriman Data *Device-to-Host*

Frekuensi *clock* yang digunakan 10Khz–16.7Khz. Jarak waktu antara perubahan *clock* menjadi *high* dengan perubahan bit pada data minimal 5 *microseconds*. Sedangkan jarak waktu antara perubahan bit pada data dengan perubahan *clock* menjadi *low* minimal 5 *microseconds* dan tidak boleh lebih dari 25 *microseconds*.

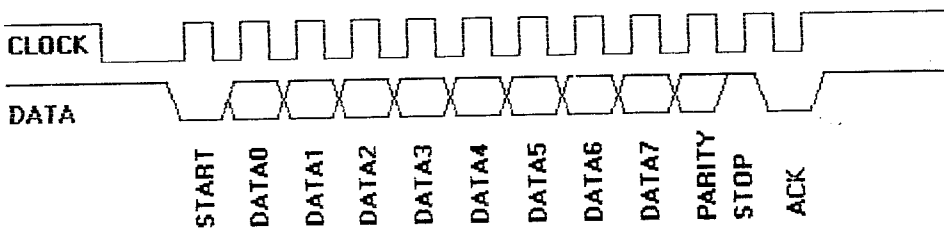
2.4.1.2 Komunikasi : *Host-to-Device*

Paket data dikirimkan sedikit berbeda pada *host-to-device communication*. Jika *host* akan mengirimkan data menuju *device* yang harus dilakukan pertama kali yaitu mengubah kondisi jalur *clock* dan jalur data pada keadaan "*Requet-to-send*" seperti berikut :

a. Mengubah jalur *clock* menjadi *low* selama minimal 100 *microseconds.*

b. Memulai proses "*Request-to-send*" dengan mengubah jalur data menjadi *low*, kemudian diikuti dengan melepaskan jalur *clock* menjadi *high* kembali.

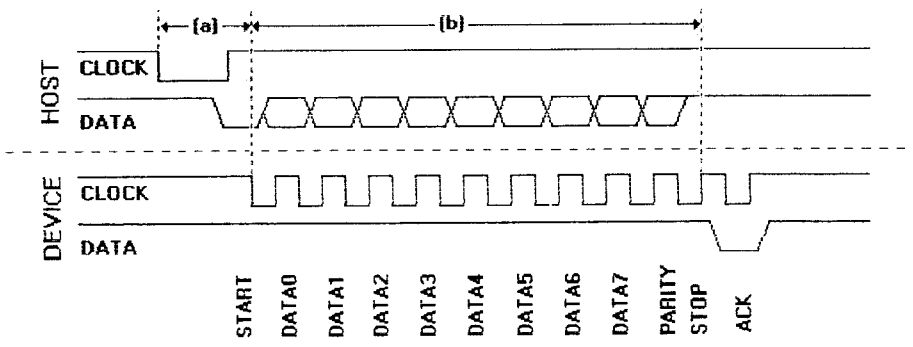
Device harus selalu mengecek kondisi ini dengan interval tidak boleh melebihi 10 *miliseconds*. Ketika kondisi ini terdeteksi oleh *device*, kemudian *device* akan menanggapinya dengan memulai menghasilkan sinyal *clock*. *Host* mengirimkan bit pada jalur data saat jalur *clock low*, dan akan dibaca oleh *device* pada saat jalur *clock high*. Setelah *stop bit* diterima, *device* kemudian megirimkan *bit acknowledge* dengan cara membuat jalur data menjadi *low* disertai dengan menghasilkan pulsa *clock* yang terakhir.



Gambar 2.10 *Timing Clock* Pengiriman Data *Host-to-Device*

Untuk lebih mudahnya, proses pengiriman data dari *host-to-device* adalah sebagai berikut :

1. Ubah jalur *clock* menjadi *low* selama minimal 100 *microseconds*
2. Ubah jalur data menjadi *low*
3. Ubah jalur *clock* menjadi *high* kembali
4. Tunggu hingga *device* mengubah jalur *clock* menjadi *low*
5. Memulai proses pengiriman bit data yang pertama (LSB)
6. Tunggu hingga *device* mengubah jalur *clock* menjadi *high*
7. Tunggu hingga *device* mengubah jalur *clock* menjadi *low*
8. Ulangi langkah 5-7 untuk tujuh bit data yang lainnya dan sebuah *bit parity*
9. Ubah jalur *clock* menjadi *high*
10. Tunggu hingga *device* megubah jalur data menjadi *low*
11. Tunggu hingga *device* mengubah jalur clock menjadi *low*
12. Tunggu hingga *device* mengubah jalur data dan *clock* jadi *high* kembali



Gambar 2.11 Proses Pengiriman Data oleh *Host* dan Pembacaan oleh *Device*

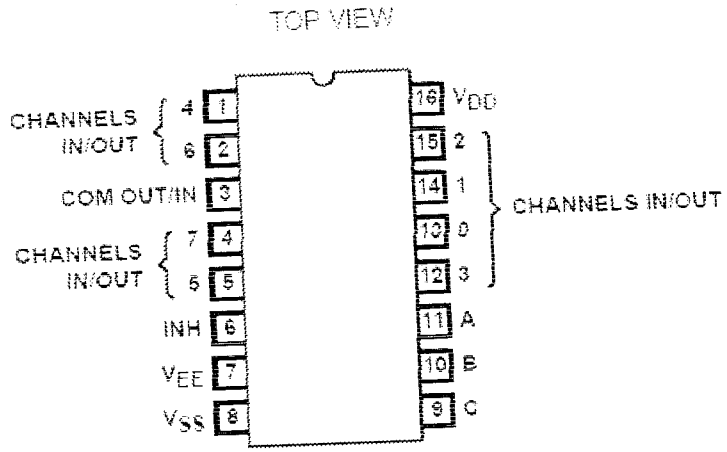
2.4.2 Inisialisasi

Pada saat PS/2 *device* dinyalakan, *device* akan mengirimkan dua *byte* data yaitu AAh dan 00h ke PS/2 *mouse port*. AAh mengindikasikan *self-test-passed* dan 00 merupakan ID *device*. Ketika kedua data ini diterima oleh komputer dan dicek, kemudian komputer menyuruh *device* untuk *reset* dan komputer akan menunggu sampai *device* mengirimkan sinyal *acknowledge* (FA). Setelah itu komputer akan menentukan variabel *scaling factor, resolution*, dan lain-lain dengan cara mengirimkan data tersebut ke *device*. Pada setiap proses pengiriman data ke *device*, komputer akan mengecek sinyal *acknowledge* yang dikerimkan oleh *device* ke komputer, menandakan data telah diterima oleh *device* dan komputer tidak akan mengulang pengiriman data itu kembali. Setelah semua pengiriman variabel telah dilakukan, komputer mengirimkan sinyal *enable* ke *device* menandakan *device* siap digunakan.

2.5 Multiplexer/Demultiplexer

Multiplexer mempunyai sifat yang berkebalikan dengan *demultiplexer*. *Multiplexer* adalah sebuah perangkat yang digunakan untuk memilih satu dari beberapa masukan sumber data, dan kemudian mengeluarkannya ke dalam satu buah jalur keluaran. *Demultiplexer* adalah sebuah perangkat yang digunakan untuk memilih satu jalur keluaran dari beberapa jalur keluaran yang tersedia, dan kemudian menghubungkan jalur keluaran terpilih tersebut ke sebuah jalur masukan. *Multiplexer* dan *demultiplexer* yang digunakan adalah IC CD4051. Jika difungsikan sebagai *multiplexer*, CD4051 mempunyai 8 jalur keluaran dan 1 jalur

masukan. Sedangkan jika digunakan sebagai *demultiplexer*, CD4051 mempunyai 1 jalur keluaran dan 8 jalur masukan. Gambar 2.12 menerangkan konfigurasi pin IC CD4051.



Gambar 2.12 Konfigurasi Pin IC CD4051

Pemilihan jalur dapat dilakukan dengan mengeset bit selector pada pin A, B, dan C. Tabel kebenaran untuk memilih jalur masukan atau jalur keluaran yang digunakan dapat dilihat pada tabel di bawah ini.

Tabel 2.3 Tabel Kebenaran IC CD4051

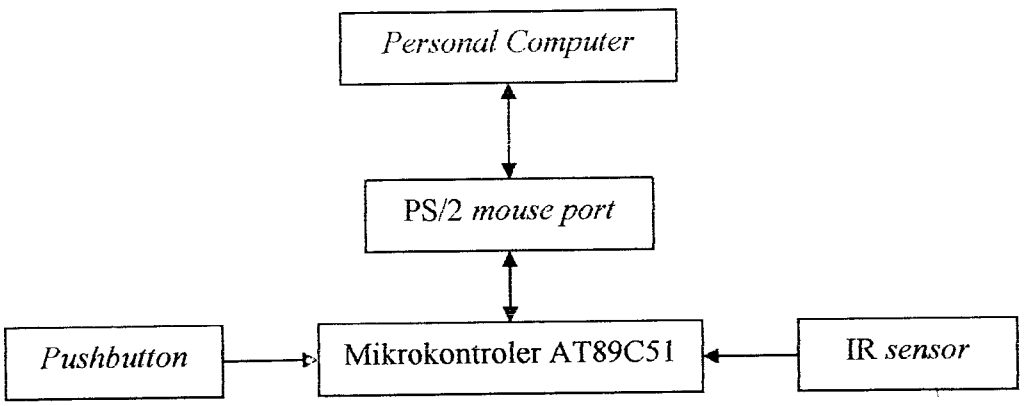
INPUT STATES				"ON" CHANNEL(S)
INHIBIT	C	B	A	
CD4051B				
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	X	X	X	None

BAB III

PERANCANGAN SISTEM

3.1 Perancangan Sistem

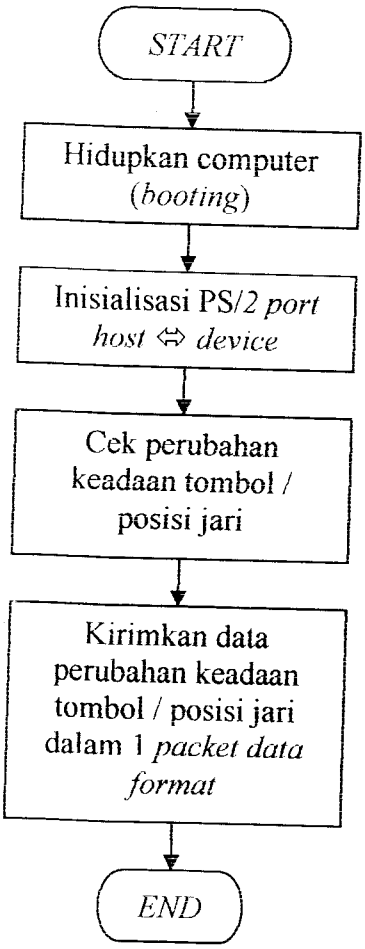
Touchpad merupakan *pointer device* yang dapat berfungsi layaknya sebuah *mouse* biasa. Secara keseluruhan, perancangan sistem *touchpad* terdiri dari perangkat keras dan perangkat lunak. Diagram blok sistem yang dirancang diperlihatkan seperti pada gambar 3.1.



Gambar 3.1 Blok Diagram Rangkaian *Touchpad*

Touchpad berkomunikasi dengan komputer melalui jalur PS/2 *port*. Komputer (*host*) akan melakukan inisialisasi terlebih dahulu sebelum *touchpad* dapat berfungsi. Jalur PS/2 *mouse port* tidak bersifat '*plug and play*', sehingga proses inisialisasi hanya terjadi pada saat komputer *booting*. PS/2 *port* menggunakan *bidirectional synchronous serial protocol*. Komputer akan

mengenali jenis dan tipe *hardware* yang dipasang melalui proses inisialisasi tersebut. Setelah proses inisialisasi dengan komputer dilakukan, *touchpad* mulai melakukan *scanning* IR *sensor* untuk mendeteksi pergerakan jari dan keadaan tombol. Apabila ada pergerakan jari atau tombol yang ditekan, mikrokontroler mengirimkan data tersebut ke komputer melalui jalur PS/2 *mouse* dalam bentuk paket-paket data.



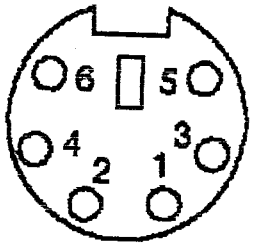
Gambar 3.2 Cara Kerja *Touchpad*

3.2 Perancangan Perangkat Keras

Perancangan perangkat keras sistem *touchpad* memiliki tiga bagian utama, yaitu masukan (*input*), pengendali (*control*), dan keluaran (*output*). Piranti masukannya berupa *sensor* inframerah sebagai pendeteksi jari dan *pushbutton* sebagai tombol seperti pada *mouse*. Piranti kontrolnya adalah mikrokontroler AT89C51, sedangkan untuk keluarannya berupa paket-paket data yang dikirimkan ke PS/2 *mouse port* untuk mengendalikan *cursor*.

3.2.1 Rangkaian Catu Daya

Rangkaian catu daya menyediakan tegangan bagi rangkaian. Tegangan yang digunakan adalah tegangan 0 volt (*ground*) dan +5 volt. Tegangan ini digunakan untuk mencatu daya keseluruhan rangkaian. Untuk menghasilkan tegangan ini dipergunakan catu daya yang berasal dari *port* PS/2. Gambar 3.3 memperlihatkan sumber catu daya dari *port* PS/2

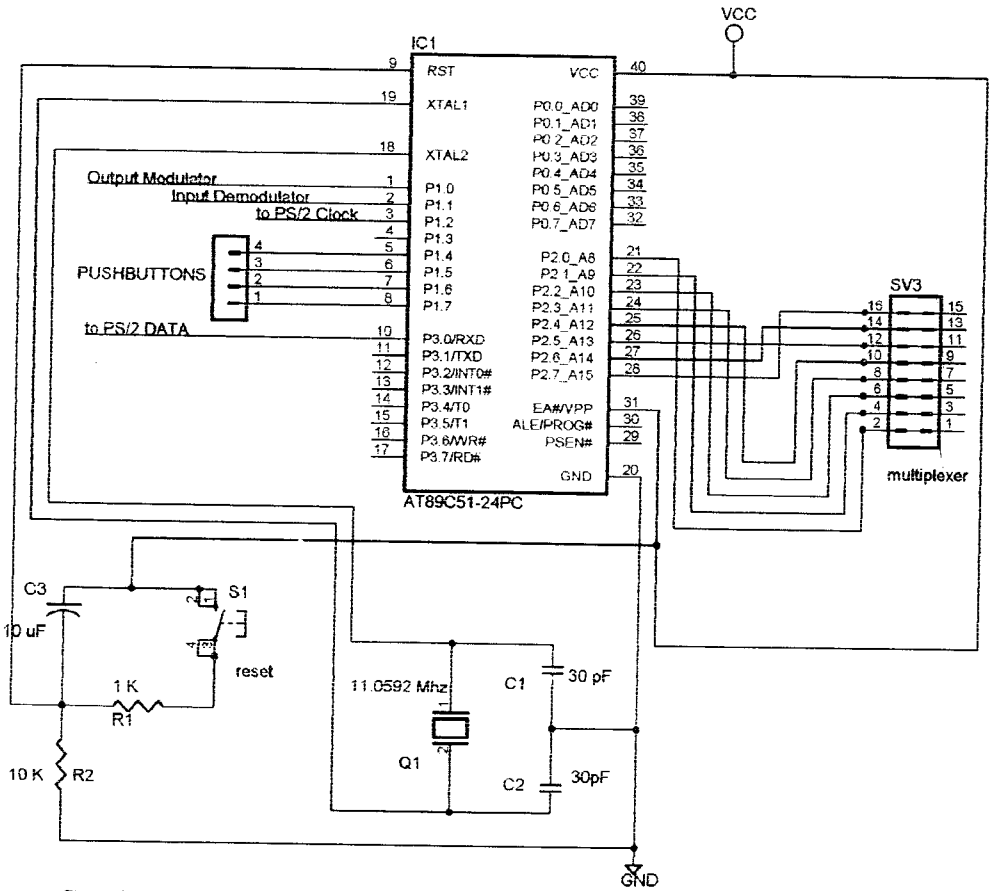


Pin	*Signal*
1	Data
2	NC
3	*Ground*
4	+ 5 V dc
5	*Clock*
6	NC

Gambar 3.3 Susunan Pin PS/2 *Port*

3.2.2 Rangkaian *Modulator* dan Rangkaian Mikrokontroler AT89C51

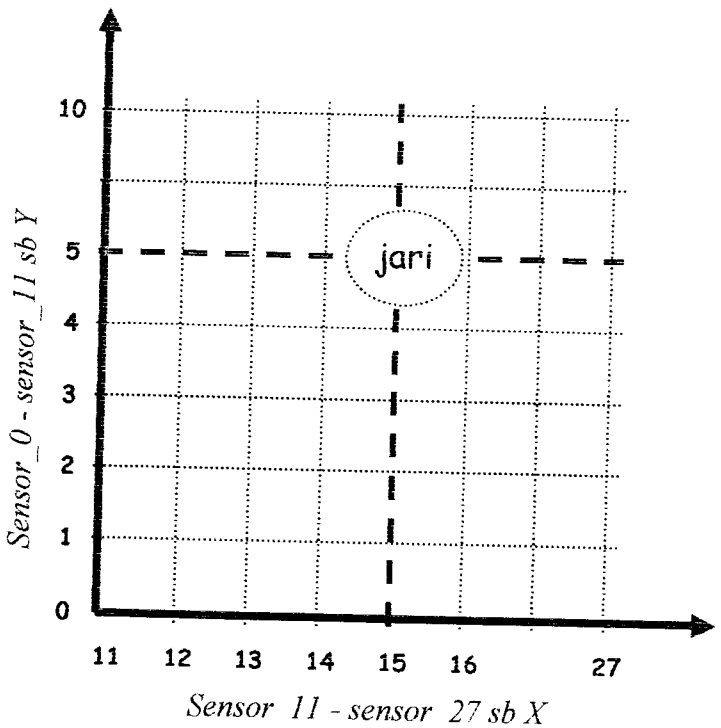
Rangkaian mikrokontroler ini merupakan pusat pengolahan data dan pusat pengendali alat. Di dalam rangkaian mikrokontroler ini terdapat tiga buah *port* yang digunakan untuk menampung *input* atau *output* data dan terhubung langsung oleh rangkaian-rangkaian dari alat pengendali. Rangkaian ini tersusun atas osilator kristal 11.0592 Mhz yang berfungsi untuk membangkitkan pulsa internal dan dua buah kapasitor sebesar 30 pF yang berfungsi untuk menstabilkan frekuensi. Kapasitor 10 uF dan *resistor* 10 KΩ berfungsi untuk rangkaian *reset*. Mikrokontroler digunakan juga sebagai pemodulasi sinyal. Kaki P1.0 sebagai output sinyal termodulasi.



