Studi Kasus Interface dengan Pemrograman Port Paralel dan Serial Menggunakan C

Oleh:

Nama : I Made Joni, M.Sc.

NIP : 132296651

Laboratorium Instrumentasi Elektronika

Jurusan Fisika FMIPA

Fakultas Matematika dan Ilmu Pengetahuan Alam

Universitas Padjadjaran

2005

Daftar Isi

1. PENDAHULUAN

Jurang kesenjangan sering terjadi antara pengausaan pemrograman dan pengetahuan tentang piranti elektronik yang dihubungkan ke komputer. Hal ini mengakibatkan terjadinya kesulitan dalam menggunakan teknik pemrograman untuk mengakses perangkat keras. Port programming adalah suatu perintah program yang digunakan untuk membaca sutau register yang ditugaskan spesifik untuk membaca piranti elektronik tertentu yang dihubungkan ke personal computer (PC), dalah hal ini akan diacu menggunakan IBM-PC. Kendala yang sering terjadi adalah sedikitnya pengalaman langsung dalam pendesainan dan pembuatan piranti elektronika sederhana sebagai media awal yang akan digunakan sebagai alat bantu dalam melatih kemampuan port programming. Agar Anda dapat menguasai teknik pemrograman port ini, Anda perlu pengalaman praktis dalam bentuk proyek kecil pembuatan perangkat lunak untuk mengakses port.

Tumbuh pesatnya perkembangan device-device yang mampu dihubungkan ke komputer mengakibatkan sebagian orang melihatnya sebagai suatu kerumitan. Perusahaan pemroduksi device biasanya sudah menyediakan driver atau suatu paket software yang mampu mendeteksi dan memfasilitasi pembacaan perangkat keras yang dipasangkan ke PC yang ditawarkannya baik untuk multimedia, aplikasi kontrol industri maupun untuk alat pengukuran saintifik. Akan tetapi tidak semua menyediakan driver, kadang kita hanya diketahui standar kominikasi yang digunakannya. Oleh kerena itu port programing sangat perlu digunakan. Aplikasi Port programing sangat luas seperti pada robotik, industri, instrumentasi dan sistem kontrol. Demikian luasnya perkembangan pemanfaatan PC dan juga interaksi dengan lingkungan di luar PC membuat motivasi dan ketertarikan tersendiri untuk menekuninya.

Berdasarkan pertimbangan inilah penulisan buku ini akan meuyertakan port programing yang akan memberikan strategi awal yang ditindaklanjuti dengan suatu proyek sederhana seperti penampilan led, hingga yang agak rumit seprti proses pengiriman karakter ke LCD. Pekerjaan-pekerjaan sederhana semacam ini pula telah dicobakan di Instrumentasi Elektronika Jurusan Fisika Universitas Padjadajran dalam rangka mengembangkan ketertertarikan awal terhadap bidang ini.

Pembahasan Bab ini akan diawali dengan pengantar teori tentang komunikasi paralel dengan standar komunikasi data IEEE 1284. Selanjutnya kita akan mempelajari juga bagaimana medesain modul baik harware maupun port programming yang akan dibahas dengan dua penekanan katagori proses yaitu pegiriman data melalui I/O ke luar komputer dan pengambilan data dari luar ke komputer. Untuk pengiriman data ari koputer ke luar akan dimulai dari yang paling sederhana semisal menyalakan lampu led melalui port LPT, menggunakan port untuk membuat musik, mengatur gerak motor dc & stepper hingaa ke yang sedikit komplek yaitu pengiriman data ke LCD matrik 2x16.

2. I/O PC

Kita perlu untuk berkomunikasi dengan komputer dan sebaliknya komputer perlu berkomunikasi dengan kita dan apapun yang dihubungkan ke komputer. Secara konsep, perpindahan sinyal listrik ke perangkat komputer disebut *input* dan pergerakan sinyal dari komputer ke perangkat luar disebut *output*. Kedua peroses ini biasanya dibahas secara

bersamaan yang secara sederhananya disebut sebagai input/output (I/O). Perangkat elektronik yang ada di dalam komputer dan berkomunikasi dengan perangkat lain yang masih ada di dalam komputer harus menggunakan system I/O. Sistem I/O juga digunakan jika Anda berkomunikasi dengan komputer melalui keyboard dan mouse atau komputer berkomunikasi dengan kita melalui monitor dan printer.

Menghubungkan dua perangkat elektronik artinya memindahkan sinyal listrik antara dua perangkat tersebut. Walaupun kelihatannya cukup sederhana, hanya dengan menghubungkan dua perangkat secara elektronik tidaklah cukup untuk melaksanakan proses I/O. Untuk itu perangkat yang dihubungankan hendaknya mempunyai kecocokan dalam hal bagaimana cara sinyal listrik tersebut dikomunikasikan, diproses dan digunakan. Bayangkan seandainya Anda ingin menekan huruf "M" pada keyboard ternyata komputer berfikir sinyal ini untuk "L". Kesalahan seperti ini tidak boleh terjadi, oleh karena itu peoses I/O adalah hal yang sangat penting dalam teknologi komputer. Hal lain yang juga menjadi perhatian dalam I/O adalah kecepatan dalam mentransfer data.

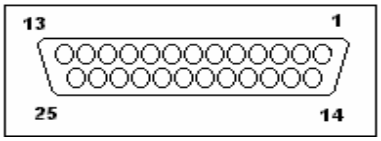
2.1. Mengenal Tarnsfer Data pada Port Paralel

Printer dihubungkan ke komputer melalui kabel 25 pin yang dihubungkan dengan konektor tipe betina DB-25 yang sering disebut sebagai metode pengkabelan *centronics* yang dapat dilihat pada Gambar 1 dan Tabel 1. Kabel ini mempunyai 8 pin data, 8 pin ground, 5 pin status dan 9 pin kontrol. Sekarang mari kita pelajari secara sederhana bagimana komputer berkomunikasi dengan printer. Salah satu kanal yaitu pin 13 membawa sinyal dari printer yang mengatakan kepada komputer bahwa printer *online* dan siap untuk menerima data. Jika komputer telah siap dengan pekerjaan yang akan diprint, komputer mengirimkan data melalui pin 2 sampai dengan pin 9 dalam bentuk tegangan. Tegangan 5 Volt merepresentasikan biner 1 dan kalau tegangannya rendah mendekati nol merepresentasikan biner 0. Setelah data siap, selanjutnya komputer mengirimkan sinyal *strobe* melalui pin 1 untuk memberi tahu printer membaca tegangan yang ada di pin 2 sampai dengan 9. Sinyal strobe ini biasanya berakhir dalam 1 mikrodetik. Jika prose berjalan lancar, maka printer membaca tegangan-tegangan pada pin 2-9 dan menterjemakanya kembali menjadi bit-bit. Printer selanjutnya memberi informasi ke komputer melalui pin 10 bahwa data telah diterima dan siap untuk pengiriman data selanjutnya. Proses ini dapat berulang ratusan bahkan ribuan kali per detik. Jika kesalahan terjadi maka printer akan memberi tahu komputer dengan mengirimkan sinyal error melalui pin tertentu.

Data dari motherboard yang dipindahkan menuju printer adalah 32 bit, sedangkan jalur data yang tersedia pada port adalah 8 bit. Perangkat yang khusus yang disebut *Port Parallel* menagani data ini dan memecahkanya menjadi empat potongan 8 bit. Port Parallel ini bertangung jawab terhadap urutan data dan membuat sinyal listrik yang kemudian ditransmisikannya ke kabel paralel. Port Paralel juga bertangung jawab menerima dan mengirimkan sinyal yang diberikan oleh printer. Misalnya jika printer kehabisan kertas, maka printer akan mengirimkan sinyal melalui pin 12 dan port paralel yang memantau pin 12 ini akan bereaksi dengan menghentikan pengiriman data dan melaporkan terjadinya kondisi *error* ini.

Awalnya printer I/O didesain dalam satu arah yang artinya data selalu dikirm ke printer dan tidak pernah printer mengirimkan informasi ke komputer. Dengan berkembangnya teknologi printer, scanner dan yang lainya yang terhubung melalui prot paralle memerlukan transfer data dari piranti luar ke komputer. Untuk menangani operasi I/0

ini dibuatlah transfer data port parallel dwi-arah. Oleh karena itu, teknologi pendesainan printer juga harus memungkinkan adanya transfer data ke komputer dan komputer juga harus mampu membaca data ini yang dikirim melalu kanal tertentu.
Pada tahun 1991 diadakanlah suatu pertemuan antara pembuat printer seperti texas Instrument, IBM, Lexmark dan juga yang lainnya. Hasil dari pertemuan ini adalah adanya kesepakatan untuk membentuk Network Printing Allience (NPA). NPA ini kemudian bekerja dalam menetapkan aturan-aturan yang harus diikuti dalam pembuatan perangkat keras agar tidak terjadi ketidakkompatibelan antara berbagai peralatan yag berbeda. Agar lebih diakui masyarakat luas dalam penggunaan aturan-atran ini, NPA mengajukannya ke Institue of electric and Electronic Engineer(IEEE) yang akhirnya disetujui sebagai keputusan IEEE 1284 pada tahun 1994. Pada IEEE 1284 ini terdapat metode pensinyalan standar antarmuka parallel dwi-arah. Standar IEEE 1284 ini masih kompatibel dengan port parallel sebelumnya. Standar ini mendefinsikan lima mode operasi antara lain *Compatibility Mode*, *Nibble Mode*, *Byte Mode*, *EPP Mode*, dan *ECP Mode*. Pada buku ini hanya akan disinggung sedikit mengenai mekanisme mode operasi EPP dan ECP. Jika Anda membutuhkan keterangan lebih lengkap tentang port parallel ini silahkan mengunjungi situs http://www.senet.com.au/~cpeacock yang membahas tentang *Interfacing The Standard Parallel Port*. Selain menambahkan komunikasi dwi-arah, port yang diteteapkan pada standar ini juga mampu menangani transfer data hingga 1 Mega Bit per detik dan memungkinkan panjang kabel hingga maksimum 10 meter.



Gambar 1. DB25 sebagai konektor port LPT parallel.

Tabel 1. Pin-pin pada konektor DB-25 dan Fungsi Registernya.

No Pin DB-25	No Pin Mode Centronic	Fungsi Register
1	1	Kontrol
2	2	Data
3	3	Data
4	4	Data
5	5	Data
6	6	Data
7	7	Data
8	8	Data
9	9	Data
10	10	Status
11	11	Status
12	12	Status
13	13	Status
14	14	Kontrol
15	32	Status
16	31	Kontrol
17	36	Kontrol
18-25	19-30	(Ground)

Dengan berkembangnya teknologi perangkat lunak yang memudahkan pemrosesan, pengolahan dan pembantu analisa data merangsang setiap orang untuk menghubungkan perangkat kerasnya ke komputer. Oleh karena itu, port paralel tidak hanya digunakan untuk menghubungkan komputer ke printer, tetapi juga biasa digunakan untuk menghubungkan atau sering disebut mengatramuka suatu alat atau perangkat keras lain ke komputer. Perangka keras ini biasanya berupa desain sendiri dengan tujuan desain dan sistem tersendiri atau alat yang dibeli dari pabrik yang telah memiliki spesifikasi standar IEEE 1284. Tentu saja port parallel bukannya satu-satunya cara, tetapi bisa juga digunakan melalui kompunikasi serial seperti RS232 atau USB.

2.2. Standar Komunikasi Paralel Menurut IEEE 1284

Keputusan IEEE 1284 mengeluarkan lima mode operasi yaitu *Compatibility Mode*, *Nibble Mode*, *Byte Mode*, *EPP Mode*, *ECP Mode*.Pembuatan standar ini bertujuan untuk mendesain driver dan device baru yang *compatible* satu dengan yang lainnya dan juga dengan Standard Parallel Port (SPP). Mode Compatibility, Nibble & Byte merupakan standar suatu perangkat keras yang aslinya tersedia dalam kartu port paralel, sedangkan mode EPP dan ECP membutuhkan perangkat keras tambahan yang memungkinkan untuk melakukan pekerjaanya lebih cepat. Transfer data dengan cara *Compatibility Mode* yang sering disebut mode *Centronics* hanya dapat mengirimkan dengan kecepatan sekitar 50 kbytes per detik dan bisa juga sampai maksimum dengan kecepatan 150 kbytes per detik. Kalau kita ingin computer mampu menerima data maka modenya harus diubah ke mode Nibble ataupun Byte. Mode Nibble dapat diberi masukan satu nibble *(4 bits)* dari suatu sistem rangkain di luar computer. Mode Byte menggunakan fitur parallel dwi-arah untuk memasukkan data satu byte *(8 bits)* data ke komputer.
Extended Parallel Ports (EPP) dan Enhanced Parallel Ports (ECP) menggunakan tambahan perangkat keras untuk membuat dan mengatur *handshaking*.
Berikut ini akan diberikan contoh penggunaan mode-mode di atas pada alat yang sering kita gunakan setiap hari yaitu printer seperti yang dicontohkan diawal pembahasan ini. Langkah-langkah atau algoritma pemrograman menggunakan compatibility mode untuk mengirim atau mengeluarkan data satu byte ke printer yang bisa mengandung infromasi text, gambar atau apa saja adalah:

1. Menulis atau mengrim satu byte ke port data
2. Mengecek apakah printer lagi sibuk melalui pin 11. Jika printer lagi sibuk tidak akan menerima data sehingga data apapun yang dikirimkan akan hilang
3. *Memberi nibel rendah(Strobe) ke Pin 1 untuk memberitahu printer bahwa ada data yang valid pada pin 2-9.*
4. *Setelah menunggu 5 mikro detik dari saat memberi strobe nibel rendah, beri strobe atau pin 1 tinggi.*

Langkah-langkah perulangan semacam ini akan mebatasi kecepatan kerja port yang semestinya. Port EPP dan ECP mengatasi hal ini dengan membiarkan hardware mengecek apakah printer sibuk dan kemudian mengirim suatu strobe atau handshaking yang sesui dengan keadaannya. Artinya hanya satu intruksi I/O yang dibutuhkan untuk melakukan pengriman data tersebut dan ini akan menigkatkan kecepatan kerja port. Port-port ini dapat mengeluarkan data sekitar 1-2 megabytes per detik. Disamping itu keuntungan lain

penggunaan Port ECP adalah penggunaan pin-pin DMA dan buffer FIFO yang memungkinkan data dipindahkan tanpa menggunakan instruksi I/O. Peran ECP semacam ini tidak akan dibahas dalam buku ini silahkan kunjungi situs yang telah diberikan di atas.

Didalam sistem komputer IBM-PC terdapat port-port standar yang dapat digunakan sebagai fasilitas input output (I/O) yang dapat menghubungkan komputer yang dalam bahasan ini akan menekankan pemakaian IBM-PC dengan piranti luar. Port paralel LPT adalah port standar, umumnya terdiri dari 25 pin yang digunakan untuk proses komunikasi data antara IBM-PC dengan piranti luar. Port ini memungkinkan kita memiliki masukan hingga 8 bit atau keluaran hingga 12 bit pada saat yang bersamaan, dengan hanya membutuhkan rangkaian eksternal sederhana untuk melakuakn suatu tugas tertentu.

2.3. Pengalamatan Port LPT Paralel

Pada umumnya port paralel memiliki tiga alamat fisik yang bisa digunakan, sebagaimana ditunjukkan pada Tabel 2. Alamat fisik yang digunakan untuk keperluan ini disebut juga *register*, sehingga dikenal ada register data, register status dan register kontrol. Bagi para programer pengertian register ini sangat penting karena sebenarnya hanya dengan bermain pada data di register ini. Alamat register 3BCH pertama kali diperkenalkan sebagai alamat port paralel pada kartu-kartu video lama, tapi kemudian alamat ini sempat menghilang ketika port paralel dicabut dari kartu-kartu video. LPT1 biasanya memiliki *alamat dasar* 378h, sedangkan LPT2 pada 278h. Alamat dasar tersebut sudah standar untuk setiap komputer, namun tidak menutup kemungkinan pada komputer satu dan komputer lainnya bisa saja berlainan.

Pada port parallel terdapat 17 pin yang digunakan sebagai jalur I/O. Komputer memiliki Register dengan 8 bit data per alamat, untuk menyesuaikan dengan kebutuhan komunikasi atau hanshaking maka dibutuhkan 3 alamat fisik untuk setiap alamat port parallel. Kebutuhan Register yang dikmasud pada proses hanshaking adalah Satu Register untuk kirim/terima Data, Satu Register Status untuk mengetahu keadaan piranti luar, dan satu Register untuk melakukan aksi terhadap status tertentu pada piranti luar. Bit-bit pada register itu tidak semuanya dipakai, jumlah regsiter yang dipakai Register Data (8 bit), Register Status (5 bit) dan Kontrol (4 bit). Alamat register tersebut berurutan, yakni jika alamat *dasar* untuk LPT1 adalah 0378h, maka alamat 0378 untuk Register Data, 0379 untuk Register Status dan 037A untuk Register Kontrol. Alamat demikian disebut juga offset yaitu jarak dari alamat awal. Register status hanya memakai 5 bit dari 8 bit yang ada, register kontrol hanya menggunakan 4 bit dari 8 bit, sedangkan register data menggunakan semua bit yang ada. Selengkapnya pemakaian alamat 8 bit untuk masing-masing register terdapat ada Tabel 3-5.