Gambar 3.4 Rangkaian Modulator dan Rangkaian Mikrokontroler

3.2.3 Rangkaian *Multiplexer*

Untuk dapat menentukan letak suatu objek dalam ruangan 2 dimensi secara tepat, dibutuhkan 2 buah sumbu (X & Y). Jumlah *sensor* cahaya yang digunakan dalam satu buah sumbu mempengaruhi tingkat resolusi *cursor*, yaitu banyaknya perpindahan *cursor* dalam satu satuan panjang. Makin banyak jumlah *sensor*, makin kecil jarak perpindahan yang dapat dilakukan, sehingga tingkat akurasi menjadi tinggi. Sedangkan jika jumlah *sensor* yang digunakan semakin sedikit, maka jarak perpindahan yang dilakukan semakin besar dan mengakibatkan tingkat akurasi menjadi rendah.



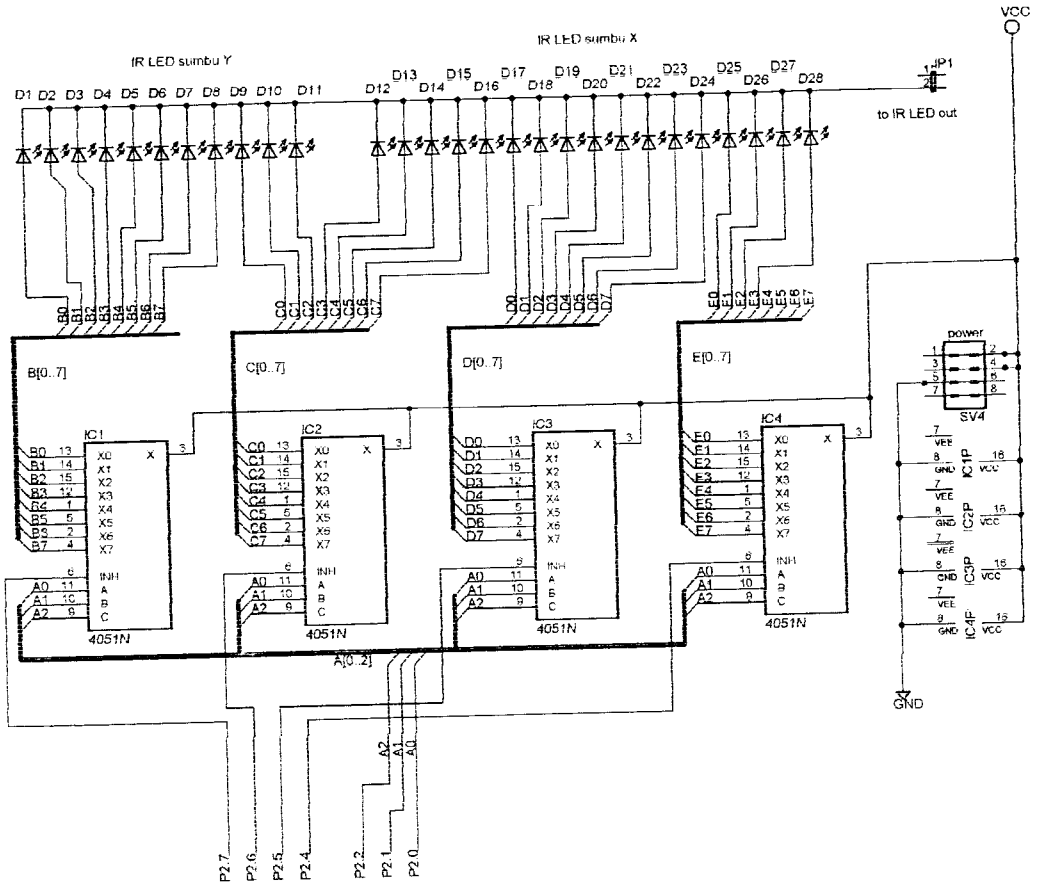
Gambar 3.5 Sistem Koordinat Menggunakan *Infrared*

Rangkaian *multiplexer* berfungsi untuk menghubungkan *sensor* inframerah dengan mikrokontroler. *Multiplexer* dapat untuk mengendalikan *sensor* yang jumlahnya banyak hanya dengan mengatur bit pada *input* selektor A, B, C dan pin *inhibit*. Rangkaian *multiplexer* menggunakan 4 buah buah IC CD4051 (*multiplexer-demultiplexer*), jika satu buah IC dapat dihubungkan dengan 8 buah *sensor*, maka total *sensor* yang bisa dihubungkan berjumlah 4x8 = 32 buah.

3.2.3.1 Rangkaian Multiplexer untuk IR LED

Rangkaian *multiplexer* IR LED berfungsi untuk mengendalikan IR LED yang akan memancarkan sinyal inframerah termodulasi. IR LED yang dipasang berjumlah 17 buah untuk sumbu X dan 11 buah untuk sumbu Y. Total IR LED yang digunakan ada 28 buah, sehingga cukup memakai 4 buah IC CD4051.

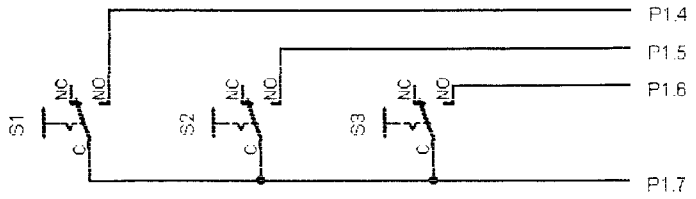
Rangkaian ini akan dihubungkan ke *port* 2 mikrokontroler AT89C51. Mikrokontroler mengendalikan satu-persatu IR LED yang akan aktif dengan mengatur bit selektor dan *inhibit* masing-masing IC CD4051. Rangkaian juga akan terhubung dengan rangkaian modulasi, sehingga cahaya yang dipancarkan oleh IR LED pada rangkaian ini merupakan cahaya inframerah yang telah termodulasi.



Gambar 3.6 Rangkaian *Multiplexer* untuk IR LED

3.2.3.2 Rangkaian *Multiplexer* untuk *Photodiode*

Rangkaian *multiplexer* untuk *photodiode* mempunyai fungsi yang sama dengan rangkaian *multiplexer* untuk IR LED. *Photodiode* yang digunakan juga berjumlah 17 buah untuk sumbu X dan 11 buah untuk sumbu Y. Rangkaian ini dihubungkan dengan *port* 2 mikrokontroler AT89C51 sebagai pengendalinya. Mikrokontroler mengatur *photodiode* yang aktif bergantian satu-persatu. Gambar 3.6 menunjukkan rangkaian *multiplexer* untuk *photodiode*.



Gambar 3.8 Rangkaian *Pushbutton*

Mikrokontroler diisi dengan program scanning *pushbuttons*. Mikrokontroler akan membuat kondisi *low* (0) pada kaki P1.7, selanjutnya mikrokontroler akan membaca P1.4, P1.5 dan P1.6. Apabila hasilnya berupa kondisi *high* (1) menandakan tombol tersebut tidak ditekan, sedangkan apabila hasilnya berupa kondisi *low* (0) menandakan tombol tersebut ditekan. Proses membaca dimulai dari P1.4 kemudian P1.6 terakhir P1.5. Hanya ada satu tombol yang dapat ditekan, jika ada penekanan tombol secara bersamaan mikrokontroler hanya mengambil satu tombol sesuai urutan prioritas masing-masing tombol.

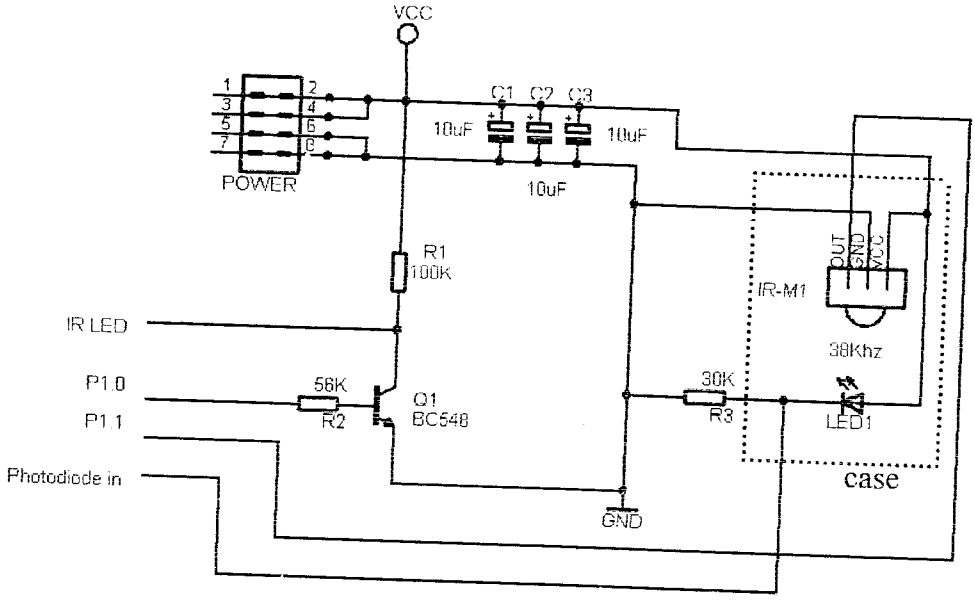
Tabel 3.1 Tabel *Scanning* Tombol

P1.7	P1.6	P1.5	P1.4	Tombol
0	x	x	0	*Left button*
0	0	x	1	*Left button*
0	1	0	1	*Mid button*
0	1	1	1	*none*

3.2.5 Rangkaian *Demodulator*

Modulator inframerah digunakan agar sinyal data yang dipancarkan IR LED lebih kebal terhadap cahaya yang berasal dari lampu dan sinar matahari. Rangkaian *demodulator* berfungsi untuk menerima sinyal termodulasi dan

kemudian memisahkan sinyal data dengan sinyal *carrier*. Rangkaian *demodulator* menggunakan IC IRM-8601 yang dapat menerima sinyal 38 khz.



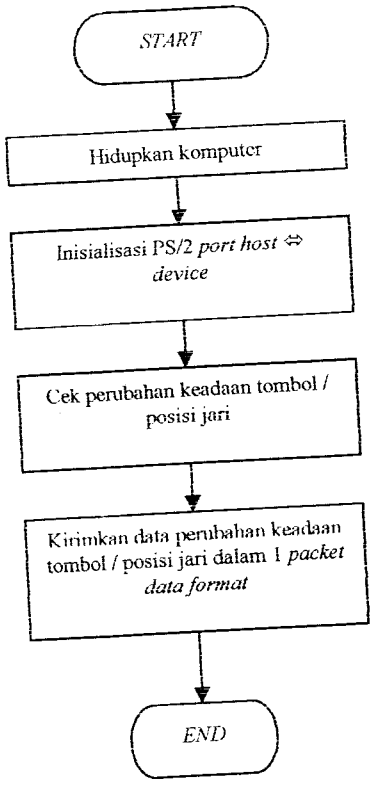
Gambar 3.9 Rangkaian *Demodulator*

Rangkaian *demodulator* dihubungkan ke rangkaian *multiplexer*. Mikrokontroler memodulasi sinyal yang akan dipancarkan oleh IR LED melalui P1.0. Data yang telah termodulasi dipancarkan oleh IR LED kemudian ditangkap oleh *photodiode*. *Photodiode* terhubung ke rangkaian demodulasi menuju LED 1. LED 1 akan menyala atau padam sesuai dengan sinyal termodulasi yang dipancarkan ke *photodiode*. LED 1 dihadapkan langsung ke IC IRM-8601 agar sinyal dapat masuk ke *demodulator*. Sinyal masuk inilah merupakan sinyal termodulasi yang dipancarkan oleh IR LED, kemudian ditangkap oleh *photodiode*, dan kemudian sinyal tersebut dipancarkan kembali ke IC IRM-8601 melalui LED 1.

3.3 Perancangan Perangkat Lunak

Perangkat lunak pada mikrokontroler diperlukan untuk mengeksekusi perintah-perintah pada masukan sehingga menghasilkan keluaran berupa data yang digunakan sebagai pengendali posisi dan keadaan *cursor*. Dalam hal ini mikrokontroler diprogram dengan menggunakan bahasa *assembler*.

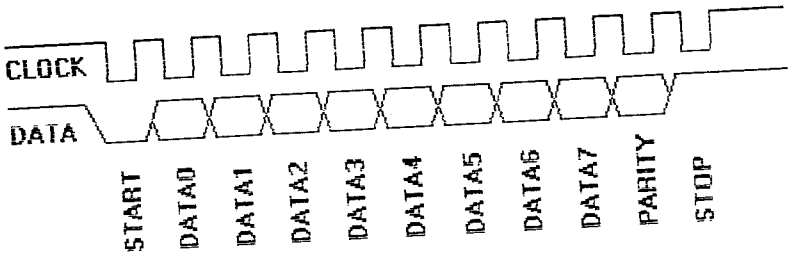
Perancangan program pada mikrokontroler AT89C51 diawali dengan membuat *listing program* pada *text editor* program 8051. *Listing program touchpad* berisi *listing* inisialisasi dengan *port* PS/2, *listing scanning sensor* inframerah dan tombol serta *listing* pengolahan data hasil dari proses *scanning*.



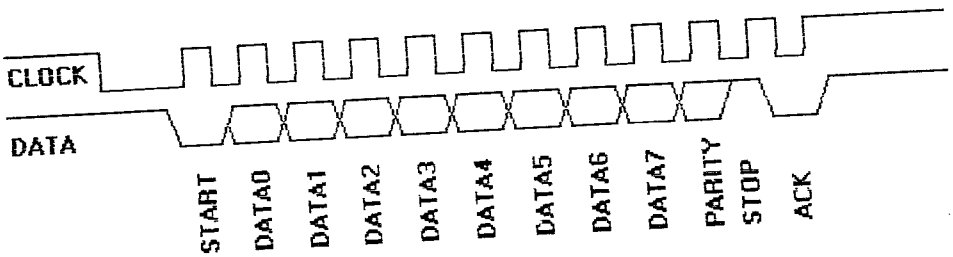
Gambar 3.10 Diagram Alir Cara Kerja *Touchpad*

3.3.1 Program *Bidirectional Synchronous Serial Protocol*

PS/2 *mouse port* berkomunikasi secara *synchronous*, yaitu dengan menggunakan bantuan pulsa *clock*. Pada proses pengiriman dan penerimaan data, sinyal *clock* diperlukan oleh peralatan penerima data untuk mengetahui adanya pengiriman setiap bit data. Sinyal *clock* tersulut pada saat pengiriman bit yang pertama dan setiap perubahan bit data.



Gambar 3.11 *Timing Clock* Pengiriman Data *Device-to-Host*



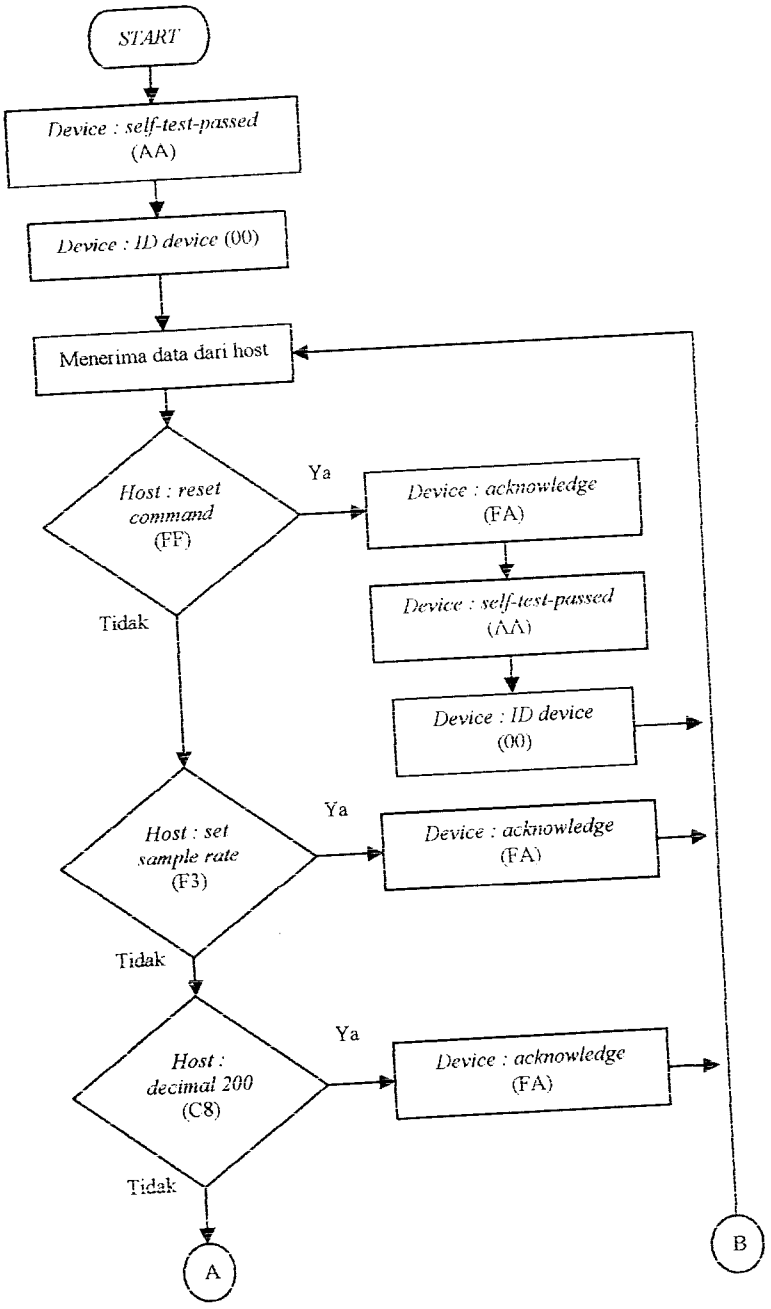
Gambar 3.12 *Timing Clock* Pengiriman Data *Host-to-Device*

Data yang dikirimkan dari *device* ke *host* akan dibaca oleh *host* ketika terjadi perubahan pulsa *clock* dari tinggi ke rendah dan sebaliknya, data yang dikirimkan dari *host* ke *device* akan dibaca oleh *device* ketika terjadi perubahan pulsa *clock* dari rendah ke tinggi.