Tabel 2. Table almat Port

Alamat	**Keterangan**
38Ch-3BFh	Port pilihan yang dikontrol melalui BIOS. Port ini tidak mendukung alamat ECP
378h-37Fh	Alamat untuk LPT1
278h-27Fh	Alamat Untuk LPT2

Berikutnya kita akan mempelajari bagimana cara pengalamatan port ini. Hal yang paling penting perlu diketahui sebelum melakukan pembuatan pemrograman port adalah menentukan alamat port dasar yang dipilih atau piranti luarnya dihubungkan ke port mana. Kemudian baru kita dapat menentukan alamat ketiga tipe rigester yaitu Rigester **Kontrol**,

Register **Data** dan Register **Status**. Alamat dasar tersbut dapat dipilih dari salah satu alamat pada Table 2.

Setelah Anda menentukan alamt dasar yang akan digunakan, Anda bisa mengetahui alokasi bit-bit untuk masing -masing register yang ditabulasikan pada Tabel 3-5.

Tabel 3. Pemetaan Bit-bit pada Register Data

Offset	Nama	Read/write	Bit ke	Properti
Alamat dasar	Port Data	Write *)	Bit 7	Data 7
			Bit 6	Data 6
+0			Bit 5	Data 5
			Bit 4	Data 4
			Bit 3	Data 3
			Bit 2	Data 2
			Bit 1	Data 1
			Bit 0	Data 0

Tabel 4. Pemetaan Bit-bit pada Register Status

Offset	Nama	Read/write	Bit ke	Properti
Alamat dasar	Port Status	Read Only	Bit 7	Busy
			Bit 6	Ack
+1			Bit 5	Paper Out
			Bit 4	Selct In
			Bit 3	Error
			Bit 2	IRQ
			Bit 1	-
			Bit 0	-

Tabel 5. Pemetaan Bit-bit pada Register Kontrol

Offset	Nama	Read/write	Bit ke	Properti
Alamat dasar	Port Kontrol	Read/write	Bit 7	-
			Bit 6	-
+2			Bit 5	Enable Bi-di
			Bit 4	Enable IRQ
			Bit 3	Select
			Bit 2	Initial
			Bit 1	Auto
			Bit 0	Strobe

Berdasarkan pemahaman terhadap Port Parallel dan prinsip dasar pengalamatan port, Anda dapat memulai suatu proyek kecil dengan menggunakan rangkaian eksternal sederhana yang akan dihubungkan ke port parallel PC. Rangkaian sederhana itu seperti yang ditunjukkan pada Gambar 2 dengan menggunakan Port Paralel LPT1 terhubung melalui DB-25. Rangakain ini digunakan untuk melatih penggunaan pemrograman port untuk melakukan keluaran dari data dari PC ke parangkat keras eksternal.

Pada pembahasan kita selanjunya akan dianggap selalu menggunakan LPT1 dengan alamat register pada alamat dasar 378H pada Standard Parellel Port (SPP). Bedasarkan teknik pengalamatan pemrograman port yang telah dijelaskan di atas, alamat port LPT yang bersangkutan menjadi seperti pada Tabel 6-9 yaitu Register Data dengan alamat dasar +0 yaitu pada alamat 0378H , Register Status dengan alamat dasar +1 yaitu pada alamat 0379H dan Register Kontrol dengan alamat dasar +2 yaitu pada alamat 037AH.

Tabel 6. Register Data dengan alamat dasar +0 yaitu pada alamat 0378H

Offset	Nama	Read/write	Bit ke	Properti
Alamat dasar+0	**Port Data**	**Write *)**	**Bit 7**	**Data 7**
			Bit 6	**Data 6**
Atau			**Bit 5**	**Data 5**
			Bit 4	**Data 4**
0378h			**Bit 3**	**Data 3**
			Bit 2	**Data 2**
			Bit 1	**Data 1**
			Bit 0	**Data 0**

Tabel 7. Register Status dengan alamat dasar +1 yaitu pada alamat 0379H

Offset	Nama	Read/write	Bit ke	Properti
Alamat dasar +1	Port Status	Read Only	Bit 7	Busy
			Bit 6	Ack
Atau			Bit 5	Paper Out
			Bit 4	Selct In
0379h			Bit 3	Error
			Bit 2	IRQ
			Bit 1	-
			Bit 0	-

Tabel 8. Register Kontrol dengan alamat dasar +2 yaitu pada alamat 037AH

Offset	Nama	Read/write	Bit ke	Properti
Alamat dasar+2	Port Kontrol	Read/write	Bit 7	-
			Bit 6	-
Atau			Bit 5	Enable Bi-di
			Bit 4	Enable IRQ
037Ah			Bit 3	Select
			Bit 2	Initial
			Bit 1	Auto Linefeed
			Bit 0	Strobe

Sekarang Anda telah siap membuat program port dengan mengetahui dan menentukan register mana yang dipakai, bit mana yang harus dibaca dan kontrol apa yang harus dilakukan untuk piranti luar. Berikut ini adalah cara mengirimkan data ke port paralel yang telah Anda tentukan. Bentuk umum perintah pemrograman port adalah:

Outportb(*alamat port, nilai*)

dengan alamat port adalah alamat yang ingin dituju dan nilai adalah data yang ingin dikirimkan. Sedangkan untuk membaca masukan dari alamat port tertentu adalah

Inportb(*alamat port*)

Contohnya, jika Anda menggunakan LPT1 artinya Register Data pada alamat 378h, register Status pada alamat 379h dan register kontrol pada alamat 037Ah. Jika Anda ingin mengirimkan data 10h keluar ke Register Data untuk piranti luar 9misalnya LED) maka perintahnya adalah

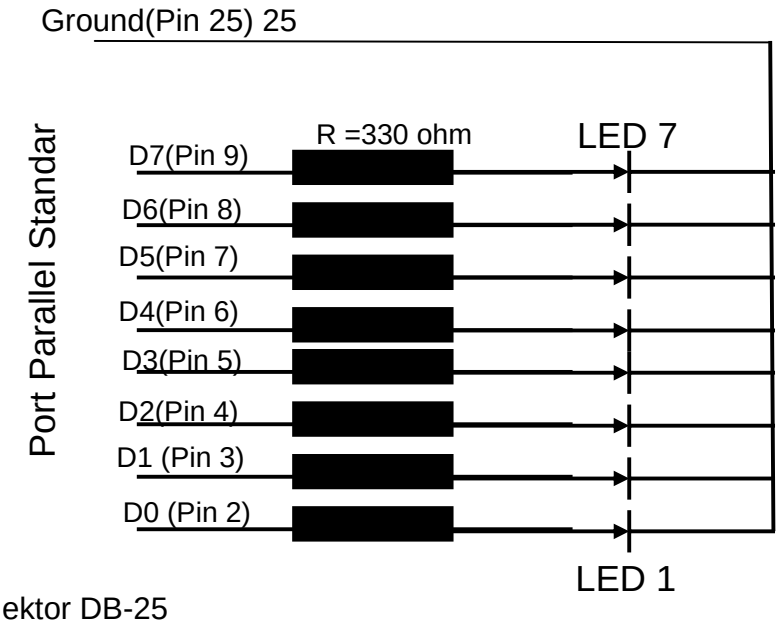
Outportb(0x378,0x10)

Sebaliknya jika Anda ingin mengambil masukan dari pirnati luar melaui Register Status adalah

inportb(0x379)

3. APLIKASI PEMROGRAMAN PORT C++ UNTUK TAMPILAN 8 BUAH LED

Pada aplikasi ini sengaja dibuat rangkain eksternal yang paling sederhana untuk memahami kerja dan cara membuat pemrograman port LPT. Rangkaian eksternal ini dapat dibuat hanya dengan menggunakan 8 buah LED dan juga dipasang 8 buah resistor 330 Ohm secara seri dengan LED agar arus tidak telalau besar ke LED. Ground untuk rangkaian ini diambil dari LPT pada pin 25. Desain rangkain yang dibuat menggunkan katoda LED yang dihubungkan ke Ground yang sering disebut common katoda (Gambar 3).