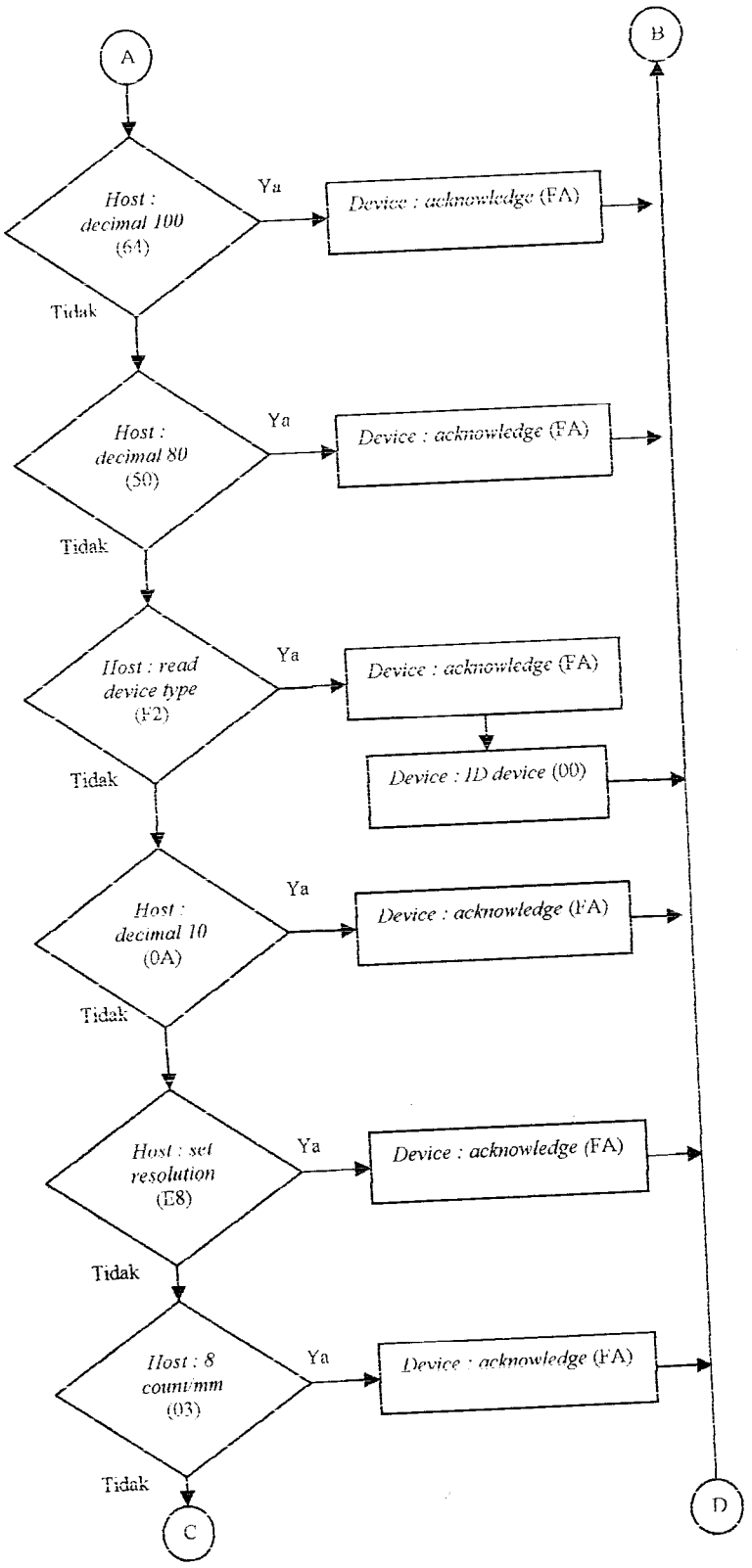
Format byte data dari *device* berupa satu *start bit*, 8 bit data, 1 bit pariti dan 1 *stop bit*. *Device* akan mulai mengirimkan data ke *host*, mula-mula bit *start* dikirimkan dengan cara membuat jalur data menjadi *low* (0) disusul perbahan jalur *clock* menjadi *low* (0). Kemudian 8 bit data yang akan dikirim dipindahkan ke akumulator agar dapat dihasilkan bit pariti. Sebelumnya, jalur *clock* harus dikembalikan ke *high* (1) terlebih dahulu. Bit pariti yang dihasilkan dari akumulator merupakan pariti ganjil, sedangkan PS/2 *port* menggunakan pariti genap, maka bit pariti harus dikomplemenkan menjadi pariti genap. Delapan bit data dikirimkan satu-persatu disertai dengan perubahan *clock* dari *high* (1) menjadi *low* (0). Setelah 8 bit data terkirim dilanjutkan dengan mengirim bit pariti disertai juga dengan perubahan *clock* dari *high* (1) ke *low* (0). Pengiriman diakhiri dengan mengirimkan bit *stop* dengan cara membuat jalur data menjadi *high* disertai pulsa *clock* terakhir.

Mikrokontroler akan menunggu balasan dari komputer dengan mengecek jalur *clock*. Jalur *clock* berubah menjadi *low* menandakan komputer akan melakukan pengiriman *byte* data. Ketika kondisi ini terdeteksi oleh *device*, kemudian *device* akan menanggapinya dengan memulai menghasilkan sinyal *clock*. *Host* mengirimkan bit pada jalur data saat jalur *clock low*, dan akan dibaca oleh *device* pada saat jalur *clock high*. Mikrokontroler akan mencacah sampai 8 yang menandakan kedelapan bit data semuanya selesai dikirimkan. Kemudian mikrokontroler akan menghasilkan dua buah pulsa *clock* dilanjutkan mengirim bit *acknowledge* dengan cara membuat jalur data menjadi *low* disertai dengan menghasilkan pulsa *clock* yang terakhir.

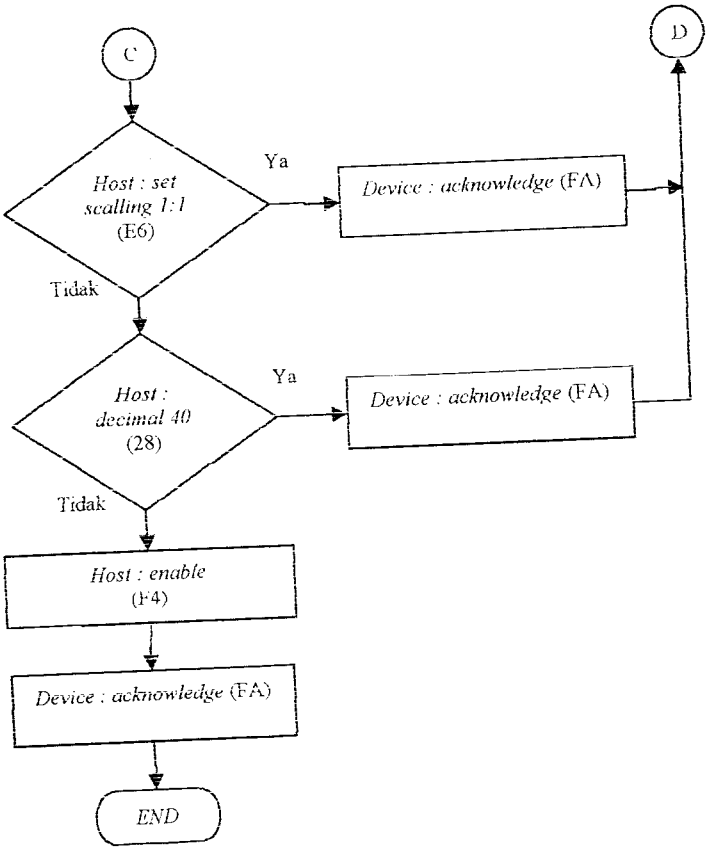
3.3.2 Program Inisialisai dengan PS/2 *Mouse Port*



Gambar 3.13a Diagram Alir Proses Inisialisasi



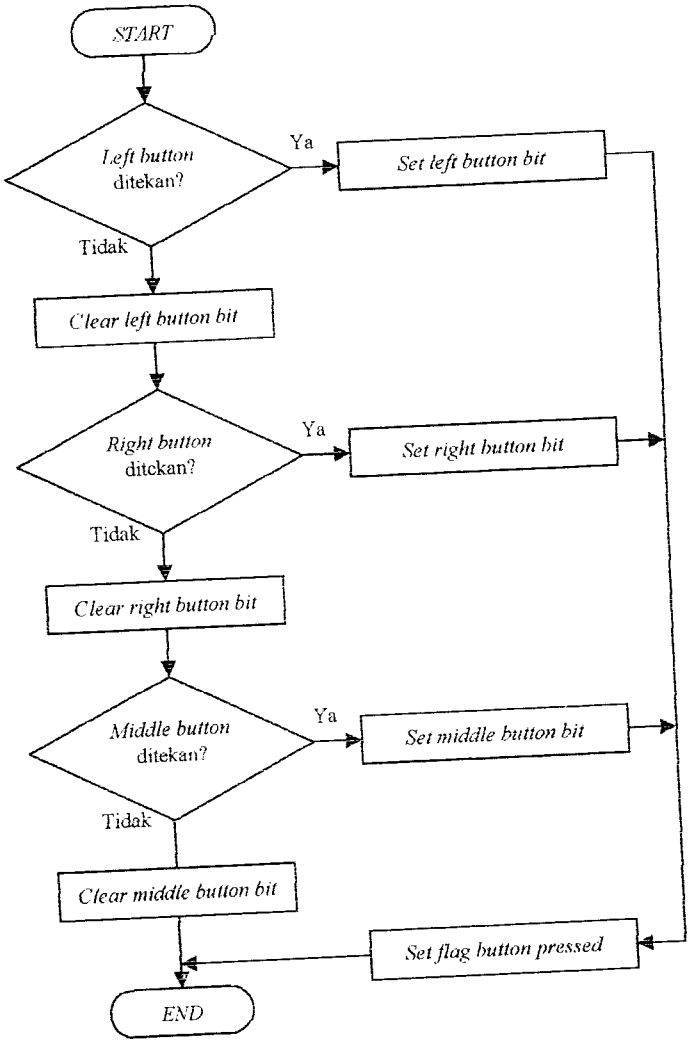
Gambar 3.13b Diagram Alir Proses Inisialisasi



Gambar 3.13c Diagram Alir Proses Inisialisasi

Device akan siap untuk digunakan apabila telah melakukan proses inisialisasi oleh *host* terhadap *device* pada saat komputer dinyalakan. Proses inisialisasi dilakukan untuk menentukan variabel *scaling factor, resolution,* dan lain-lain kepada *device*. *Device* harus membalasnya dengan jawaban yang tepat agar proses *syncronization* dapat berjalan dengan lancar dan *device* dapat digunakan dengan baik tanpa ada masalah.

Proses inisisalisasi hanya akan terjadi saat 500 ms setelah komputer dihidupkan. Mikrokontroler akan menunggu hingga jalur *clock* menjadi *low* dan juga perubahan jalur data menjadi *low*. Kondisi tersebut menandakan komputer



Gambar 3.14 Diagram Alir *Scanning* Tombol

Mikrokontroler akan mengeset P1.7 menjadi *low*, sehingga jika ada *pushbutton* yang ditekan maka P1.4, P1.5 atau P1.6 akan terbaca *low* (0) sesuai dengan letak tombol yang ditekan. *Scanning* dimulai dari membaca keadaan kaki P1.4. Jika tombol *left button* ditekan, kaki P1.4 akan terbaca *low* dan *scanning* tidak akan dilanjutkan. Jika tombol *left button* tidak ditekan, kaki P1.4 akan terbaca *high* dan proses *scanning* akan dilanjutkan menuju kaki P1.6.

Apabila tombol *right button* yang ditekan, maka kaki P1.6 terbaca *low* dan proses *scanning* dihentikan. Jika tombol *right button* tidak ditekan, maka kaki P1.6 terbaca *high* dan proses *scanning* akan dilanjutkan ke kaki P1.5. Apabila hanya tombol *middle button* yang ditekan, maka kaki P1.5 akan terbaca *low*.

Setelah keadaan tombol diketahui, selanjutnya adalah memasukkannya ke dalam paket data yang akan dikirimkan ke komputer. Jika tombol *left button* yang ditekan, maka bit pertama dari *packet data 1* yang menandakan kondisi tombol left button bit akan *set* (1) dan bit yang lain akan *clear* (0).

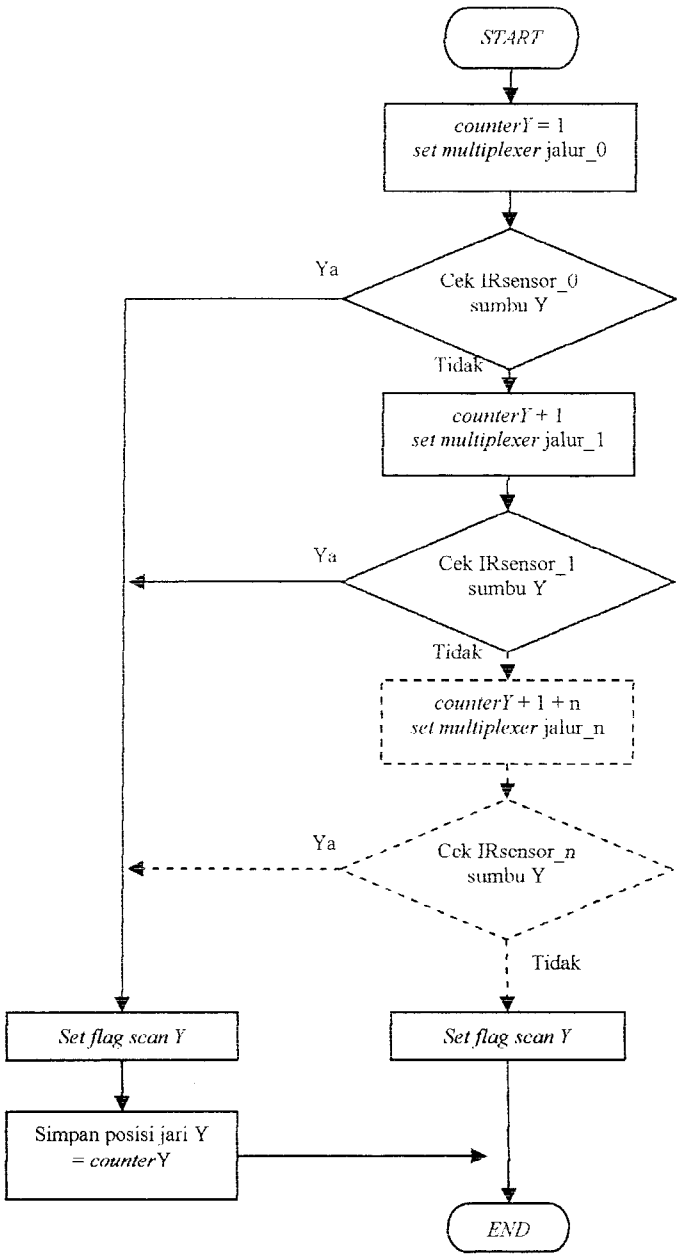
3.3.4 Program *Scanning* Sumbu X dan Sumbu Y

Listing program scanning sumbu X dan sumbu Y berfungsi untuk mendeteksi posisi jari yang berada pada bidang *touchpad*. Jari akan menutupi pancaran cahaya inframerah termodulasi yang akan menuju ke penerima *photodiode*.

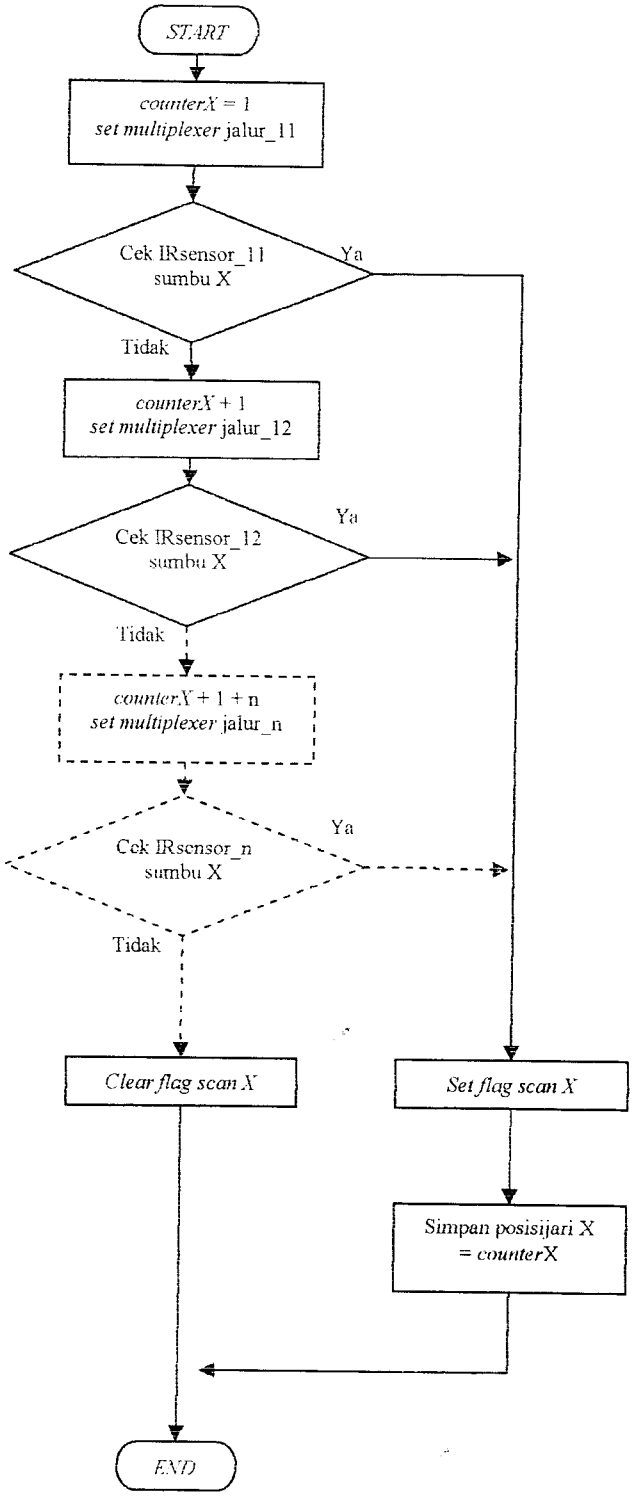
Program modulasi terdiri dari *listing program* modulasi X dan *listing program* modulasi Y. Perbedannya hanya pada lama proses *loop* dilakukan, pada modulasi Y *loop* dilakukan sebanyak 40 kali, sedangkan pada modulasi X hanya dilakukan *loop* sebanyak 15 kali. Hal ini disebabkan karena jarak antara IR LED dengan *photodiode* pada sumbu Y lebih jauh (7 cm) dibandingkan dengan jarak IR LED dengan *photodiode* pada sumbu X (4,5 cm). Jarak yang jauh membutuhkan waktu modulasi yang lebih lama agar *photodiode* dapat menangkap cahaya IR LED dengan baik. Modulasi diawali dengan mengubah *output* menjadi *low*, sehingga IR LED akan menyala. Tundaan NOP akan

menunda lamanya IR menyala sebanyak sebanyak 1 siklus mesin. Jika menggunakan kristal 11.0592 Mhz, maka lamanya 1 siklus mesin adalah 1.085 µs, sehingga jika NOP yang digunakan ada 11, lamanya tundaan adalah 11.9 µs. Lamanya IR menyala yaitu 11.9+1.085 = 12.98 µs. Setelah IR LED menyala selama 13 µs, instruksi SETB OUTPUT akan mematikan IR LED. Perintah NOP digunakan untuk menunda lamanya IR LED padam. Lamanya tundaan NOP yang digunakan tergantung dari besarnya frekuensi yang akan dihasilkan dalam modulasi. LOOP berfungsi untuk mengulang satu gelombang sebanyak n kali agar *demodulator* dapat menerima sinyal termodulasi dengan baik, sehingga dihasilkan *output* yang benar.