Gambar 3. Tampilan LED dengan konfigurasi Common Katoda

Berikut ini adalah cara untuk memprogram port menggunakan C++ untuk menyalakan LED:

```
/* ------------------------------------------------------------------------------*/
/*     Modul Pemrograman Port Untuk LED                                          */
/*     D0-D7 dihubungkan ke Pin2-Pin9 konektor DB25                              */
/*     Ground dihubungkan ke Ground pada Pin 25 konektor DB25                    */
/*------------------------------------------------------------------------------ */
#include<iostream.h>
#include<string.h>
#include<dos.h>
void main ()
{
      unsigned int far *h;
      clrscr();
      outportb(0x378,0x10);
}
```

Perintah outportb(0x378,0x20) artinya adalah Anda mengirimkan data heksa 10 atau (0000 1010)b ke alamat register 378h yang diteruskan ke pin 2-9 sehinga LED yang menyala adalah LED 2 dan LED 4.

4. APLIKASI PEMROGRAMAN PORT C++ UNTUK MUSIK

IBM PC biasanya mengunakan speaker untuk bunyi sebagai tanda kondisi error. Lebih dari itu, Anda juga dapat menggunakan speaker ini untuk menghasilkan irama musik. Metode ini menghasilkan timbre yang sangat menarik, apalagi jika digabungkan dengan efek grafik suatu animasi kita akan mendapatkan sensasi yang menarik.

Untuk menghasilkan irama, Anda harus membuat pulsa dan mengrimkanya ke speaker. Jika arus diberikan ke koil speaker, secara fisik membran speaker bergetar dan jika arusnya dimatikan speaker diam. Seandainya Anda memberikan sederetan sinyal on atau off ke speaker misalnya 100 kali per detik, artinya Anda telah membuat irama dengan frekuensi 100 Hz.

Untuk mengirimkan sinyal untuk menhidupkan speaker pada IBM PC, Anda dapat melakukannya dengan mengirimkan angka 2 ke port yang beralamat 61h. Sedangkan untuk mematikan Speaker Anda dapat mengirimkan 0 ke port 61h. Cara mengirimkannya adalah sebagi berikut:

Outport(*alamat port, nilai*)

dengan alamat *port* pada 61h dan *nilai* adalah 2 atau 0.

Outport(0x61h, 0x02h) untuk menghidupkan dan

Outport(0x61h, 0x0h) untuk mematikan

Berikut ini akan kita coba program yang mengkonversi apa yang anda ketik di keyboard menjadi tone musik dengan berbagai variasi pitch. Anda juga dengan mudah dapat mengganti fungsi **play**() untuk memasukkan durasi pitch yang diinginkan. Silahkan Anda bermain tone dengan keyboard.

```
/* ------------------------------------------------------------------------------*/
/*        Modul Pemrograman Port Untuk Musik Menggunakan Spkaer      */          */
/*        Mengaktifkan speaker dengan alamat  61h                    */
/*        Data 2 untuk on dan o untu off                             */
/*------------------------------------------------------------------------------ */
#include<iostream.h>
#include<string.h>
#include<dos.h>
void main ()

void main ( )
{
          int note, durasi;
        durasi=10;
        do {
                note=getchar( );
                play(note, durasi);
        }while (note!='q');
}
play(note,d)
int note,d;
{
        int t,tone;
        d=d+1000/note;
        for (;d:d--){
                tone=note;
                outport(0x61,2); /* menghidupkan spekaer*/
                outport(0x61,2); /* mematikan speaker*/
                for (;tone;--tone);
        }
}
```

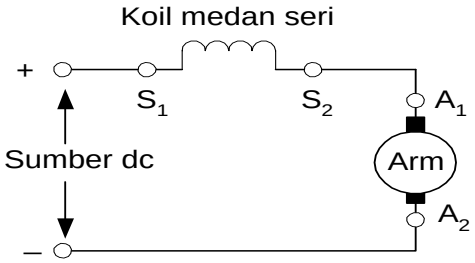
5. APLIKASI PEMROGRAMAN PORT C++ UNTUK MENGGERAKKAN MOTOR DC

Motor DC (Direct *Curent)* adalah motor yang menggunakan sumber tegangan searah. Ada beberapa jenis motor dc yang tersedia diantaranya adalah motor DC dengan koil medan dan motor DC dengan magnet permanen. Motor DC dengan koil medan membutuhkan arus yang lebih besar dari motor magnet permanen untuk mengatur medan magnet pada koil magnet. Oleh karena itu, koil medan mempunyai range kecepatan yang lebih luas dan torsi yang lebih besar. Motor DC magnet permanen tidak dapat mengubah besarnya medan magnet sehingga range torsi dan kecepatannya kecil.

5.1. Motor DC dengan Koil Medan

Motor dc Jenis Seri

Motor dc jenis seri terdiri dari medan seri yang diidentifikasi dengan S1 dan S2 yang terbuat dari sedikit lilitan kawat besar yang dihubungkan seri dengan jangkar yang diidentifikasi dengan A1 dan A2. Jenis motor dc ini mempunyai karakteristik torsi start dan kececpatan variabel yang tinggi. Ini berarti bahwa motor dapat start atau menggerakkan beban yang sangat berat, tetapi kecepatan akan bertambah kalau beban turun. Motor dc seri dapat membangkitkan tori starting yang besar karena arus yang sama yang melewati jangkar juga melewati medan. Jadi, jika jangkar memerlukan arus lebih banyak (membangkitkan torsi lebih besar), arus ini juga melewati medan, menambah kekuatan medan. Oleh karena itu, motor seri berputar cepat dengan beban ringan dan berputar lambat pada saat beban ditambahkan. Sifat istimewa terpenting dari motor dc seri adalah kemampuannya untuk start atau menjalankan beban yang sangat berat. Karena alasan itu, motor sering digunakan pada



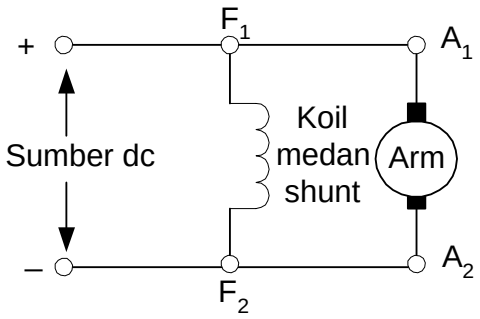
Gambar 4. Motor DC Jenid Seri

gerakan dan elevator. Untuk membalik arah putaran motor dc seri, balik arah arus pada kumparan seri atau kumparan jangkar.

Jenis motor dc ini juga disebut motor universal karena dapat dioperasikan baik dengan arus searah maupun dengan arus bolak-balik. Alasan untuk ini adalah bahwa motor dc akan terus berputar pada arah yang sama jika arus yang mengalir pada jangkar dan arus yang mengalir pada medan yang dibalik pada waktu bersamaan.

Motor dc Jenis Shunt

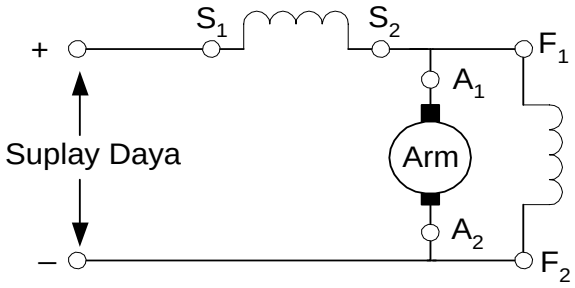
Pada motor ddc jenis shunt, kumparan medan shunt (diidentifikasikan dengan F1 dan F2) dibuat dengan banyak lilitan kawat kecil', karena itu mempunjyai tahanan yang tinggi. motor shunt mempunyai rangkaian jangkar dan medan yang dihubungkan pararel yang memberikan kekuatan medan dan kecepatan motor yang sangat konstan. Motor shunt digunakan jika diperlukan pengatur kecepatan yang bagus pada mesin yang digerakkan. Dengan menambah rheostat yang dipasang seri dengan rangkain medan shunt, kecepatan motor dapat dikontrol di atas keceepatan dasar. Kecepatan motor akan berbanding terbalik dengan dengan arus medan. Ini berarti motor shunt berputar cepat dengan arus medan rendah, dan bertambah pada saat arus ditambah. Motor shunt dapat melaju pada kecepatan tinggi yang berbahaya jika arus kumparan medan hilang.