Program kedua yang terdapat dalam *program scanning* sumbu X dan Y adalah program pengaturan *multiplexer*. Program ini berfungsi untuk mengendalikan IR LED dan *photodiode* yang aktif melalui jalur-jalur yang ada pada *multiplexer*. Posisi jari dapat ditentukan dengan cara mengetahui *output* dari *sensor infrared* pada kedua buah sumbu yaitu sumbu X dan sumbu Y. *Sensor* tidak semuanya akan aktif dalam waktu yang bersamaan tetapi secara bergantian. Mula-mula yang akan diperiksa adalah sumbu Y terlebih dahulu, baru kemudian sumbu X.



Gambar 3.15a Diagram Alir Proses *Scanning* Sumbu Y



Gambar 3.15b Diagram Alir Proses *Scanning* Sumbu X

Sensor yang terdapat pada sumbu Y ada 11 buah (*sensor_0 – sensor_10*), sehingga memerlukan jalur *multiplexer* sebanyak 11 buah yang dapat disediakan oleh 2 buah IC CD4051 (8 jalur dari IC_0 dan 3 jalur dari IC_1). Mikrokontroler mulai memeriksa sumbu Y dengan terlebih dahulu menetapkan hitungan awal *counter* R1 pada nilai 1. Setelah *counter* bernilai 1, baru kemudian *sensor_0* diaktifkan dengan cara mengisi P2 dengan data 11110000b kemudian CLR P2.7 untuk mengaktifkan IC_0 saja. Mikrokontroler akan mengirimkan data termodulasi ke IR_0 dan kemudian membaca *output* dari *demodulator*. Jika tidak ada objek yang menghalangi *sensor_0* maka mikrokontroler akan menentukan giliran *sensor* yang akan aktif berikutnya, dengan terlebih dahulu menaikkan hitungan *counter* dari 1 menjadi 2. Giliran *sensor* yang aktif setelah *sensor_0* adalah *sensor_1*. Mikrokontroler akan mengaktifkan *sensor_1* dengan terlebih dahulu menaikkan P2 menjadi 11110001b kemudian CLR P2.7 untuk mengaktifkan IC_0 saja. Setelah itu mikrokontroler akan mendeteksi apakah ada objek yang menghalangi *sensor_1*. Jika tidak ada, maka mikrokontroler melanjutkan proses *scanning* dengan terlebih dahulu menaikkan hitungan *counter* dari 2 menjadi 3 kemudian mengaktifkan *sensor_2* dan dicek *output*nya, begitu seterusnya sampai semua *sensor* yang terhubung pada IC_0 selesai di*scanning*. Jika semua jalur pada IC_0 telah diaktifkan satu persatu, yang akan aktif berikutnya adalah jalur_0 pada IC_1. Caranya sama dengan mengaktifkan jalur pada IC_0 hanya CLR P2.7 digantikan oleh CLR P2.6 untuk mengaktifkan IC_1. Jalur pada IC_1

hanya ada 3 buah yang terhubung ke sumbu Y dan sisanya terhubung ke sumbu X.

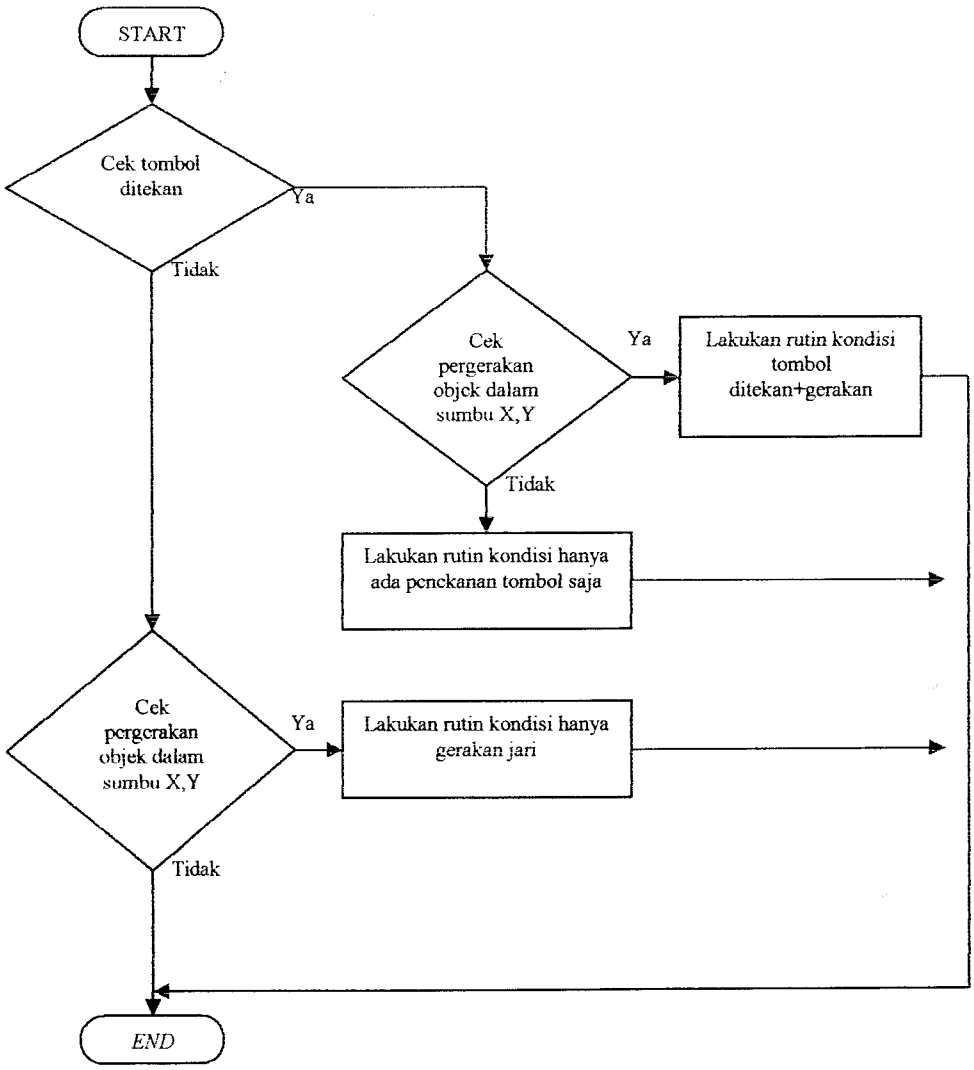
Mikrokontroler membaca *output demodulator* untuk mengetahui ada atau tidaknya objek yang menghalagi *sensor*. Jika ada objek yang menghalagi *sensor* ke-n, maka proses *scanning* langsung dihentikan walaupun belum semua *sensor* telah selesai dicek. Posisi objek dapat diketahui dengan melihat nilai yang ada pada *counter*. Jika nilai pada *counter* bernilai 6, maka objek tepat berada di depan *sensor*_5.

Jika objek telah selesai dideteksi oleh *sensor* sumbu Y, kemudian dilakukan proses *scanning* pada sumbu X. *Sensor* yang digunakan pada sumbu X berjumlah 17 buah (*sensor 11 – sensor 27*), sehingga memerlukan jalur *multiplexer* sebanyak 17 buah yang dapat disediakan oleh 3 buah IC CD4051 (5 jalur dari IC_1, 8 jalur dari IC_2 dan 4 jalur dari IC_3). Proses *scanning* sumbu X sama dengan proses pada *scanning* sumbu Y. *Counter* dimulai dari 1, kemudian satu per satu *sensor* dicek dan juga menaikkan hitungan *counter*nya. Proses *scanning* dimulai dari mengaktifkan *sensor*_11 sumbu X yang terhubung ke IC_1 dengan cara mengisi P2 oleh data 11110011b kemudian CLR P2.6. Mikrokontroler mengecek *output demodulator* ada atau tidaknya objek yang menghalangi *sensor*_11. Jika tidak ada objek yang menghalangi *sensor* 11, proses *scanning* dilanjutkan dengan mengecek *sensor*_12 dan menaikkan hitungan *counter* R1 dari 1 menjadi 2. Jika tetap objek tidak ditemukan, mikrokontroler akan mengecek *sensor-sensor* berikutnya hingga *sensor* yang terakhir yaitu *sensor*_27 dengan nilai *counter* R1 berisi 17. Proses *scanning*

dihentikan setelah objek terdeteksi dan kemudian nilai *counter* disimpan. Posisi objek diketahui dengan melihat nilai yang ada pada dua buah *counter* (X dan Y).

3.3.5 Program Utama Kendali Touchpad

Program ini berisi program utama *touchpad* berbasis mikrokontroler AT89C51. Pengaruh dari pergerakan jari dan penekanan tombol akan mengakibatkan posisi *cursor* bergeser serta fungsi seperti *click* kiri, *double click* dan *hold button* akan ditentukan sesuai isi dari program ini.



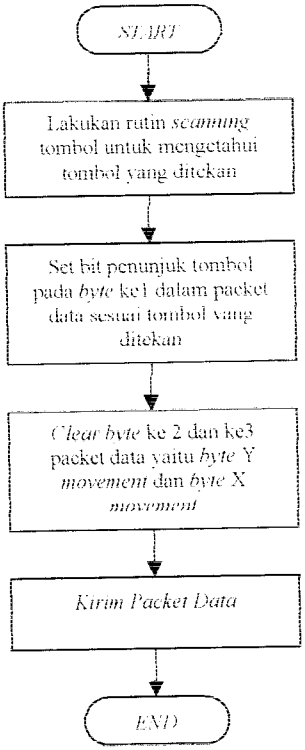
Gambar 3.16 Diagram Alir Program Program Utama *Touchpad*

Program utama *touchpad* memiliki tiga buah keadaan atau kondisi yang dapat terjadi yaitu:

1. Kondisi ada penekanan tombol saja
2. Kondisi hanya ada pergerakan jari
3. Kondisi ada penekanan tombol dan pergerakan jari

3.3.5.1 Kondisi Penekanan Tombol

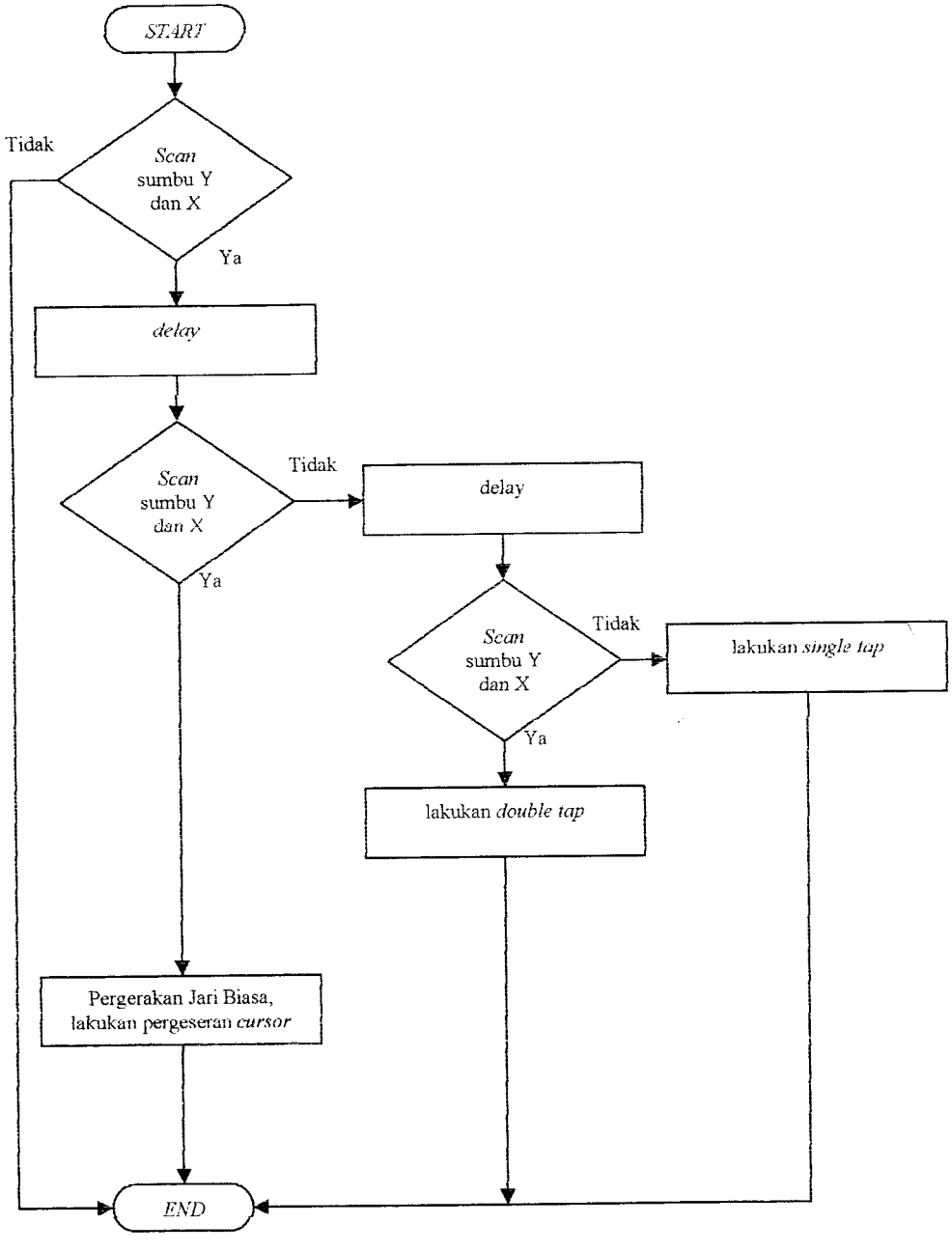
Kondisi pertama yaitu kondisi yang hanya terdapat penekanan tombol saja tanpa disertai adanya sentuhan jari ke bidang *touchpad*. Kondisi ini mengakibatkan terjadinya *click* kiri, *click* kanan atau *click* tengah tanpa disertai pergeseran *cursor*. Rutin *scanning* tombol dijalankan untuk mengetahui letak tombol yang ditekan. *Bit* penunjuk tombol yang terdapat pada *byte* ke 1 *packet data* akan *set* sesuai letak tombol yang ditekan, sedangkan *byte* ke 3 dan ke 4 *packet data* dinolkan. Jika terjadi penekanan lebih dari dua tombol, mikrokontroler akan mengambil salah satu tombol saja. Hal ini disebabkan ketidakmampuan komputer untuk menangani penekanan tombol lebih dari satu. Tombol yang diambil ketika terjadi penekanan tombol lebih dari satu adalah tombol yang memiliki prioritas lebih tinggi dibandingkan dengan tombol lain yang juga ditekan. Prioritas utama adalah tombol *left button*, prioritas kedua yaitu tombol *right button* dan yang ketiga adalah tombol dengan prioritas paling rendah yaitu *middle button*.



Gambar 3.17 *Flowchart* Rutin Kondisi Hanya Ada Penekanan Tombol Saja

3.3.5.2 Kondisi Pergerakan Jari

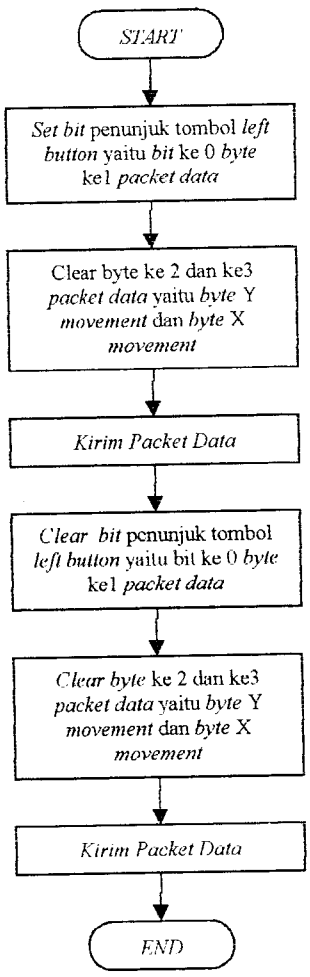
Kondisi kedua yaitu kondisi yang hanya terdapat pergerakan jari saja tanpa disertai dengan penekanan tombol. Kondisi ini mengakibatkan terjadinya *single tap*, *double tap* dan pergeseran *cursor* pada layar monitor.



Gambar 3.18 *Flowchart* Rutin Kondisi Hanya Ada Pergerakan Jari Saja

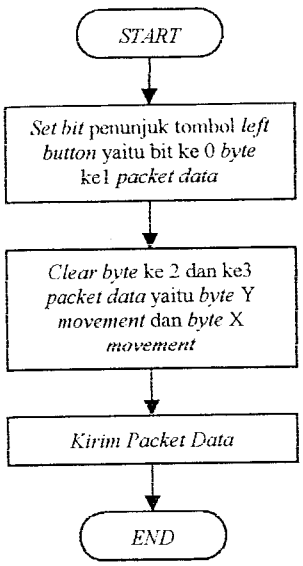
Program dimulai dengan *scanning* sumbu Y dan sumbu X untuk mengetahui ada atau tidaknya jari. Apabila jari ditemukan berada pada bidang

sentuh, kemudian program delay akan dijalankan untuk memberikan waktu jari untuk berpindah. *Scanning* sumbu Y dan sumbu X dilakukan kembali untuk mengetahui jari masih berada pada bidang sentuh atau tidak. Jika jari sudah tidak berada pada bidang *touchpad,* program *delay* kembali dijalankan untuk memberikan waktu jari kembali ke bidang sentuh. *Scanning* sumbu Y kembali dilakukan untuk mencari jari, jika jari tidak ditemukan program akan mendeteksi keadaan ini sebagai *single tap.* Sedangkan apabila jari kembali ditemukan, program akan mendeteksinya sebagai *double tap.*



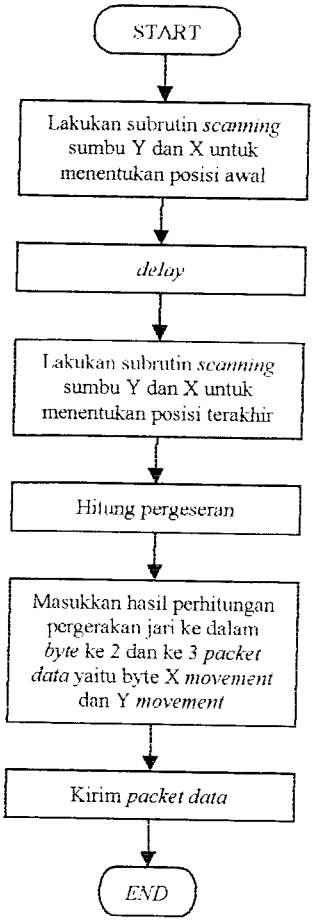
Gambar 3.19 Diagram Alir Penanganan *Single Tap*

Single tap berfungsi untuk menggantikan *click* kiri. Apabila terjadi *single tap*, program akan mengeset *bit* ke 0 pada *byte* ke 1 *packet data* yang merupakan *bit* penunjuk *left button* sedangkan *byte* ke 3 dan ke 4 dikosongkan. *Packet data* dikirimkan, kemudian *bit* ke 0 kembali dinolkan. *Packet data* dikirimkan kembali.



Gambar 3.20 Diagram Alir Penanganan *Double Tap*

Double tap berfungsi seperti saat tombol kiri ditahan. Penanganan *double tap* hampir sama dengan *single tap* hanya berbeda pengirimannya dilakukan satu kali. *Bit* ke 0 diset, *byte* ke 2 dan ke 3 dikosongkan, kemudian *packet data* dikirimkan.



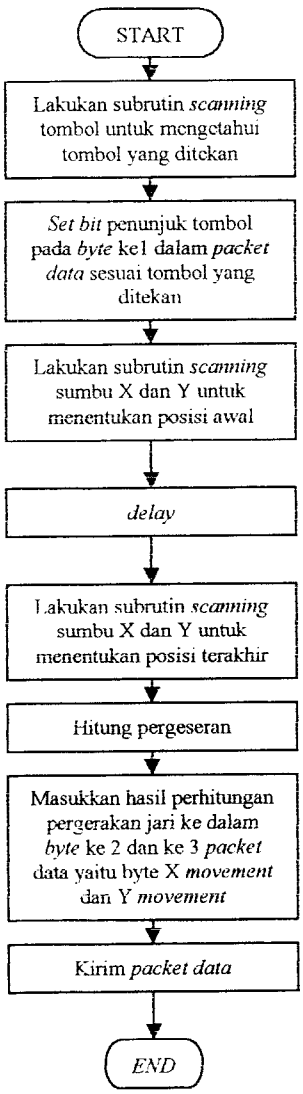
Gambar 3.21 Diagram Alir Penanganan Gerakan Jari Biasa

Penanganan gerakan jari biasa dilakukan untuk memperoleh data pergeseran jari. Rutin *scan* sumbu Y dan sumbu X dijalankan untuk mendapatkan posisi awal jari dalam koordinat X,Y. Setelah itu, *delay* dijalankan untuk memberi waktu jari untuk bergeser posisinya. Rutin *scan* sumbu Y dan sumbu X dilakukan kembali untuk memperoleh posisi terakhir jari setelah bergeser. Koordinat posisi terakhir dikurangi dengan koordinat posisi awal, apabila hasilnya negatif menandakan jari bergeser ke arah kiri untuk sumbu X

atau ke arah bawah untuk sumbu Y. Hasil perhitungan dimasukkan ke dalam *byte* ke 2 dan ke 3 *packet data* untuk kemudian dikirimkan ke komputer.

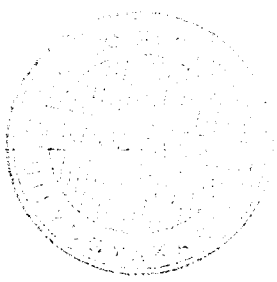
3.3.5.3 Kondisi Pergeseran Jari dan Penekanan Tombol

Kondisi ketiga yaitu kondisi penekanan tombol disertai dengan adanya pergerakan jari.



Gambar 3.22 *Flowchart* Rutin Kondisi Pergeseran Jari dan Penekanan Tombol

Program dimulai dengan mendeteksi keadaan tombol, jika tombol ditekan program akan mengeset salah satu *bit* data dalam *packet byte data* yang merupakan bit-bit keadaan *pushbutton*. Program dilanjutkan dengan proses *scanning* sumbu Y dan X untuk mendapatkan posisi awal jari. *Delay* dilakukan untuk memberikan waktu jari untuk bergeser. Seteleh itu *scanning* sumbu Y dan sumbu X kembali dilakukan untuk mendapatkan posisi akhir jari. Koordinat posisi terakhir dikurangi dengan koordinat posisi awal, apabila hasilnya negatif menandakan jari bergeser ke arah kiri untuk sumbu X atau ke arah bawah untuk sumbu Y. Hasil perhitungan dimasukkan ke dalam *byte* ke 2 dan ke 3 *packet* data untuk kemudian dikirimkan ke komputer.



BAB IV

ANALISIS DAN PEMBAHASAN

Perangkat keras *touchpad* ini dibentuk dalam beberapa rangkaian yang terdiri dari rangkaian catu daya, rangkaian mikrokontroler, rangkaian *multiplexer-demultiplexer*, dan rangkaian *demodulator*. *Touchpad* dihubungkan dengan komputer melalui PS/2 *port*. Komputer yang digunakan dalam pengujian memiliki spesifikasi sebagai berikut :

1. *Processor* Intel Pentium 1.8 GHz
2. RAM 512 M*byte*
3. Sistem Operasi Windows XP SP2
4. *Monitor* 17 inchi resolusi 800 x 600

4.1 Pengujian *Hardware*

4.1.1 Pengujian Rangkaian Catu Daya

Rangkaian catu daya yang digunakan berasal dari *Port* PS/2 untuk mensuplai daya 5 volt bagi keseluruhan rangkaian. Besar tegangan yang dihasilkan dari *Port* PS/2 diperlihatkan pada tabel 4.1.

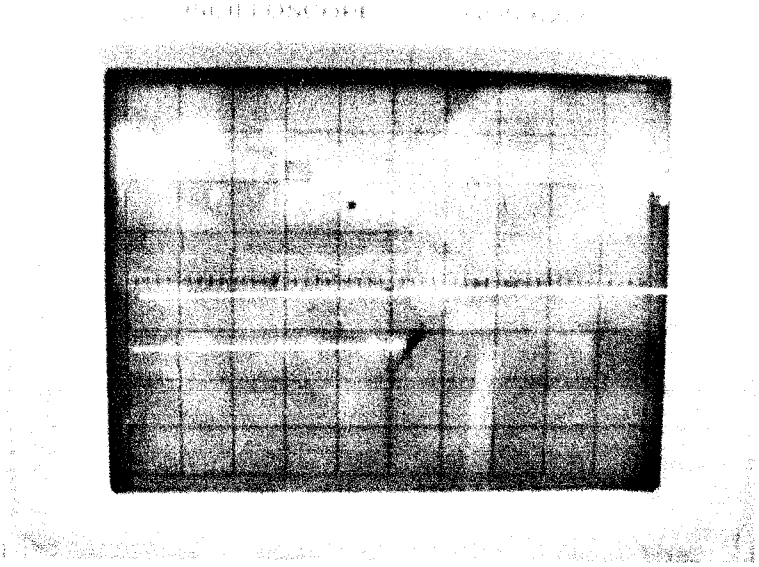
Tabel 4.1 Tegangan Keluaran *Port* PS/2

Karakteristik	Maksimum	Minimum
Tegangan Keluaran	5,01 volt	4,98 volt

Hasil pengukuran tegangan tidak murni 5 volt, tetapi sebesar 4,98 – 5,01 volt. Hasil tersebut dikarenakan beberapa faktor, diantaranya kualitas dari PSU (*Power Suply Unit*) yang digunakan oleh komputer bukan yang terbaik. Selain itu, tegangan jala-jala listrik yang digunakan tidak stabil.

4.1.2 Pengujian Rangkaian *Modulator*

Pengujian rangkaian *modulator* dilakukan dengan cara mengukur tegangan output port 1.0 mikrokontroler menggunakan *oscilloscope*. Gambar 4.1 menunjukkan hasil pengujian rangkaian *modulator*.



Gambar 4.1 Sinyal Modulasi

4.1.3 Pengujian Rangkaian *Multiplexer-Demultiplexer*

Pengujian rangkaian *multiplexer* dilakukan dengan cara mengukur tegangan sinyal modulasi pada IR LED yang aktif setelah rangkaian *multiplexer* dikendalikan oleh mikrokontroler melalui *port* 2.

Tabel 4.2 Hasil Pengujian Rangkaian *Multiplexer*

Bit Output Port 2								IR LED Aktif	Tegangan IR LED Aktif		Tegangan IR LED Tidak Aktif	
7	6	5	4	3	2	1	0		Kaki (+)	Kaki (-)	Kaki (+)	Kaki (-)
0	1	1	1	X	0	0	0	IR LED 0	2,85 V	1,85 V	0,72 V	1,88 V
0	1	1	1	X	0	0	1	IR LED 1	2,81 V	1,86 V	0,80 V	1,86 V
0	1	1	1	X	0	1	0	IR LED 2	2,84 V	1,86 V	0,84 V	1,86 V
0	1	1	1	X	0	1	1	IR LED 3	2,79 V	1,85 V	0,77 V	1,87 V
0	1	1	1	X	1	0	0	IR LED 4	2,77 V	1,85 V	0,78 V	1,86 V
0	1	1	1	X	1	0	1	IR LED 5	2,83 V	1,88 V	0,85 V	1,88 V
0	1	1	1	X	1	1	0	IR LED 6	2,85 V	1,89 V	0,81 V	1,89 V
0	1	1	1	X	1	1	1	IR LED 7	2,82 V	1,85 V	0,74 V	1,85 V
1	0	1	1	X	1	0	0	IR LED 8	2,82 V	1,85 V	0,77 V	1,85 V
1	0	1	1	X	0	0	1	IR LED 9	2,79 V	1,83 V	0,77 V	1,89 V
1	0	1	1	X	0	1	0	IR LED 10	2,83 V	1,85 V	0,78 V	1,90 V
1	0	1	1	X	0	1	1	IR LED 11	2,85 V	1,85 V	0,79 V	1,86 V
1	0	1	1	X	0	0	0	IR LED 12	2,85 V	1,85 V	0,81 V	1,88 V
1	0	1	1	X	1	0	1	IR LED 13	2,84 V	1,85 V	0,78 V	1,88 V
1	0	1	1	X	1	1	0	IR LED 14	2,84 V	1,86 V	0,76 V	1,87 V
1	0	1	1	X	1	1	1	IR LED 15	2,83 V	1,85 V	0,76 V	1,88 V
1	1	0	1	X	1	0	0	IR LED 16	2,85 V	1,84 V	0,75 V	1,87 V
1	1	0	1	X	1	0	1	IR LED 17	2,86 V	1,83 V	0,78 V	1,89 V
1	1	0	1	X	0	1	0	IR LED 18	2,83 V	1,84 V	0,75 V	1,88 V
1	1	0	1	X	0	1	1	IR LED 19	2,84 V	1,85 V	0,73 V	1,88 V
1	1	0	1	X	0	0	0	IR LED 20	2,85 V	1,86 V	0,73 V	1,86 V
1	1	0	1	X	0	0	1	IR LED 21	2,86 V	1,86 V	0,73 V	1,86 V
1	1	0	1	X	1	1	0	IR LED 22	2,86 V	1,84 V	0,76 V	1,85 V
1	1	0	1	X	1	1	1	IR LED 23	2,85 V	1,85 V	0,74 V	1,85 V
1	1	1	0	X	1	0	0	IR LED 24	2,75 V	1,83 V	0,78 V	1,89 V
1	1	1	0	X	1	0	1	IR LED 25	2,85 V	1,87 V	0,78 V	1,87 V
1	1	1	0	X	1	1	0	IR LED 26	2,86 V	1,86 V	0,74 V	1,86 V
1	1	1	0	X	0	1	1	IR LED 27	2,86 V	1,85 V	0,74 V	1,88 V

Pengujian rangkaian *demultiplexer* dilakukan dengan cara mengukur tegangan sinyal modulasi pada *photodiode* yang aktif setelah rangkaian *demultiplexer* dikendalikan oleh mikrokontroler melalui *port* 2.

Tabel 4.3 Hasil Pengujian Rangkaian *Demultiplexer*

Bit Output Port 2								*Photodiode* Aktif	Tegangan *Photodiode* Aktif Tanpa Penghalang		Tegangan *Photodiode* Aktif Saat Ada Penghalang	
7	6	5	4	3	2	1	0		Kaki (+)	Kaki (-)	Kaki (+)	Kaki (-)
0	1	1	1	X	0	0	0	*Photodiode* 0	17,0 mV	4,04 V	6,6 mV	4,04 V
0	1	1	1	X	0	0	1	*Photodiode* 1	17,1 mV	4,04 V	6,6 mV	4,04 V
0	1	1	1	X	0	1	0	*Photodiode* 2	17,1 mV	4,04 V	6,4 mV	4,04 V
0	1	1	1	X	0	1	1	*Photodiode* 3	17,3 mV	4,04 V	6,5 mV	4,04 V
0	1	1	1	X	1	0	0	*Photodiode* 4	17,0 mV	4,04 V	6,4 mV	4,04 V
0	1	1	1	X	1	0	1	*Photodiode* 5	17,0 mV	4,04 V	6,6 mV	4,04 V
0	1	1	1	X	1	1	0	*Photodiode* 6	17,1 mV	4,04 V	6,4 mV	4,04 V
0	1	1	1	X	1	1	1	*Photodiode* 7	17,0 mV	4,04 V	6,6 mV	4,04 V
1	0	1	1	X	1	0	0	*Photodiode* 8	17,0 mV	4,04 V	6,6 mV	4,04 V
1	0	1	1	X	0	0	1	*Photodiode* 9	17,0 mV	4,04 V	6,6 mV	4,04 V
1	0	1	1	X	0	1	0	*Photodiode* 10	17,0 mV	4,04 V	6,6 mV	4,04 V
1	0	1	1	X	0	1	1	*Photodiode* 11	28,4 mV	4,01 V	11,3 mV	4,01 V
1	0	1	1	X	0	0	0	*Photodiode* 12	28,3 mV	4,01 V	11,3 mV	4,01 V
1	0	1	1	X	1	0	1	*Photodiode* 13	28,3 mV	4,01 V	11,3 mV	4,01 V
1	0	1	1	X	1	1	0	*Photodiode* 14	28,3 mV	4,01 V	11,3 mV	4,01 V
1	0	1	1	X	1	1	1	*Photodiode* 15	28,3 mV	4,01 V	11,3 mV	4,01 V
1	1	0	1	X	1	0	0	*Photodiode* 16	28,4 mV	4,01 V	11,3 mV	4,01 V
1	1	0	1	X	1	0	1	*Photodiode* 17	28,3 mV	4,01 V	11,3 mV	4,01 V
1	1	0	1	X	0	1	0	*Photodiode* 18	28,4 mV	4,01 V	11,2 mV	4,01 V
1	1	0	1	X	0	1	1	*Photodiode* 19	28,4 mV	4,01 V	11,3 mV	4,01 V
1	1	0	1	X	0	0	0	*Photodiode* 20	28,4 mV	4,01 V	11,2 mV	4,01 V
1	1	0	1	X	0	0	1	*Photodiode* 21	28,4 mV	4,01 V	11,2 mV	4,01 V
1	1	0	1	X	1	1	0	*Photodiode* 22	28,5 mV	4,01 V	11,3 mV	4,01 V
1	1	0	1	X	1	1	1	*Photodiode* 23	28,5 mV	4,01 V	11,3 mV	4,01 V
1	1	1	0	X	1	0	0	*Photodiode* 24	28,3 mV	4,01 V	11,2 mV	4,01 V
1	1	1	0	X	1	0	1	*Photodiode* 25	28,4 mV	4,01 V	11,3 mV	4,01 V
1	1	1	0	X	1	1	0	*Photodiode* 26	28,4 mV	4,01 V	11,3 mV	4,01 V
1	1	1	0	X	0	1	1	*Photodiode* 27	28,4 mV	4,01 V	11,3 mV	4,01 V

4.1.4 Pengujian Rangkaian *Demodulator*

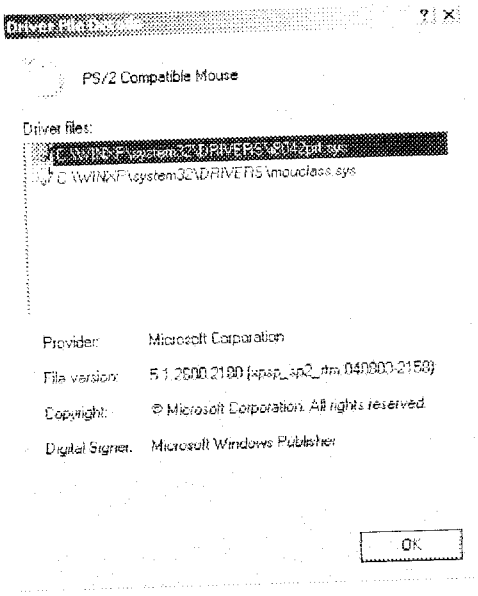
Pengujian rangkaian *demodulator* dilakukan dengan mengukur tegangan pada keluaran IRM-8601 untuk mengetahui besarnya tegangan keluaran pada saat

demodulator bekerja, sehingga dapat memberikan logika '0' atau logika '1' pada mikrokontroler.

Tabel 4.4 Hasil Pengukuran Tegangan Keluaran *Demodulator*

Sumbu	Sensor	*Output Demodulator*	
		Tidak Ada Penghalang	Ada Penghalang
Y	Sensor_0	47,0 mV	3,8 V
Y	Sensor_1	47,1 mV	3,9 V
Y	Sensor_2	47,0 mV	3,8 V
Y	Sensor_3	47,0 mV	3,8 V
Y	Sensor_4	47,2 mV	3,9 V
Y	Sensor_5	47,2 mV	3,9 V
Y	Sensor_6	47,1 mV	3,8 V
Y	Sensor_7	47,0 mV	3,8 V
Y	Sensor_8	47,0 mV	3,8 V
Y	Sensor_9	47,0 mV	3,8 V
Y	Sensor_10	47,0 mV	3,8 V
X	Sensor_11	47,0 mV	3,8 V
X	Sensor_12	46,9 mV	3,7 V
X	Sensor_13	46,9 mV	3,7 V
X	Sensor_14	46,9 mV	3,7 V
X	Sensor_15	47,0 mV	3,8 V
X	Sensor_16	47,0 mV	3,8 V
X	Sensor_17	47,1 mV	3,8 V
X	Sensor_18	46,9 mV	3,7 V
X	Sensor_19	47,0 mV	3,8 V
X	Sensor_20	47,0 mV	3,8 V
X	Sensor_21	47,0 mV	3,8 V
X	Sensor_22	47,1 mV	3,8 V
X	Sensor_23	47,1 mV	3,8 V
X	Sensor_24	46,9 mV	3,7 V
X	Sensor_25	47,0 mV	3,8 V
X	Sensor_26	47,0 mV	3,8 V
X	Sensor_27	47,0 mV	3,8 V

Pengujian diatas memberikan dua logika masukan ke mikrokontroler. Logika '0' untuk tegangan 46,9 mV – 47,2 mV dan logika '1' untuk tegangan antara 3,7 V – 3,9 V. Keluaran demodulator terhubung dengan *port* P1.1 mikrokontroler.



Gambar 4.3 *Driver File Details*

4.2.2 Pengujian Pengendalian Arah Gerak *Cursor*

Pengujian ini dilakukan dengan cara menggerakkan *cursor* melalui *touchpad*. Jari diletakkan pada bidang sentuh kemudian digerakkan menuju arah tertentu agar *cursor* pada monitor ikut bergerak mengikuti arah pergerakan jari. *Setting mouse* yang digunakan yaitu:

- *Speed maximum*.