Gambar 5. Motor DC Jenid Shunt

Motor dc jenis Gabungan

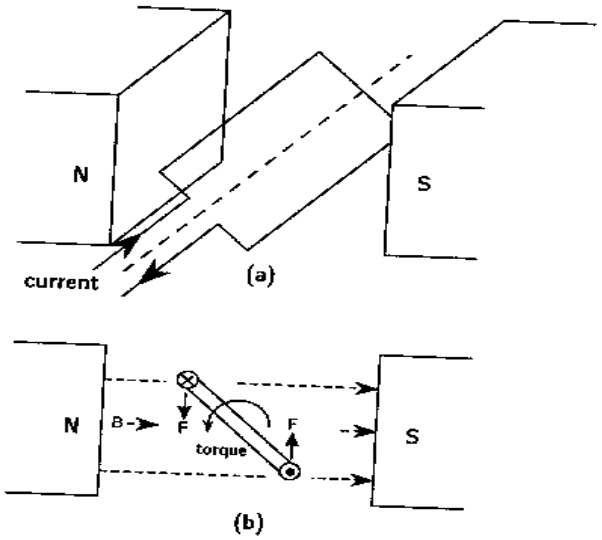
Motor dc jenis compound menggunakan lilitan seri dan lilitan shunt, yang umumnya dihubungkan dengan medan-medannya bertambah secara kumulatif. Hubungan dua lilitan ini menghasilkan karakteristik pada motor medan shunt dan motor medan seri. Kecepatan motor tersebut bervariasi lebih sedikit dibandingkan motor shunt, tetapi tidak sebanyak motor seri. Motor dc jenis compound juga mempunyai torsi starting yang agak besar- jauh lebih besar dibandingkan dengan motor shunt, tetapi sedikit lebih kecil dibandingkan motor seri. Keistimewaan gabungan ini membuat motor compound memberikan variasi penggunaan yang luas.



Gambar 6. Motor DC Jenid Shunt

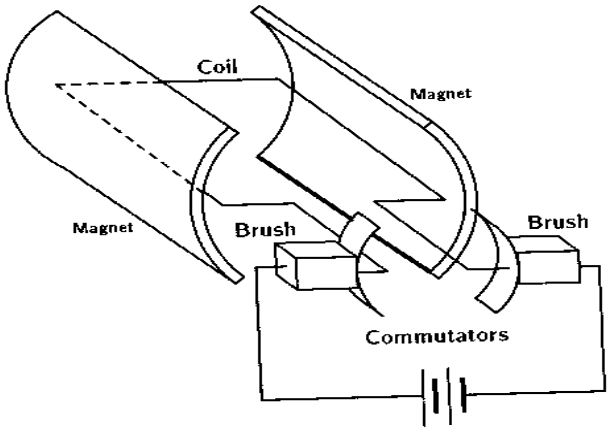
5.2. Motor DC Magnet Permanen

Motor DC magnet permanen terdapat dua bagian, yaitu stator dan rotor. Stator adalah bagian yang diam sedangkan rotor adalah bagian yang ikut berputar. Perputaran rotor ini disebabkan karena adanya torka yang dibangkitkan oleh konduktor yang dialiri arus dalam medan magnet seperti terlihat dalam Gambar 11.6



Gambar 7. *Torka yang dibangkitkan oleh arus yang mengalir dalam rotor*

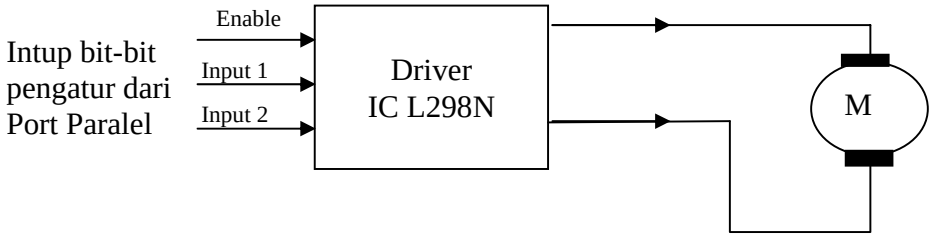
Memperhatikan Gambar 7., motor akan berputar bolak balik karena setelah lengan berputar 180^0 atau tegak lurus dengan arah medan magnet arah torsi berbalik arah. Untuk menghindari gerakan bolak balik ini maka arus yang diberikan harus dibalik pada saat lengan tegak lurus dengan medan magnet proses ini disebut *Commutation.* Untuk merubah arah arus tersebut dapat dilakukan dengan menggunakan sikat atau *commutators*. Gambar 8. memperlihatkan cara kerja *commutators* dari motor DC magnet permanen.



Gambar 8. *Cara kerja commutator*

5.3. Pengaturan Gerak Motor DC

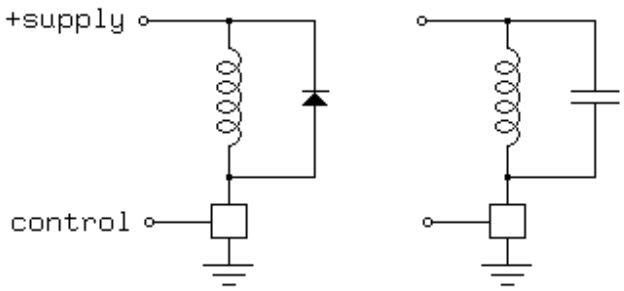
Salah satu diagram Blok untuk menggerkakan motor dc seperti pada Gambar 9. dengan driver IC L298N.



Gambar 9. *Diagram Blok Pengaturan motor Dc dengan Drriver IC L298N*

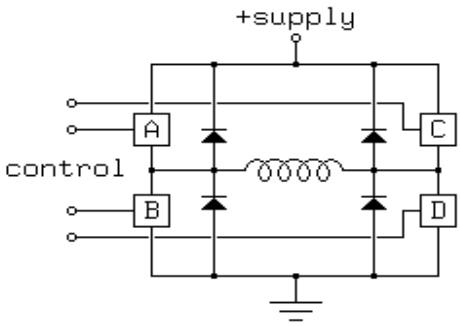
Driver Motor DC

Driver motor digunakan untuk mengontrol lamanya putaran motor dan arah putaran motor. Prinsip kerja dari driver ini hanya dengan melewatkan arus pada motor dan menghentikan arus yang melewati motor serta mengatur arah arusnya dengan menggunakan *switch.* Kumparan motor merupakan beban induktif, sehingga arus yang melewati kumparan motor tidak dapat dinyalakan dan dimatikan dengan segera. Ketika *switch* pengontrol kumparan motor terhubung maka arus akan naik dengan perlahan. Sedangkan pada saat *switch* pengontrol ini dilepas arus dari kumparan turun perlahan, akibatnya terjadi *Voltage spike* yang dapat merusak *switch*. Untuk menghindari *voltage spike* ini digunakan dioda atau kapasitor seperti pada gambar berikut :



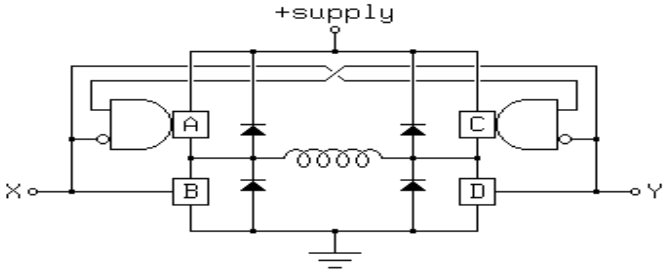
Gambar 10. *Driver motor dengan pengaman*

Driver pada Gambar 10. merupakan driver untuk mengontrol lamanya perputaran motor. Dari gambar tersebut *switch* digerakkan oleh suatu pulsa yang dibangkitkan oleh dari PC. Untuk mengontrol arah putaran motor dapat digunakan jembatan H *driver* dapat dibuat seperti rangkaian pada Gambar 11.



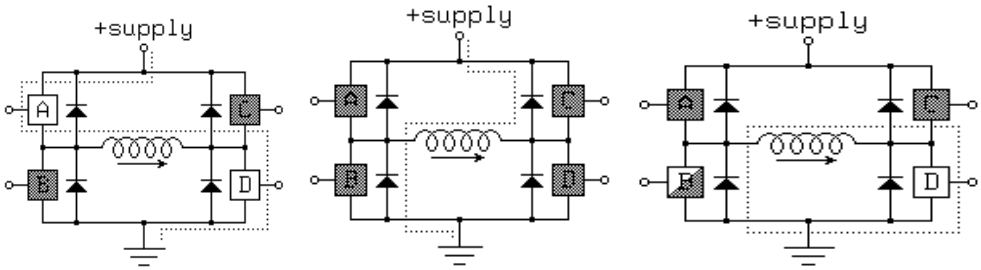
Gambar 11. *Driver jembatan H.*

Dengan menggunakan 4 switch, jembatan H akan memberikan 16 kemungkinan dengan 7 kemungkinan mengakibatkan hubung singkat. Untuk mengatasi hubung singkat ini maka jembatan H dibuat sebagai berikut :



Gambar 12. *Jembatan H untuk menghindari hubung singkat.*

Dari rangkain di atas hanya terdapat 4 kemungkin. Kemungkinan pertama yaitu pada saat X = 0 dan Y = 1 keadaan ABCD = 1001, motor akan berputar kedepan seperti pada Gambar 13(a) sedangkan pada saat X = 1 dan Y = 0 keadaan *switch* ABCD menjadi 0110 motor berbalik arah . Pada keadaan X = 0 dan Y = 0, semua *switch* dalam keadaan terbuka semua arus yang berasal dari kumparan motor akan cepat meluruh, sehingga motor berhenti secara perlahan. Keadaan X = 1 dan Y = 1 mengakibatkan *switch* B dan D tertutup, sehingga apabila motor sedang bergerak kemudian diberi pulsa seperti ini, maka arus kumparan motor akan meluruh dengan lambat, akibatnya motor akan mengerem. Prinsif dasar jembatan inilah yang ada pada driver IC L298N.

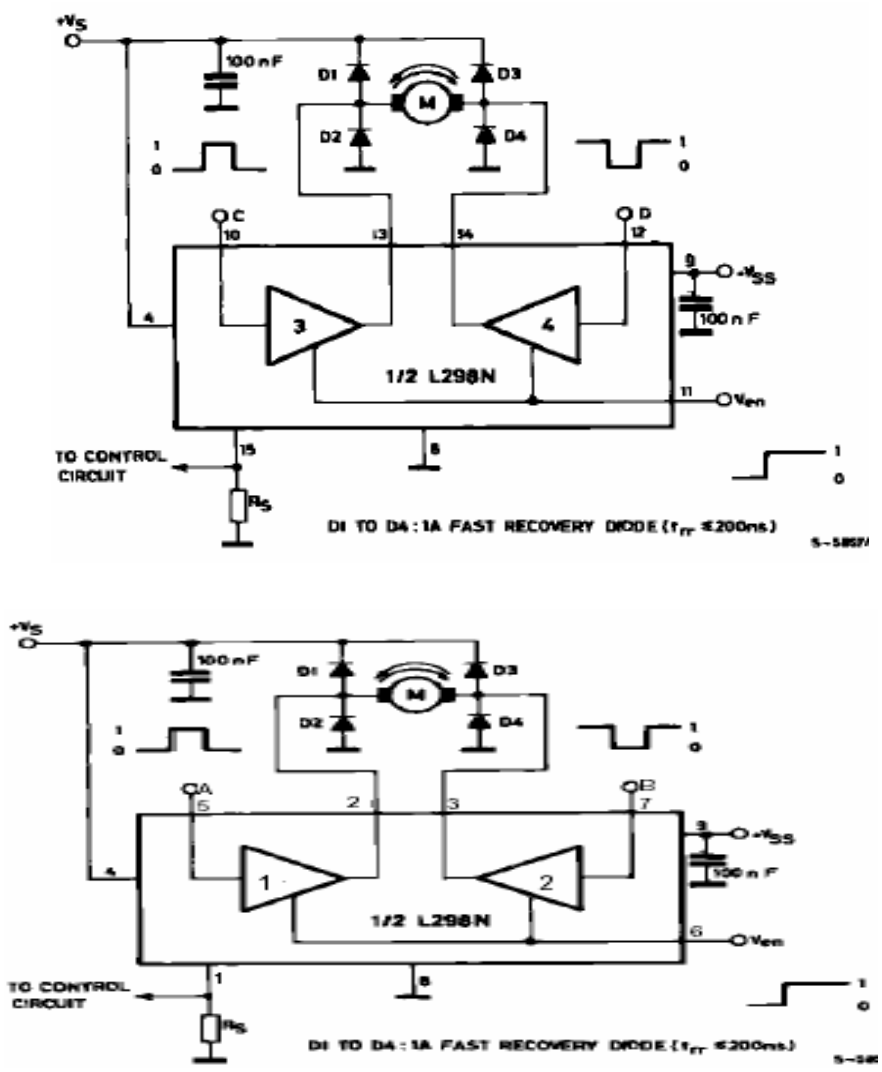


(a)berputar *(b)tergelincir* *(c)mengerem*

Gambar 13. *Keadaan Jembatan H*

Umumnya penggunaan motor DC tidak digerakkan dalam satu arah melainkan dapat digerakkan secara bolak-balik. Untuk mengatur perubahan arah gerak motor ini digunakan *driver*. Driver yang digunakan adalah IC L298N yang merupakan jenis jembatan H (*H-Bridge*) yang terdiri dari dua buah jembatan.

Masing-masing jembatan terdiri dari tiga masukan, yaitu dua masukkan In dan sebuah masukan En. Masukan En digunakan untuk mengktifkan jembatan H sedangkan masukan In untuk mengatur putaran motor seperti terlihat pada Tabel 9 logika putaran motor. Seperti pada Gambar 14. dua buah jembatan H yang terdapat dalam satu IC digunakan untuk mengatur dua motor.



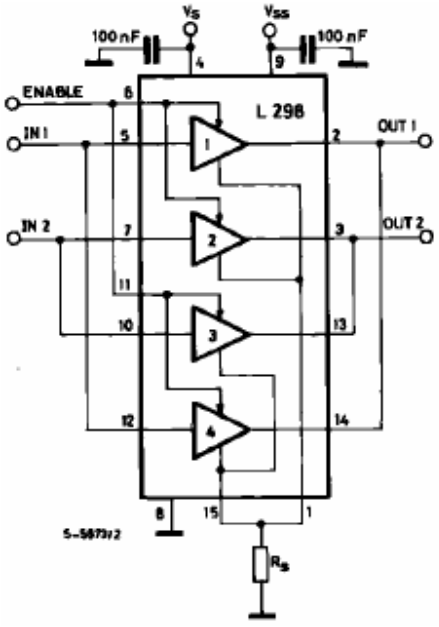
Gambar 14. *Rangkaian Driver motor L298*

Tabel 9. Logika perputaran motor

Masukan		Fungsi
En = H	In1 = H;In2 = L	Berputar ke kanan
	In1 = L;In2 = H	Berputar ke kiri
	In1 = In2	Mengerem
En = L	In1 = X;In2 = X	Berhenti perlahan-lahan

Keterangan : H = tinggi, L = rendah, X = tinggi atau rendah

Dioda D1 sampai D4 digunakan untuk melewatkan arus yang dihasilkan dari putaran motor saat switch dimatikan sehingga tidak akan merusak IC. Tegangan output *sense* dapat digunakan untuk mengontrol arus dengan R_s digunakan untuk mendeteksi intensitas arus. Rangkaian *driver* di atas digunakan untuk menggerakkan motor dengan arus kurang dari 2 A, sedangkan untuk menggerakkan motor dengan arus lebih dari 2 A dapat digunakan rangkaian dengan konfigurasi pararel seperti pada Gambar 14.