  Dilakukan melalui ***Start→Control Panel→Mouse→Pointer Options***, geser sampai didapatkan kecepatan maksimal.

- *Enhance Pointer Precision* Aktif

a. Geser Kanan

Posisi awal *cursor* diletakkan di sisi kiri layar monitor, dapat dilakukan dengan menggunakan *keyboard* atau *mouse usb* untuk memindahkan *cursor*. Setelah posisi *mouse* berada pada sebelah kiri monitor, geser *cursor* ke arah kanan dengan menggunakan *touchpad*. Tabel 4.5 merupakan tabel hasil pegujian *touchpad* untuk menggerakkan *cursor* ke arah kanan.

Tabel 4.5 Hasil Pengujian Geser Kanan

Pengujian ke :	Gerakan Jari	Pergeseran *Cursor*
1	pelan	2 cm
2	sedang	3,5 cm
3	cepat	8,5 cm

Semakin cepat jari digeser, semakin jauh pergeseran *cursor*, sedangkan semakin pelan jari digeser, semakin dekat pergeseran *cursor*. Hal ini dapat terjadi karena *enhance pointer precision* diaktifkan. Jika *enhance pointer precision* dimatikan, jauh dekatnya pergeseran *cursor* hanya dipengaruhi oleh jauh dekatnya jari digeser bukan oleh cepat lambatnya jari digeser.

b. Geser Kiri

Posisi awal *cursor* diletakkan di sisi kanan layar monitor, dapat dilakukan dengan menggunakan *keyboard* atau *mouse usb* untuk memindahkan *cursor*. Setelah posisi *mouse* berada pada sebelah kanan monitor, geser *cursor* ke arah kiri dengan menggunakan *touchpad*. Tabel 4.6 merupakan tabel hasil pegujian *touchpad* untuk menggerakkan *cursor* ke arah kiri.

Tabel 4.6 Hasil Pengujian Geser kiri

Pengujian ke :	Gerakan Jari	Pergeseran *Cursor*
1	pelan	2 cm
2	sedang	3,5 cm
3	cepat	8,5 cm

c. Geser Atas

Posisi awal *cursor* diletakkan di sisi bawah layar monitor, dapat dilakukan dengan menggunakan *keyboard* atau *mouse usb* untuk memindahkan *cursor*. Setelah posisi *mouse* berada pada sebelah bawah monitor, geser *cursor* ke arah atas dengan menggunakan *touchpad*. Tabel 4.7 merupakan tabel hasil pegujian *touchpad* untuk menggerakkan *cursor* ke arah atas.

Tabel 4.7 Hasil Pengujian Geser Atas

Pengujian ke :	Gerakan Jari	Pergeseran *Cursor*
1	pelan	0,5 cm
2	sedang	8 cm
3	cepat	23 cm

d. Geser bawah

Posisi awal *cursor* diletakkan di sisi atas layar monitor, dapat dilakukan dengan menggunakan *keyboard* atau *mouse usb* untuk memindahkan *cursor*. Setelah posisi *mouse* berada pada sebelah atas monitor, geser *cursor* ke arah bawah dengan menggunakan *touchpad*. Tabel 4.8 merupakan tabel hasil pegujian *touchpad* untuk menggerakkan *cursor* ke arah bawah.

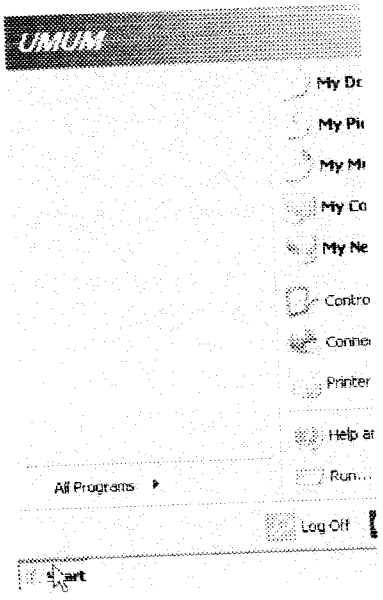
Tabel 4.8 Hasil Pengujian Geser Bawah

Pengujian ke :	Gerakan Jari	Pergeseran *Cursor*
1	pelan	0,5 cm
2	sedang	8 cm
3	cepat	23 cm

4.2.3 Pengujian *Pushbutton*

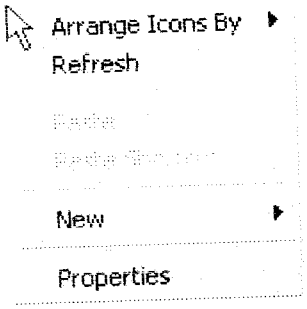
Pengujian dilakukan dengan cara menekan tombol yang ada pada *touchpad* yang memiliki fungsi yang sama dengan tombol pada *mouse* biasa. *Touchpad* memiliki tiga buah tombol yaitu *left button*, *right button* dan *middle button*.

Pengujian *left button* dilakukan dengan cara mengarahkan *cursor* ke tombol *start* yang ada pada sudut kiri bawah layar monitor. Setelah *cursor* berada di tombol *start*, *left button touchpad* ditekan dan dilepas. Gambar 4.4 menunjukkan hasil dari penekanan *left button* pada tombol *start*.



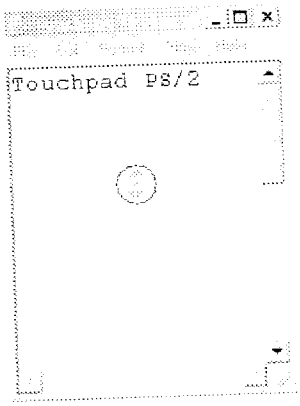
Gambar 4.4 *Left Button Click*

Pengujian *right button* dilakukan dengan menekan *right button* kemudian melepasnya pada *wallpaper* layar monitor. Gambar 4.5 menunjukkan hasil penekanan *right button* pada *wallpaper* komputer.

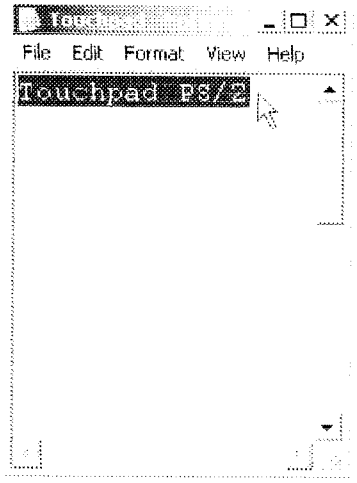


Gambar 4.5 *Right Button Click*

Pengujian *middle button* dilakukan dengan cara menekan *middle button* saat *cursor* berada pada program *text editor notepad* yang memiliki *vertical scroll bar*. *Middle button* ditekan kemudian dilepas pada bagian *text* sehingga *cursor* berubah wujudnya menjadi gambar panah ke atas dan ke bawah. Gambar 4.6 menunjukkan hasil penekanan *middle button*.



Gambar 4.6 *Middle Button Click*



Gambar 4.8 *Double Tap*

BAB V

PENUTUP

5.1 Kesimpulan

Berdasarkan hasil perancangan dan pengujian rangkaian "Perancangan *Touchpad* PS/2 Berbasis Mikrokontroler AT89C51", maka dapat diambil kesimpulan sebagai berikut :

1. Proses yang terjadi pada *touchpad* PS/2 dalam mengendalikan cursor yaitu proses pembacaan posisi dan pembacaan keadaan tombol, proses penentuan fungsi yang terjadi akibat dari pembacaan posisi dan tombol, dan proses komunikasi serial secara sinkron.
2. *Optocoupler* dapat dimanfaatkan sebagai alat yang dapat membaca posisi benda secara dua dimensi jika dirangkaikan pada dua buah sumbu X dan sumbu Y.
3. Penggunaan sinyal termodulasi pada IR LED akan memperpanjang jarak pancaran dan membuat sinyal menjadi lebih kebal terhadap gangguan dari luar.
4. Fungsi-fungsi yang dapat dilakukan *touchpad* sama seperti fungsi-fungsi yang dapat dilakukan oleh *mouse* standar biasa, seperti menggerakkan *cursor* ke segala arah, *click* kiri, *click* kanan dan *click* tengah.

5. Komunikasi dua arah antara mikrokontroler dan personal komputer (PC) dapat dilaksanakan dengan baik dengan memanfaatkan pengiriman data secara *bidirectional synchronous serial* melalui PS/2 *port*.

5.2 Saran

1. *Optocoupler* yang digunakan dibuat lebih banyak, lebih rapat dan lebih tipis, sehingga gerakan cursor lebih halus dan akurat.
2. Penambahan fungsi *scroll* ke atas dan ke bawah.
3. Penambahan *pushbuttons* sebagai tombol *shortcut* yang dapat diprogram.

DAFTAR PUSTAKA

Budiharto, Widodo, 2004. *Interfacing Komputer dan Mikrokontroler*. Jakarta: PT.Elex Media Komputindo.

Nalwan, A.P, 2003. *Teknik Antarmuka dan Pemrograman Mikrokontroler AT89C51*. Jakarta: PT.Elex Media Komputindo.

Prasetia, Retna, 2004. *Interfacing Port Paralel dan Port Serial Komputer dengan Visual Basic 6.0* . Yogyakarta: ANDI.

Putra, Agfianto Eko, 2004 *Belajar Mikrokontroler AT89C51/52/55 Teori dan Aplikasi*. Edisi II.. Yogyakarta: Gava Media.

Sudjadi, 2005. *Teori dan Aplikasi Mikrokontroler Aplikasi pada Mikrokontroler AT89C51*. Yogyakarta: Graha Ilmu.

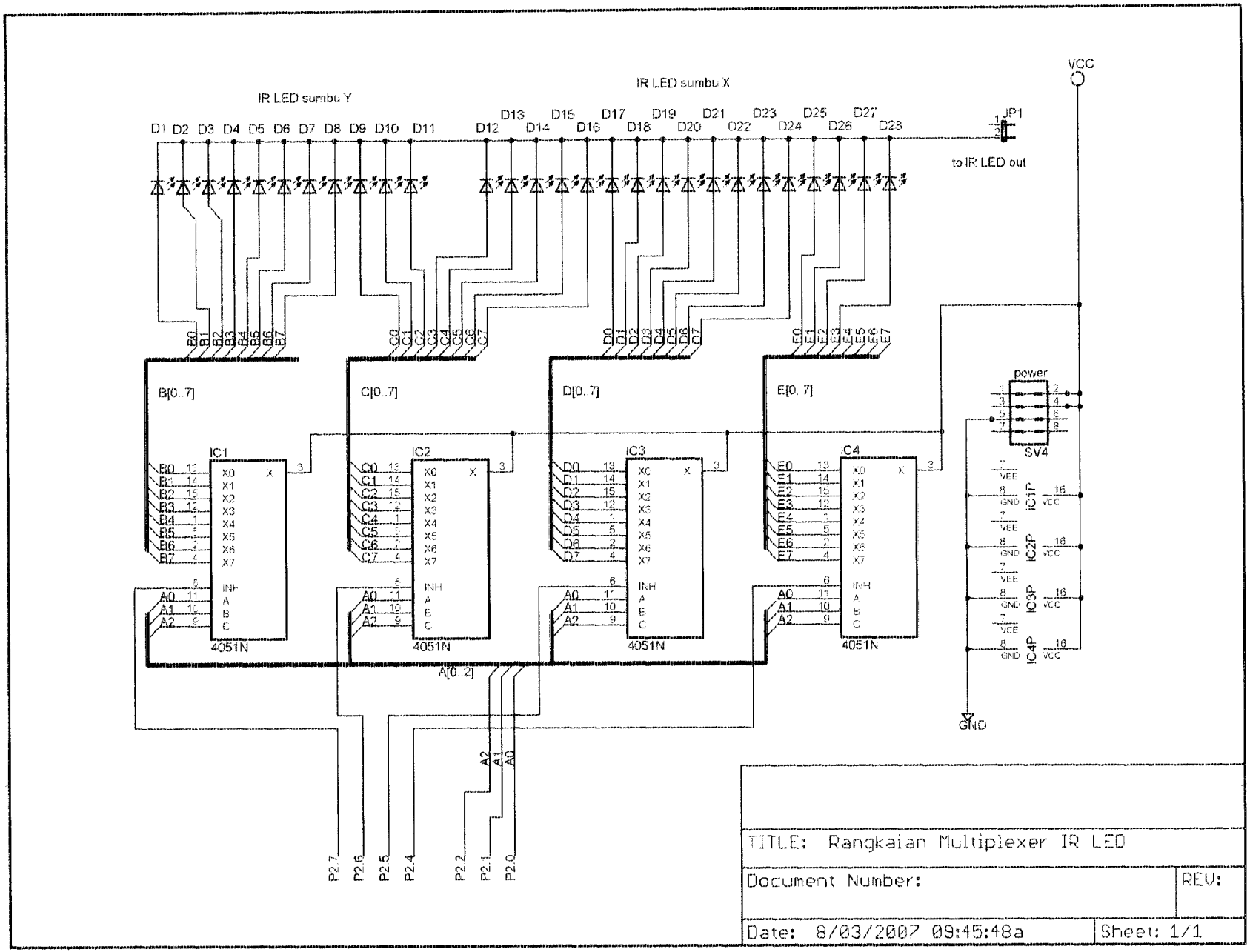
http://www.atmel.com.2001

http://www.computer-engineering.org

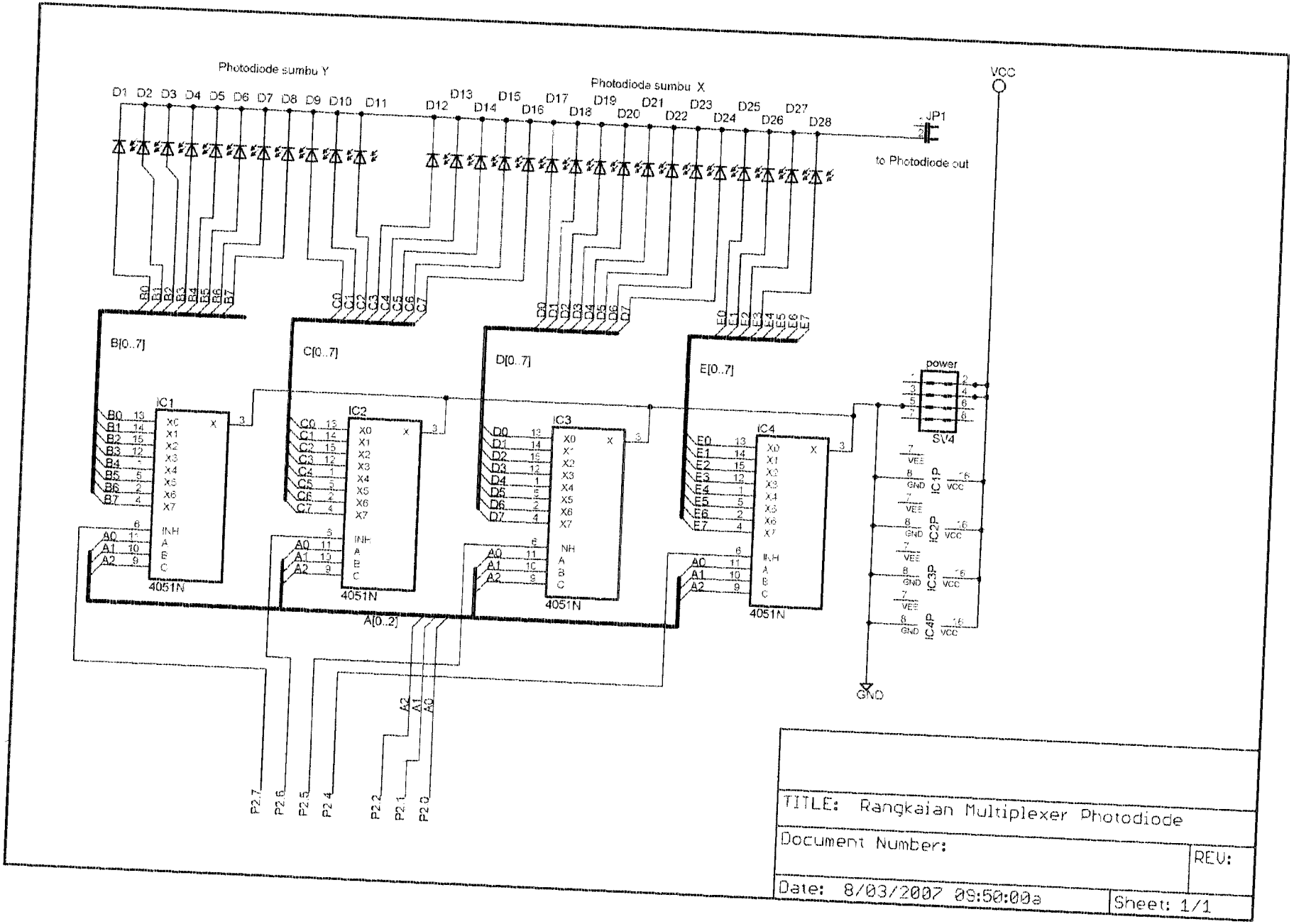
http://www.hep.wisc.edu/~gowrisha/554/Ps2-vga.pdf

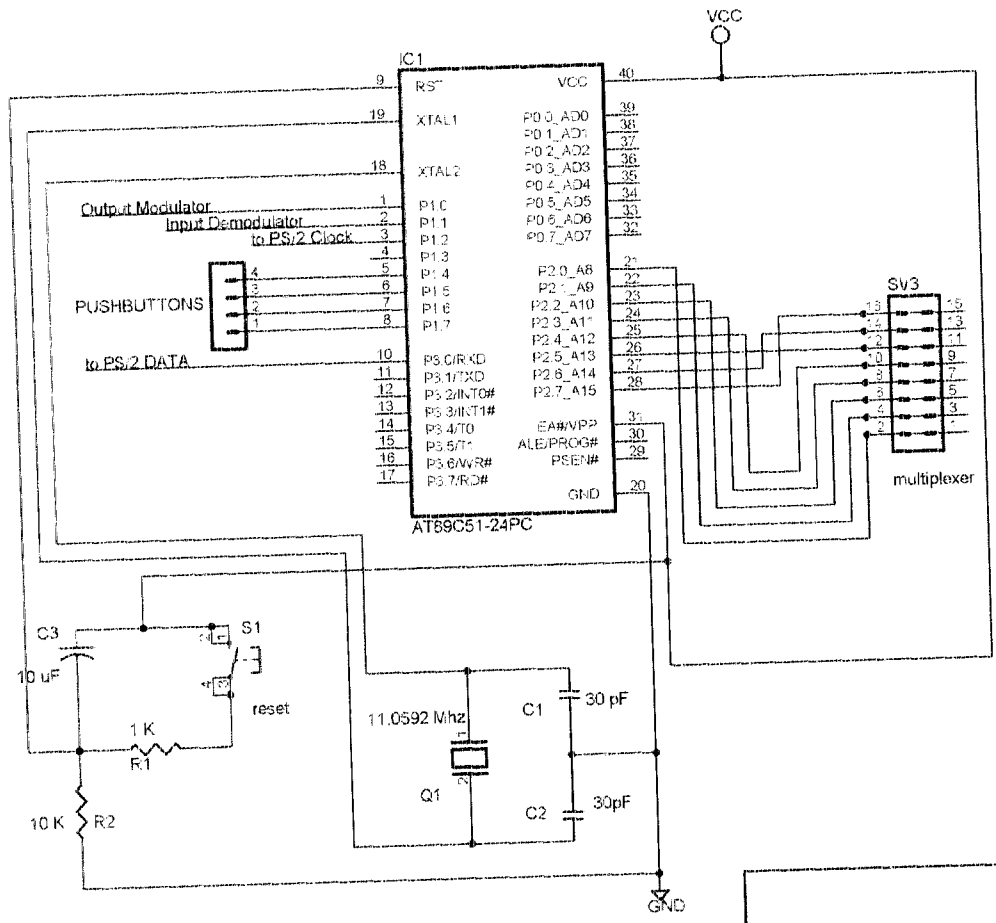
http://en.wikipedia.org/wiki/touchpad

LAMPIRAN

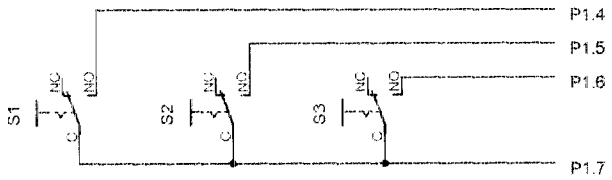


TITLE: Rangkaian Multiplexer IR LED	
Document Number:	REV:
Date: 8/03/2007 09:45:48a	Sheet: 1/1


Photodiode sumbu Y
Photodiode sumbu X
D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11
D12 D13 D14 D15 D16 D17 D18 D19 D20 D21 D22 D23 D24 D25 D26 D27 D28
JP1
to Photodiode out
VCC
B[0..7]
C[0..7]
D[0..7]
E[0..7]
IC1
IC2
IC3
IC4
4051N
A[0..2]
power
SV4
IC1P
IC2P
IC3P
IC4P
GND
P2.7
P2.6
P2.5
P2.4
P2.2
P2.1
P2.0
A2
A1
A0
TITLE: Rangkaian Multiplexer Photodiode
Document Number:
REV:
Date: 8/03/2007 09:50:00a
Sheet: 1/1



TITLE: Rangkaian Mikrokontroller dan Modulator	
Document Number:	REV:
Date: 8/03/2007 09:54:58a	Sheet: 1/1