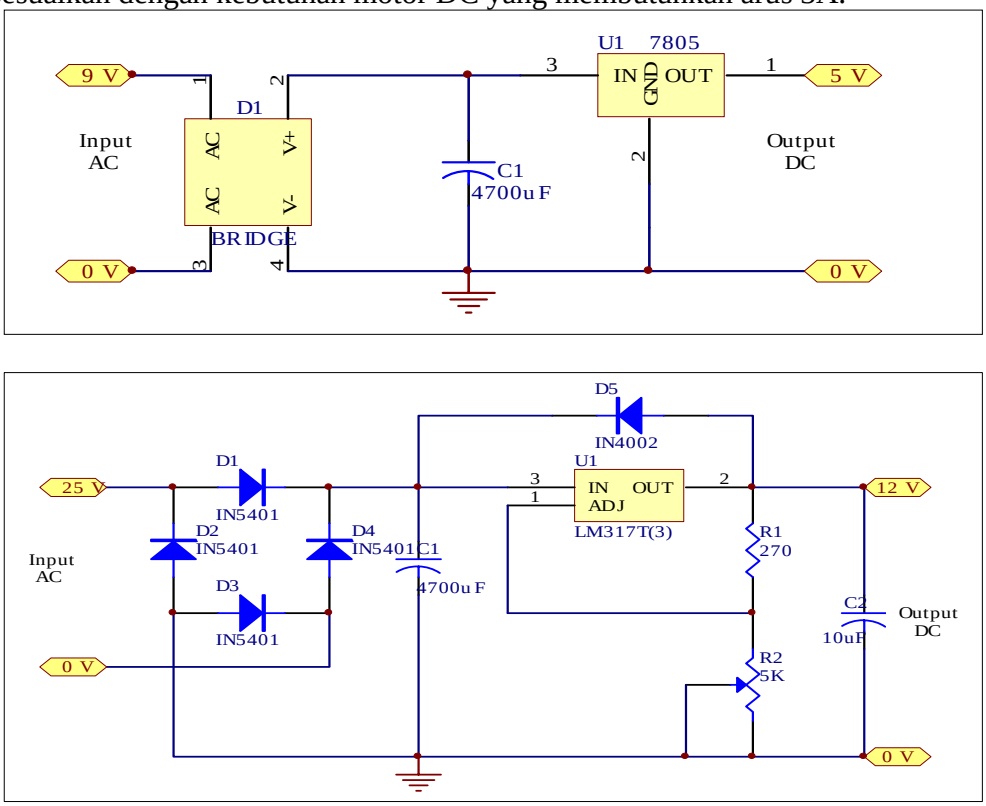


Gambar 15. *Rangkaian Driver motor untuk I> 2A*

Motor yang digunakan dalam perancangan ini ada dua jenis yaitu dua buah motor *gearbox* dengan arus kurang dari 2 A dan sebuah motor dengan arus 3 A tanpa *gearbox*. Penggunaan *gearbox* ini mengakibatkan kecepatan motor menjadi rendah dan torsi motor meningkat hal ini perlu diberikan pada motor dengan arus yang kecil sedangkan motor dengan arus 3 A memiliki kecepatan dan torsi yang tinggi. Penggunaan torsi yang tinggi dikarenakan motor harus dapat menggerakkan tempat pakan yang memiliki beban yang besar.

Sumber Tegangan

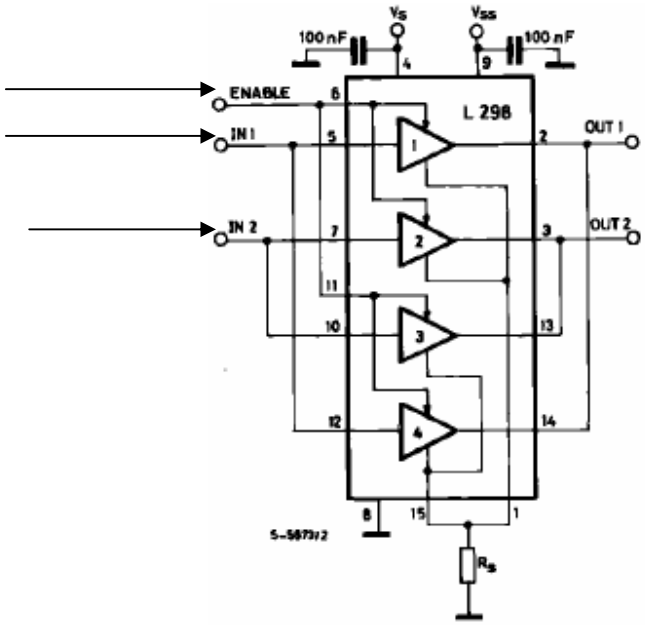
Sumber tegangan yang dibutuhkan untuk motor dc ini adalah 5 V dan 12 V. Tegangan 5 V digunakan untuk mengaktifkan *driver*, sedangkan tegangan 12 V digunakan untuk menggerakkan motor DC. Sumber tegangan yang dibuat menggunakan transformator 3 A, hal ini disesuaikan dengan kebutuhan motor DC yang membutuhkan arus 3A.



Gambar 16. *Power supply*

Pemrograman Port Paralel Untuk Pengaturan motor dc

Rangakai ke komputer atau port paralel ini sangat sederhanan seperti yang ditunjukkan Gambar 11.14 yaitu dengan menghubungkan motoR DC pada *Enable* ke Port Kontrol LPT. Sama Umumnya Port parallel mempunyai resitor untuk internal pull-up, tetapi beberapa mungkin tidak dilengkapi dengan resistor ini. Anda disarankan lebih baik melengkapi saja Port Kontrol ini dengan sebuah resitor 10K untuk eksternal pull up agar rangkain dapat lebih portable untuk berbagai jenis komputer yang mungkin tidak memiliki resistor sebagai internal pull up.



Gambar 17. Hubungan Dari LPT ke Motor DC

Untuk mengerakkan motor dc ini dari komputer dibutuhkan tiga bit data dengan kondisi-kondisi seperti pada Tabel 9 1 bit untuk enable yang membuat motor bergerak atau berhenti perlahan dan 2 bit untuk arah grakan kiri, gerak kanan atau mengerem. Alokasi port paralel untuk mengerkakkan motor dc ini adalah pada Register Data digunakan pin 2 dan 3 untuk arah gerak pada In 1 dan in 2 motor dc dan Register Kontrol pada pin 4 untuk enable pada motor dc.

Instruksi untuk bergerak atau berhenti dikirimkan ke motor dc yang dikontrol oleh *Register Select* pada Pin 4. Ketika pin 4 rendah instruksi register kontrol diambil diambil dan ketika tinggi register data harus diambil. Instruksi ini kita hubungkan ke Port Parallel *Select Printer* yang kadang diinversi oleh hardware, sehingga Anda perlu menuliskan logika '1' ke bit 3 pada Register Kontrol agar *Select Printer* menjadi rendah.

Agar Anda dapat mengirimkan *instruksi* ke motor dc, maka *Register Select* harus dibuat rendah. Hal ini dapat dilakukan dengan mengeset bit 3 dari Register Kontrol menjadi '1'. Tentu saja kita tidak ingin register ini diset ulang pada saat pengirimin bit lain pada Port Registre Kontrol. Cara menaggulanginya adalah dengan membaca Port kontrol dan melakukan operasi OR dengan 0x80, berikut contoh programnnya

```
outportb(KONTROL, inportb(KONTROL) | 0x08);
```

Operasi ini hanya akan mengset bit 3 pada Register Kontrol. Setelah itu Anda menempatkan data byte pada baris data untuk menentukan arah gerak motor dc. Data tersebut kemudian dikirm ke motor dc dengan pewaktuan (timing) dari transisi tinggi ke rendah. *Strobe* diinversi secara hardware yaitu dengan mengeset bit 0 pada Register Kontrol sehingga Anda

mendapatkan transisi tinggi ke rendah pada baris *Strobe*. Kemudian kita menunggu delay dan kembali ke level tinggi untuk pengiriman byte berikutnya. Byte ini sebenarnya dibutuhkan hanya 2 bit yaitu bit o dan bit 1 pada register data. Jika anda ingin memutar ke arah kanan Anda harus mengirimkan byte 0001 atau 1h dan jika anda ingin berputar ke arah kiri anda harus mengirimkan byte 0010 atau 2h. Jika ingin motor direm maka Anda harus memberikan kedu bit tersebut rendah atau tinggi artinya Anda bisa mengirm 0000 atau 0h atau 0011 atau 3h.

```
outportb(Data, 0x01);   untuk arah gerak kanan

outportb(Data, 0x02);   untuk arah gerak kiri

outportb(Data, 0x03);   untuk mengeremi
```

Setelah Anda mengirimkan data arah putaran motor dc yang dikirim melalui Register Data ke Input 1 dan Input 2 motor dc, Anda perlu menjadikan bit 3 nol. Untuk itu kita bisa menggunkan perintah

```
outportb(KONTROL, inportb(KONTROL) & 0xF7);
```

Langkah berikutnya adalah melakukan perulangan sesui dengan keinginan Anda mengatur gerakan motor dc berikutnya. Perintah perulangan dapat dilakukan dengan menggunakan *for*.

Dilay yang Anda berikan harus cocok cukup untuk motor dc merespon perintah dari komputer. Jika ternyata motor dc tidak melakukan gerakan maka Anda dapat menaikkan delay. Agar lebih mudah, kita masih menggunakan LPT1 dengan Standard Parellel Port (SPP) pada alamat register dasar 378H. Jadi kita menggunakan Register Data dengan alamat dasar +0 yaitu pada alamat 0378H dan Register Kontrol dengan alamat dasar +2 yaitu pada alamat 037AH. Dalam hal ini kita tidak menggunakan Register Status dengan alamat dasar +1 yaitu pada alamat 0379H.