TITLE: Rangkaian Pushbuttons	
Document Number:	REV:
Date: 8/03/2007 10:03:28a	Sheet: 1/1

```
CACAH                   EQU   -50000
CACAH1                  EQU   30
DDATA                   BIT   P3.0
DCLOCK                  BIT   P1.2
ACKNOWLEDGE             EQU   7BH
STAT_REQ_2              EQU   7AH
STAT_REQ_1              EQU   79H
MOUSE_ID                EQU   78H
SELF_TEST_PASSED        EQU   77H

SAVED_X0                EQU   7FH
SAVED_Y0                EQU   7EH
SAVED_X1                EQU   7DH
SAVED_Y1                EQU   7CH

LAST_PACKET_1           EQU   76H

SAVED_Yn                EQU   75H
SAVED_Xn                EQU   74H

BUFFER_X                EQU   73H
BUFFER_Y                EQU   72H

PACKET_1                EQU   2DH   ;SIGN_BIT_&_BUTTON_BIT
PACKET_2                EQU   2EH   ;MOVEMENT_X
PACKET_3                EQU   2FH   ;MOVEMENT_Y

OUTPUT                  BIT   P1.0  ;1=IR_ON, 0=IR_OFF
INPUT                   BIT   P1.1  ;1=OBJ_FOUND, 0=OBJ_NOT_FOUND
LEFT_BUTTON_INPUT       BIT   P1.4  ;0=LEFT_PRESSED,1=LEFT_NOPRESS
MIDDLE_BUTTON_INPUT     BIT   P1.5  ;0=RIGHT_PRESS,1=RIGHT_NOPRESS
RIGHT_BUTTON_INPUT      BIT   P1.6  ;0=MID_PRESS, 1=MID_NOPRESS
COMMON_BIT_BUTTON       BIT   P1.7

LEFT_BUTTON_BIT         BIT   68H   ;1=LEFT_PRESSED,0=LEFT_NOPRESS
RIGHT_BUTTON_BIT        BIT   69H   ;1=RIGHT_PRESS,0=RIGHT_NOPRESS
MIDDLE_BUTTON_BIT       BIT   6AH   ;1=MID_PRESS,0=MID_NOPRESS
ALWAYS_1_BIT            BIT   6BH
X_SIGN_BIT              BIT   6CH   ;1=LEFT, 0=RIGHT
Y_SIGN_BIT              BIT   6DH   ;1=DOWN, 0=UP
X_OVERFLOW_BIT          BIT   6EH
Y_OVERFLOW_BIT          BIT   6FH

FLAG_BUTTONS_PRESSED    BIT   60H   ;1=YES_PRESS,0=NO_NOPRESS
FLAG_SCAN_Y             BIT   61H   ;1=YES_OBJFOUND,0=NO_OBJ_NO
FLAG_SCAN_X             BIT   62H   ;1=NOT_FINISH, 0=FINISH
FLAG_CEK_MOVEMENT       BIT   63H   ;1=OBJ_MOVE, 0=OBJ_STOP
FLAG_COMPARE            BIT   64H   ;1=STILL, 0=MOVE


ORG     00H
SJMP    MULAI
;=================================================================

ORG     30H
MULAI:
```

```
        CLR     EA
        JMP     PREPARE_INITIALIZATION


;SCAN_Y_AXIS=============================================================;
;SET_R0_COUNTER                                                          ;
;IC_0=8x, IC_1=3x, TOTAL_Y=11                                            ;
;MODULATION ~40kHZ                                                       ;
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~;
SCAN_Y:
        MOV     A,#11110000B
        MOV     R0,#08H
        MOV     R1,#01H
READY_Y:
Y_IC_0:
        MOV     P2,A
        CLR     P2.7
        ACALL   MODULASI_Y
        JB      INPUT,HASIL_Y
        INC     A
        INC     R1
        DJNZ    R0,Y_IC_0
        MOV     R0,#03H
        MOV     A,#11110000B
Y_IC_1:
        MOV     P2,A
        CLR     P2.6
        ACALL   MODULASI_Y
        JB      INPUT,HASIL_Y
        INC     A
        INC     R1
        DJNZ    R0,Y_IC_1
        CLR     FLAG_SCAN_Y
RET

;SUB-MODULASI_Y^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^;
;-------------------------------------------------------------------------;
MODULASI_Y:
        MOV     R2,#35
LOOP_MODULASI_Y:
        DJNZ    R2,START_MODULASI_Y
        CLR     OUTPUT
        RET
START_MODULASI_Y:
        CLR     OUTPUT
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
```

```
        INC     R1
        DJNZ    R0,X_IC_3
        CLR     FLAG_SCAN_X
RET
;SUB-MODULASI_X^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^;
;-----------------------------------------------------------------------------;
MODULASI_X:
        MOV     R2,#15
LOOP_MODULASI_X:
        DJNZ    R2,START_MODULASI_X
        CLR     OUTPUT
        RET
START_MODULASI_X:
        CLR     OUTPUT
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        SETB    OUTPUT
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
        NOP
SJMP    LOOP_MODULASI_X
;SUB-HASIL_X^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^;
;-----------------------------------------------------------------------------;
                                                        ;SAVE_X_POSSTION
HASIL_X:
        SETB    FLAG_SCAN_X
RET
;#############################################################################;

;SCAN_BUTTONS_STATE==========================================================;
;LEFT_BUTTON => PRESSED=1, NOT_PRESSED=0                                      ;
;RIGHT_BUTTON => PRESSED=1, NOT_PRESSED=0                                     ;
;MIDDLE_BUTTON => PRESSED=1, NOT_PRESSED=0                                    ;
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~;
SCAN_BUTTONS:
        SETB    COMMON_BIT_BUTTON

        CLR     COMMON_BIT_BUTTON
        JNB     LEFT_BUTTON_INPUT,LEFT_BUTTON_PRESSED
```

```
        CLR         FLAG_BUTTONS_PRESSED              ;CLEAR_LEFT_BUTTON_BIT
        CLR         LEFT_BUTTON_BIT
        SETB        COMMON_BIT_BUTTON

        CLR         COMMON_BIT_BUTTON
        JNB         RIGHT_BUTTON_INPUT,RIGHT_BUTTON_PRESSED
        CLR         FLAG_BUTTONS_PRESSED              ;CLEAR_RIGHT_BUTTON_BIT
        CLR         RIGHT_BUTTON_BIT
        SETB        COMMON_BIT_BUTTON

        CLR         COMMON_BIT_BUTTON
        JNB         MIDDLE_BUTTON_INPUT,MIDDLE_BUTTON_PRESSED
        CLR         FLAG_BUTTONS_PRESSED              ;CLEAR_MIDDLE_BUTTON_BIT
        CLR         MIDDLE_BUTTON_BIT
        SETB        COMMON_BIT_BUTTON
RET
;SUB-BUTTON_PRESSED^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^;
;------------------------------------------------------------------------------;
LEFT_BUTTON_PRESSED:
        SETB        FLAG_BUTTONS_PRESSED
        SETB        LEFT_BUTTON_BIT
        RET
RIGHT_BUTTON_PRESSED:
        SETB        FLAG_BUTTONS_PRESSED
        SETB        RIGHT_BUTTON_BIT
        RET
MIDDLE_BUTTON_PRESSED:
        SETB        FLAG_BUTTONS_PRESSED
        SETB        MIDDLE_BUTTON_BIT
        RET
;###########################################################################;

;===========================================================================;
;                                                                           ;
;CALCULATE==================================================================;
;FIND_DIFFERENT_BETWEEN_X0-X1                                               ;
;FIND_DIFFERENT_BETWEEN_Y0-Y1                                               ;
;IF_Y_MOVEMENT=1,FAILED
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~;
CALCULATE:
X_MOVEMENT:
        MOV         A,SAVED_X1
        SUBB        A,SAVED_X0
        JNC         KANAN
        CLR         C
        SJMP        KIRI

KANAN:
        CLR         X_SIGN_BIT
        MOV         PACKET_2,A
        CJNE        A,#1,PASS_RIGHT
        MOV         PACKET_2,#0
        MOV         SAVED_X1,SAVED_X0
        SJMP        Y_MOVEMENT

PASS_RIGHT:
        MOV         B,#1
        MOV         A,PACKET_2
```

```
        MUL   AB
        MOV   PACKET_2,A
        SJMP  Y_MOVEMENT
KIRI:
        SETB  X_SIGN_BIT
        MOV   PACKET_2,A
        CJNE  A,#0FFH,PASS1_LEFT
FAILED:
        CLR   X_SIGN_BIT
        MOV   PACKET_2,#0
        MOV   SAVED_X1,SAVED_X0
        SJMP  Y_MOVEMENT
PASS1_LEFT:
        SJMP  Y_MOVEMENT
;-------------------------------------------
Y_MOVEMENT:
        MOV   A,SAVED_Y1
        SUBB  A,SAVED_Y0
        JNC   ATAS
        CLR   C
        SJMP  BAWAH
ATAS:
        CLR   Y_SIGN_BIT
        MOV   PACKET_3,A
        CJNE  A,#1,PASS_UP
        MOV   PACKET_3,#0
        MOV   SAVED_Y1,SAVED_Y0
        RET
PASS_UP:
        MOV   A,PACKET_3
        JZ    LEWAT
        DEC   A
        MOV   PACKET_3,A
LEWAT:
        MOV   B,PACKET_3
        MOV   A,PACKET_3
        MUL   AB
        MOV   PACKET_3,A
        RET
BAWAH:
        SETB  Y_SIGN_BIT
        MOV   PACKET_3,A
        CJNE  A,#0FFH,PASS_DOWN
        CLR   Y_SIGN_BIT
        MOV   PACKET_3,#0
        MOV   SAVED_Y1,SAVED_Y0
        RET
PASS_DOWN:
        MOV   A,#0
        SUBB  A,PACKET_3
        MOV   PACKET_3,A
        MOV   A,PACKET_3
        MOV   B,PACKET_3
        MUL   AB
        MOV   B,A
        MOV   A,#0
```

```
        SUBB    A,B
        CLR     C
        MOV     PACKET_3,A
        RET
;#########################################################################;

;DELAY=====================================================================;
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~;
DELAY:
        MOV     R4,#100
DELAY_1:
        MOV     R5,#0
        DJNZ    R5,$
        DJNZ    R4,DELAY_1

 RET
 ;#########################################################################;

 ;WAIT======================================================================;
 ;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~;
 WAIT:
        MOV     R6,#5
 WAIT_1:
        MOV     R4,#40
 WAIT_2:
        MOV     R5,#0
        DJNZ    R5,$
        DJNZ    R4,WAIT_2
        DJNZ    R6,WAIT_1

  RET
  ;#########################################################################;

  ;CEK_MOVEMENT=============================================================;
  ;SET_FLAG_CEK_MOVEMENT_BIT                                               ;
  ;1=OBJ_MOVE                                                              ;
  ;0=OBJ_STOP                                                              ;
  ;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~;
  CEK_MOVEMENT:
        MOV     A,PACKET_2
        JNZ     OBJ_MOVE
        MOV     A,PACKET_3
        JNZ     OBJ_MOVE

  OBJ_STOP:
        CLR     FLAG_CEK_MOVEMENT
        RET

  OBJ_MOVE:
        SETB    FLAG_CEK_MOVEMENT

  RET
  ;#########################################################################;

  ;SET_X0_AND_Y0=============================================================;
  ;X1->X0                                                                   ;
```

```
                                                                              ;
;Y1->Y0
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~;
SET_NEW_X0Y0:
SET_NEW_X0:
        MOV      SAVED_X0,SAVED_X1
SET_NEW_Y0:
      MOV      SAVED_Y0,SAVED_Y1
RET
;#############################################################################;

;COMPARE=====================================================================;
;X0<->Xn                                                                     ;
;Y0<->Yn
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~;
COMPARE:
       MOV    A,SAVED_Yn
       CJNE   A,#SAVED_Y0,PASS_COMPARE_1
       SJMP   COMPARE2
PASS_COMPARE_1:
       INC    A
       CJNE   A,#SAVED_Y0,PASS_COMPARE_2
       SJMP   COMPARE2
PASS_COMPARE_2:
       DEC    A
       DEC    A
       CJNE   A,#SAVED_Y0,PASS_COMPARE_3
       SJMP   COMPARE2
PASS_COMPARE_3:
       CLR    FLAG_COMPARE
       CLR    LEFT_BUTTON_BIT
       RET

COMPARE2:
       MOV    A,SAVED_Xn
       CJNE   A,#SAVED_X0,PASS_COMPARE2_1
       SETB   FLAG_COMPARE
       SETB   LEFT_BUTTON_BIT
       RET
PASS_COMPARE2_1:
       INC    A
       CJNE   A,#SAVED_X0,PASS_COMPARE2_2
       SETB   FLAG_COMPARE
       SETB   LEFT_BUTTON_BIT
       RET
PASS_COMPARE2_2:
       DEC    A
       DEC    A
       CJNE   A,#SAVED_X0,PASS_COMPARE_3
       SETB   FLAG_COMPARE
       SETB   LEFT_BUTTON_BIT
       RET
;#############################################################################;

;TOUCHPAD====================================================================;
                                                                             ;
;MAIN_PROGRAM
```

```
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~;
MAIN_PROGRAM:
        SETB    ALWAYS_1_BIT


START:

        ACALL   SCAN_BUTTONS
        JB      FLAG_BUTTONS_PRESSED,PRE_CASE_1_LINK
CASE_0:
        ACALL   SCAN_Y
        JNB     FLAG_SCAN_Y,CASE_00
        MOV     SAVED_Y0,R1
CASE_01:
        ACALL   SCAN_X
        JNB     FLAG_SCAN_X,START
        MOV     SAVED_X0,R1
CASE_02:
        ACALL   DELAY
CASE_02_R:
        ACALL   SCAN_Y
        JNB     FLAG_SCAN_Y,CASE_020
        MOV     SAVED_Y1,R1
CASE_03:
        ACALL   SCAN_X
        JNB     FLAG_SCAN_X,CASE_030
        MOV     SAVED_X1,R1
CASE_04:
        ACALL   CALCULATE
        ACALL   CEK_MOVEMENT
        JNB     FLAG_CEK_MOVEMENT,CASE_040
CASE_05:
        ACALL   SET_NEW_X0Y0
        ACALL   SCAN_BUTTONS
        JNB     FLAG_BUTTONS_PRESSED,CASE_050
CASE_06:
        ACALL   SEND_PACKET
        ACALL   DELAY
CASE_06_R:
        ACALL   SCAN_Y
        JNB     FLAG_SCAN_Y,CASE_060
        MOV     SAVED_Y1,R1
CASE_07:
        ACALL   SCAN_X
        JNB     FLAG_SCAN_X,CASE_070
        MOV     SAVED_X1,R1
CASE_08:
        ACALL   CALCULATE
        ACALL   SET_NEW_X0Y0
        ACALL   SCAN_BUTTONS
        JNB     FLAG_BUTTONS_PRESSED,CASE_080
CASE_09:
        SJMP    CASE_06
```

```
;^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^;
PRE_CASE_1_LINK:
      JMP    PRE_CASE_1
;-----------------------------------------------------------------;



CASE_050:
        SJMP     CASE_06

CASE_060:
        ACALL    SCAN_X
        JB       FLAG_SCAN_X,CASE_06_R
        ACALL    SCAN_BUTTONS
        JNB      FLAG_BUTTONS_PRESSED,CASE_0600
CASE_0601:
        MOV      PACKET_2,#0
        MOV      PACKET_3,#0
        CLR      X_SIGN_BIT
        CLR      Y_SIGN_BIT
        ACALL    SEND_PACKET
        ACALL    WAIT
        SJMP     START

CASE_0600:
        SJMP     CASE_0601

CASE_070:
        SJMP     START

CASE_080:
        SJMP     CASE_06



CASE_00:
        SJMP     START
CASE_010:
        ACALL    SCAN_BUTTONS
        JNB      FLAG_BUTTONS_PRESSED,CASE_0100_LINK
CASE_0101:
        SJMP     START

CASE_020:
        ACALL    SCAN_X
        JB       FLAG_SCAN_X,CASE_02_R
        SJMP     CASE_010



CASE_030:
        SJMP     CASE_010

CASE_040:
```

```
        ACALL   SCAN_BUTTONS
        JB      FLAG_BUTTONS_PRESSED,CASE_0401_LINK
CASE_0400:
        ACALL   DELAY
CASE_0400_R:
        ACALL   SCAN_Y
        JNB     FLAG_SCAN_Y,CASE_04000_LINK
        MOV     SAVED_Y1,R1
CASE_04001:
        ACALL   SCAN_X
        JB      FLAG_SCAN_X,CASE_04002_LINK
CASE_040010:
        SETB    LEFT BUTTON BIT
        ACALL   WAIT
CASE_040010_R:
        ACALL   SCAN_Y
        JNB     FLAG_SCAN_Y,CASE_0400100_LINK
        MOV     SAVED_Yn,SAVED_Y0
        MOV     SAVED_Y0,R1
CASE_0400101:
        ACALL   SCAN_X
        JNB     FLAG_SCAN_X,CASE_04001010_LINK
        MOV     SAVED_Xn,SAVED_X0
        MOV     SAVED_X0,R1
CASE_0400102:
        ACALL   DELAY
        ACALL   SCAN_BUTTONS
        JB      FLAG_BUTTONS_PRESSED,CASE_0400103_LINK
        ACALL   COMPARE
        JNB     FLAG_COMPARE,CASE_04001020
        SETB    LEFT_BUTTON_BIT
CASE_04001020:
        SETB    LEFT_BUTTON_BIT
        MOV     PACKET_2,#0
        MOV     PACKET_3,#0
        CLR     X_SIGN_BIT
        CLR     Y_SIGN_BIT
        ACALL   SEND_PACKET
        ACALL   SCAN_Y
        JNB     FLAG_SCAN_Y,CASE_040010200_LINK
        MOV     SAVED_Y1,R1
CASE_040010201:
        ACALL   SCAN_X
        JNB     FLAG_SCAN_X,CASE_0400102010_LINK
        MOV     SAVED_X1,R1
CASE_040010202:
        ACALL   CALCULATE
        ACALL   SEND_PACKET
        ACALL   SET_NEW_X0Y0
        SJMP    CASE_0400102


;^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^;
CASE_0100_LINK:
```

```
      JMP    CASE_0100
CASE_040010_R_LINK:
      JMP    CASE_040010_R
CASE_040010200_LINK:
      JMP    CASE_040010200
CASE_0400102010_LINK:
      JMP    CASE_0400102010
CASE_0400_R_LINK:
      JMP    CASE_0400_R
CASE_0401_LINK:
      JMP    CASE_0401
CASE_04000_LINK:
      JMP    CASE_04000
CASE_04002_LINK:
      JMP    CASE_04002
CASE_0400100_LINK:
      JMP    CASE_0400100
CASE_04001010_LINK:
      JMP    CASE_04001010
CASE_0400103_LINK:
      JMP    CASE_0400103
;-----------------------------------------------------------------;



CASE_0100:
        ACALL   WAIT
CASE_0100_R:
        ACALL   SCAN_Y
        JNB     FLAG_SCAN_Y,CASE_01000
        MOV     SAVED_Yn,SAVED_Y0
        MOV     SAVED_Y0,R1
CASE_01001:
        ACALL   SCAN_X
        JNB     FLAG_SCAN_X,CASE_010010
        MOV     SAVED_Xn,SAVED_X0
        MOV     SAVED_X0,R1
CASE_01002:
        ACALL   COMPARE
        JNB     FLAG_COMPARE,PASS_01002
        SETB    LEFT_BUTTON_BIT
        MOV     PACKET_2,#0
        MOV     PACKET_3,#0
        CLR     X_SIGN_BIT
        CLR     Y_SIGN_BIT
        ACALL   SEND_PACKET
PASS_01002:
        SJMP    CASE_0400102

CASE_01000:
        ACALL   SCAN_X
        JB      FLAG_SCAN_X,CASE_0100_R
        SETB    LEFT_BUTTON_BIT
CASE_01000_F:
        MOV     PACKET_2,#0
```

```
        MOV     PACKET_3,#0
        CLR     X_SIGN_BIT
        CLR     Y_SIGN_BIT
        ACALL   SEND_PACKET
        ACALL   DELAY
        CLR     LEFT_BUTTON_BIT
        ACALL   SEND_PACKET
        LJMP    START


CASE_010010:
        SJMP    CASE_01000_F

CASE_0401:
        JMP     START

CASE_04000:
        ACALL   SCAN_X
        JB      FLAG_SCAN_X,CASE_0400_R_LINK
        JMP     CASE_040010

CASE_04002:
        MOV     SAVED_X1,R1
        JMP     CASE_04

CASE_0400100:
        ACALL   SCAN_X
        JB      FLAG_SCAN_X,CASE_040010_R_LINK
CASE_0400100_F:
        CLR     LEFT_BUTTON_BIT
        MOV     PACKET_2,#0
        MOV     PACKET_3,#0
        CLR     X_SIGN_BIT
        CLR     Y_SIGN_BIT
        ACALL   SEND_PACKET
        JMP     START

CASE_04001010:
        ACALL   SCAN_Y
        JNB     FLAG_SCAN_Y,CASE_0400100_F
        MOV     SAVED_Y0,R1
        JMP     CASE_0400101

CASE_0400103:
        CLR     LEFT_BUTTON_BIT
        MOV     PACKET_2,#0
        MOV     PACKET_3,#0
        CLR     X_SIGN_BIT
        CLR     Y_SIGN_BIT
        ACALL   SEND_PACKET
        ACALL   WAIT
        ACALL   WAIT
        JMP     START
```

```
;=========================================================================
===
CASE_04001020_LINK:
      JMP    CASE_04001020
;=========================================================================
===

CASE_040010200:
      ACALL SCAN_X
      JB    FLAG_SCAN_X,CASE_04001020_LINK
CASE_040010200_F:
      ACALL SCAN_BUTTONS
      JB    FLAG_BUTTONS_PRESSED,CASE_0400102001
      SETB  LEFT_BUTTON_BIT
CASE_0400102000:
      DJNZ  R7,CASE_0400102000_FF
      CLR   LEFT_BUTTON_BIT
      MOV   PACKET_2,#0
      MOV   PACKET_3,#0
      CLR   X_SIGN_BIT
      CLR   Y_SIGN_BIT
      ACALL SEND_PACKET
      JMP   START
CASE_0400102000_FF:
       ACALL   SCAN_Y
       JNB     FLAG_SCAN_Y,CASE_04001020000
       MOV     SAVED_Y0,R1
CASE_04001020001:
       ACALL   SCAN_X
       JNB     FLAG_SCAN_X,CASE_040010200010
       MOV     SAVED_X0,R1
CASE_04001020002:
       ACALL   DELAY
CASE_04001020002_R:
       ACALL   SCAN_Y
       JNB     FLAG_SCAN_Y,CASE_040010200020
       MOV     SAVED_Y1,R1
CASE_04001020003:
       ACALL   SCAN_X
       JNB     FLAG_SCAN_X,CASE_040010200030
       MOV     SAVED_X1,R1
CASE_04001020004:
       ACALL   CALCULATE
       ACALL   SEND_PACKET
       MOV     R7,#100
       ACALL   SET_NEW_X0Y0
       ACALL   SCAN_BUTTONS
       JB      FLAG_BUTTONS_PRESSED,CASE_04001020005
       SETB    LEFT_BUTTON_BIT
       SJMP    CASE_04001020002


CASE_0400102010:
       SJMP    CASE_040010200_F
```

```
CASE_0400102001:
        CLR     LEFT_BUTTON_BIT
        MOV     PACKET_2,#0
        MOV     PACKET_3,#0
        CLR     X_SIGN_BIT
        CLR     Y_SIGN_BIT
        ACALL   SEND_PACKET
        ACALL   WAIT
        JMP     START


CASE_04001020000:
        ACALL   SCAN_X
        JB      FLAG_SCAN_X,CASE_0400102000
        SJMP    CASE_040010200_F

CASE_0400102000010:
        SJMP    CASE_040010200_F

CASE_040010200020:
        ACALL   SCAN_X
        JB      FLAG_SCAN_X,CASE_04001020002_R
        SJMP    CASE_040010200_F

CASE_040010200030:
        SJMP    CASE_040010200_F

CASE_04001020005:
        SJMP    CASE_0400102001


PRE_CASE_1:
      MOV   PACKET_2,#0
      MOV   PACKET_3,#0
      CLR   X_SIGN_BIT
      CLR   Y_SIGN_BIT
      ACALL SEND_PACKET
CASE_1:
        ACALL   SCAN_Y
        JNB     FLAG_SCAN_Y,CASE_10
        MOV     SAVED_Y0,R1
CASE_2:
        ACALL   SCAN_X
        JNB     FLAG_SCAN_X,CASE_20
        MOV     SAVED_X0,R1
CASE_3:
        ACALL   DELAY
CASE_3_R:
        ACALL   SCAN_Y
        JNB     FLAG_SCAN_Y,CASE_30
        MOV     SAVED_Y1,R1
CASE_4:
        ACALL   SCAN_X
        JNB     FLAG_SCAN_X,CASE_40
```

```
        MOV     SAVED_X1,R1
CASE_5:
        ACALL   CALCULATE
        ACALL   SCAN_BUTTONS
        JNB     FLAG_BUTTONS_PRESSED,CASE_50
CASE_6:
        ACALL   SEND_PACKET
        ACALL   SET_NEW_X0Y0
        SJMP    CASE_3



CASE_10:
        ACALL   SCAN_BUTTONS
        JNB     FLAG_BUTTONS_PRESSED,CASE_100
CASE_101:
        SJMP    CASE_1

CASE_100:
        MOV     PACKET_2,#0
        MOV     PACKET_3,#0
        CLR     X_SIGN_BIT
        CLR     Y_SIGN_BIT
        ACALL   SEND_PACKET
        JMP     START

CASE_20:
      SJMP  CASE_1



CASE_30:
      ACALL SCAN_X
      JB    FLAG_SCAN_X,CASE_3_R
      SJMP  CASE_10

CASE_40:
      SJMP  CASE_10

CASE_50:
      SJMP  CASE_6


;SEND_PACKET=================================================;
;SEND_PACKET_1                                               ;
;SEND_PACKET_2                                               ;
;SEND_PACKET_3                                               ;
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~;
SEND_PACKET:
      MOV   A,PACKET_1
      CJNE  A,LAST_PACKET_1,GO_SEND
      MOV   A,PACKET_2
      CJNE  A,#0,GO_SEND
      MOV   A,PACKET_3
      CJNE  A,#0,GO_SEND
      RET
```

```
GO_SEND:
        JNB     Y_SIGN_BIT,BERIKUTNYA
        MOV     A,#0
        SUBB    A,PACKET_3
        MOV     BUFFER_Y,A
        SJMP    BERIKUTNYA2
BERIKUTNYA:
        MOV     BUFFER_Y,PACKET_3
BERIKUTNYA2:
        JNB     X_SIGN_BIT,LANGSUNG
        MOV     A,#0
        SUBB    A,PACKET_2
        MOV     BUFFER_X,A
        MOV     A,BUFFER_X
        SUBB    A,BUFFER_Y
        JNC     REPEAT_BUFFER_X
        MOV     R7,BUFFER_Y
        SJMP    SEND_NOW

REPEAT_BUFFER_X:
        MOV     R7,BUFFER_X
        SJMP    SEND_NOW

LANGSUNG:
        MOV     A,PACKET_2
        SUBB    A,BUFFER_Y
        JNC     REPEAT_PACKET_2
        MOV     R7,BUFFER_Y
        SJMP    SEND_NOW

REPEAT_PACKET_2:
        MOV     R7,PACKET_2

SEND_NOW:
        MOV     R1,#2DH                 ;2DH->POINTER_PACKET_ADDRESS
        CLR     A
        ACALL   KIRIM
        MOV     LAST_PACKET_1,PACKET_1

        INC     R1
        CLR     A
        ACALL   KIRIM
        INC     R1
        CLR     A
        ACALL   KIRIM

        CJNE    R7,#0,MORE
        RET
MORE:
        ACALL   DELAY_2
        MOV     R1,#2DH
        CLR     A
        ACALL   KIRIM
        INC     R1
        CLR     A
```

```
        ACALL KIRIM
        INC   R1
        CLR   A
        ACALL KIRIM
        DJNZ  R7,MORE
RET
;#####################################################################;

;CREATED BY==========================================================;
;RAHAN S.Y                                                           ;
;03524038                                                            ;
;T. ELEKTRO UII                                                      ;
;--------------------------------------------------------------------;

;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++;

;POWER-ON RESET======================================================;
;TUNGGU ~500mS                                                       ;
;SETELAH START-UP                                                    ;
;COMPUTER                                                            ;
;--------------------------------------------------------------------;

PREPARE_INITIALIZATION:
        MOV     TMOD,#01H

ULANG:
        MOV     R0,#0AH

LAGI:
        MOV     TL0,#LOW CACAH
        MOV     TH0,#HIGH CACAH
        SETB    TR0

TUNGGU:
        JNB     TF0,$
        ACALL   RESET_TIMER
        DJNZ    R0,LAGI
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++;

;KIRIM INISIALISASI BYTE=============================================;
;SELF TEST PASSED                                                    ;
;ID DEVICE                                                           ;
;ACKNOWLEDGE                                                         ;
;--------------------------------------------------------------------;

        MOV     R1,#77H                         ;POINTER
        MOV     SELF_TEST_PASSED,#0AAH          ;SELF TEST PASSED
        MOV     MOUSE_ID,#00H                   ;MOUSE ID
        MOV     STAT_REQ_1,#02H                 ;STAT REQ
        MOV     STAT_REQ_2,#064H                ;STAT REQ
        MOV     ACKNOWLEDGE,#0FAH               ;ACKNOWLEDGE

CEK_CONNECT:
        JNB     DCLOCK,$
        JNB     DDATA,$
```

```
        ACALL   KIRIM

        INC     R1
        ACALL   KIRIM

SCAN:
        ACALL   TERIMA
CEK_FF:
        CJNE    R2,#0FFH,CEK_GET_ID
        MOV     R1,#7BH                         ;0xFA
        ACALL   KIRIM
        MOV     R1,#77H                         ;0xAA
        ACALL   KIRIM
        INC     R1                              ;0x00
        ACALL   KIRIM
        SJMP    SCAN
CEK_GET_ID:
        CJNE    R2,#0F2H,CEK_STAT_REQ
        MOV     R1,#7BH                         ;0xFA
        ACALL   KIRIM
        MOV     R1,#78H                         ;0x00
        ACALL   KIRIM
        SJMP    SCAN
CEK_STAT_REQ:
        CJNE    R2,#0E9H,CEK_DAN_LAIN2
        MOV     R1,#7BH                         ;0xFA
        ACALL   KIRIM
        MOV     R1,#78H                         ;0x00
        ACALL   KIRIM
        INC     R1                              ;0x02
        ACALL   KIRIM
        INC     R1                              ;0x64
        ACALL   KIRIM
        SJMP    SCAN
CEK_DAN_LAIN2:
        MOV     R1,#7BH                         ;0xFA
        ACALL   KIRIM
CEK_ENABLE:
        CJNE    R2,#0F4H,SCAN
        JMP     MAIN_PROGRAM
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++;

;MENGIRIMKAN BYTE=================================================;
;FORMAT                                                           ;
;- 1 BIT START = LOW (0)                                          ;
;- 8 BITS DATA = LOW/HIGH (0/1)                                   ;
;- 1 BIT PARITY = LOW (0) JIKA GANJIL                             ;
;- 1 BIT STOP = HIGH (1)                                          ;
;-----------------------------------------------------------------;

KIRIM:

CEK_CLOCK:
        JNB     DCLOCK,$
        ACALL   GO_TIMER
        JNB     DCLOCK,CEK_CLOCK
```

```
CEK_DATA_LINE:
        JNB     DDATA,SCAN

SEND_START_BIT:
        CLR     DDATA
        ACALL   GO_TIMER
        CLR     DCLOCK
        ACALL   GO_TIMER

PREP_DATA:
        MOV     A,@R1
        MOV     C,P
        CPL     C
        MOV     00H,C                                   ;BIT PARITI
        SETB    01H                                     ;BIT STOP
        MOV     R0,#08H

SEND_DATA:
        ACALL   SEND
        DJNZ    R0,SEND_DATA
        CLR     A

SEND_PARITI:
        MOV     R0,#02H
        MOV     A,20H

SEND_STOP:
        ACALL   SEND
        DJNZ    R0,SEND_STOP
        CLR     A
RET
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++;

;SUB-RUTIN CALL=================================================;
;- SEND                                                         ;
;- SET_TIMER                                                    ;
;- RESET_TIMER                                                  ;
;- GO_TIMER                                                     ;
;---------------------------------------------------------------;
SEND:
        SETB    DCLOCK
        MOV     P3,A
        ACALL   GO_TIMER
        CLR     DCLOCK
        ACALL   GO_TIMER
        SETB    DCLOCK
        RR      A
RET

SET_TIMER:
      MOV     TMOD,#01H
        MOV     TL0,#LOW CACAH1
        MOV     TH0,#HIGH CACAH1
        SETB    TR0
        JNB     TF0,$
      CLR   TR0
```

```
        CLR    TF0
RET

RESET_TIMER:
        CLR     TR0
        CLR     TF0
RET

GO_TIMER:
        ACALL   SET_TIMER
        ACALL   RESET_TIMER
RET
;++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++;

;TERIMA BYTE=============================================================;
;FORMAT                                                                   ;
;- 1 BIT START = LOW (0) SAAT CLOCK = LOW (0)                             ;
;- 8 BITS DATA = LOW/HIGH (0/1)                                          ;
;- 1 BIT PARITI = LOW (0) JIKA GANJIL                                    ;
;- 1 BIT STOP = HIGH (1)                                                 ;
;- ACKNOWLEDGE BIT                                                       ;
;-------------------------------------------------------------------------;

GO_TO_SCAN:
        JMP       SCAN


TERIMA:

CEK_CLOCK_TERIMA:
        JB      DCLOCK,$
        JB      DDATA,$
        JNB     DCLOCK,$
        ACALL   GO_TIMER

AMBIL_DATA:
        CLR     DCLOCK
        MOV     R0,#08H
        ACALL   RECEIVE
        MOV     R2,A
        CLR     A
RET

RECEIVE:
        ACALL   GO_TIMER
        SETB    DCLOCK
        MOV     C,DDATA
        RR      A
        MOV     ACC.7,C
        ACALL   GO_TIMER
        CLR     DCLOCK
        DJNZ    R0,RECEIVE
        MOV     R0,#02H

RECEIVE_PARITI:
        CLR     DCLOCK
```

```
            ACALL   GO_TIMER
            SETB    DCLOCK
            ACALL   GO_TIMER                                ;STOP BIT
            CLR     DCLOCK
            ACALL   GO_TIMER
            SETB    DCLOCK
            CLR     DDATA
            ACALL   GO_TIMER                                ;ACK
            CLR     DCLOCK
            ACALL   GO_TIMER
            SETB    DCLOCK
            SETB    DDATA
RET
;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++;

DELAY_2:
      MOV   R4,#10
DELAY_3:
      MOV   R5,#0
      DJNZ  R5,$
      DJNZ  R4,DELAY_3
RET

END
```