6. APLIKASI PEMROGRAMAN PORT UNTUK PENGATUR GERAK MOTOR STEPER

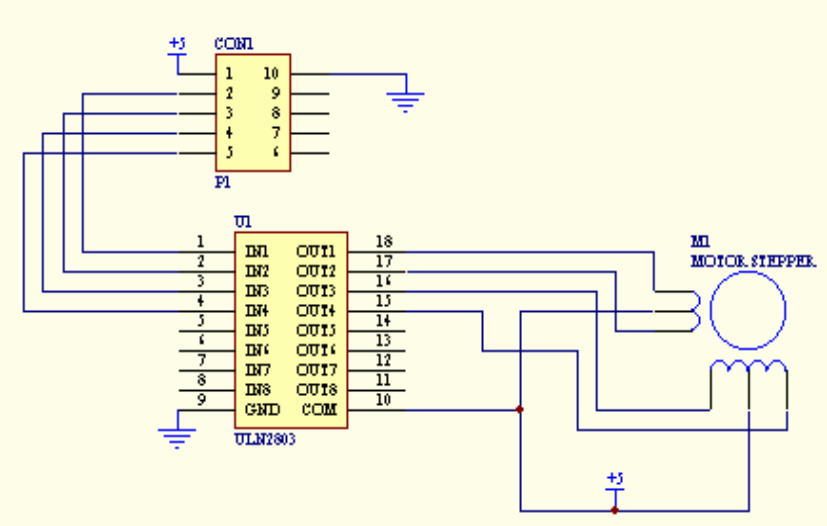
Motor stepper memiliki kontrol posisi dan kecepatan yang tepat serta memiliki torsi yang besar pada kecepatan rendah. Tegangan masukan stepper motor berupa sinyal pulsa yang akan menggerakan rotor dalam posisi diskrit.. Untuk menggerakan motor stepper dapat dilakukan dengan mengirimkan data digital 4 bit yang digeser secara berurutan. Data yang dikirm untuk mengerakkan motor stepper dapat dapat dilihat pada Tabel 12.2.

Gambar 18. Bentuk fisik Motor Stepper

Tabel 10. Data yang diberikan untuk menggerakan motor stepper

Data Binner	Data Decimal	Langkah motor stepper
0001	1	1
0010	2	2
0100	4	3
1000	8	4

Bila sudah dilangkah ke-4 ingin dilanjutkan kelangkah ke-5, maka dapat dilakukan dengan mengulang kembali mengirimkan data 0001 atau 1 desimal dan ke langkah ke-6 juga dengan mingirim data 0010 atau 2 desimal dan begitu seterusnya.



Gambar 19. Rangkaian Interface Motor Stepper

Data digital output dari LPT memiliki arus yang kecil sehingga tidak dapat digunakan untuk menggerakan motor stepper, sehingga perlu ditambahkan rangkaian *driver* untuk menguatkan arus sehingga dapat digunakan untuk menggerak motor stepper. Salah satu driver yang dapat digunakan adalah IC ULN2803 yang berfungsi untuk mengutakan arus. Skematik rangkaian driver motor stepper dapat dilihat seperti pada Gambar 19.

```
/*  Pemrograman Port untuk  pengaturan motor steper          */

#include <dos.h>
#include <string.h>

#define alamatportdasar 0x378  /* Memasukkan alamat dasar port LPT yang dipakai

#define DATA alamatportdasar+0
#define STATUS alamatportdasar+1
#define KONTROL alamatportdasar+2

void main(void)
{
```

```
unsigned int far *h;
int cacahan;
clrscr();
a=1
for (cacahan = 0; cacahan <= 7; cacahan++) /* Tanya reapet until keypreesed??*/
  {
  a=data
 outportb(DATA, a);
a.=a.>1
 }
```

7. APLIKASI PEMROGRAMAN PORT C++ UNTUK TAMPILAN LCD DUA BARIS DENGAN 16 KARAKTER

7.1. Gambaran umum pengoerasian LCD

LCD merupakan salah satu perangkat *display* yang bisa menampilkan gambar atau karakter yang diinginkan. Dalam hubungannya dengan mikrokontroler, LCD dapat menampilkan data ascii dalam suatu program yang telah dimasukan ke dalam chip ketika chip tersebut dijalankan.

LCD mempunyai *display dot matrix*, yang dilengkapi dengan panel dan rangkaian *controller/driver*. LCD ini bisa menampilkan 1 baris, 2 baris, atau 4 baris, tiap baris memiliki 16 karakter. Untuk kegunaan display, LCD biasanya mempunyai rangkaian pengontrol, data RAM, dan ROM pembangkit karakter.

Skema pin LCD untuk 2x16 beserta bilangan hexadesimal dari inisialisasinya pada pemakaian assembler dapat dilihat pada lampiran. Untuk pengiriman dan penerimaan data dari dan ke LCD, dibutuhkan pin-pin kontrol (Tabel 11).

Tabel 11. Fungsi Sinyal Kontrol

SINYAL KONTROL	FUNGSI
E	**Menyebabkan kondisi data/kontrol di-latch** Transisi Naik : me-latch kondisi kontrol (RS dan R_W) Transisi Turun : me-latch data
RS	**Register Select Kontrol** 1 = LCD pada mode data 0 = LCD pada mode command
R_W	**Read/Write Control** 1 = LCD menulis data 0 = LCD membaca data

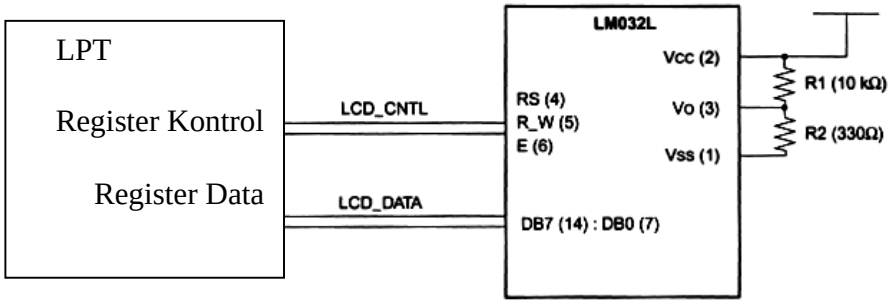
Beberapa LCD dapat dioperasikan menjadi beberapa mode atau cara.. LCD jenis LM032L dapat dioperasikan menjadi 2 cara. Cara 1 interface data 4 bit, dan yang ke dua interface data 8 bit. Jika pengoperasiannya menggunakan data 4 bit maka dibutuhkan 2 kali pengiriman data per karakter, sedangkan menggunakan pengiriman data 8 bit relatif lebih mudah, karena tidak menghabiskan memori program tapi membutuhkan 4 tambahan jalur I/O.

Dalam implementasinya secara umum ada 3 cara yang sering digunakan:

1. Interface 8 bit
2. Interface data 4 bit, dengan pengiriman data *high nibble* pada port
3. Interface data 4 bit, dengan pengiriman data *low nibble* pada port

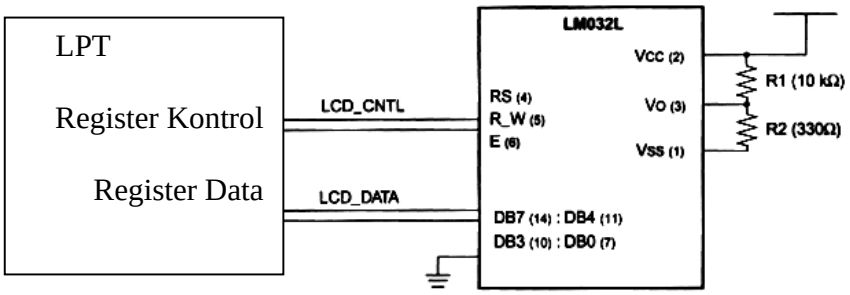
Ketiga cara pengoperasian data pada LCD seperti yang telah ditulis di atas, dapat dilihat pada skema/gambar berikut :

1. Interface Data 8 bit



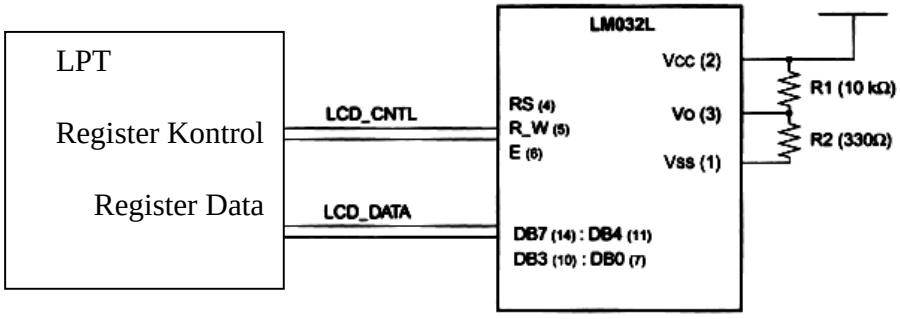
Gambar 20. Interface data 8 bit

2. Interface data 4 bit, pengiriman data high nibble pada port



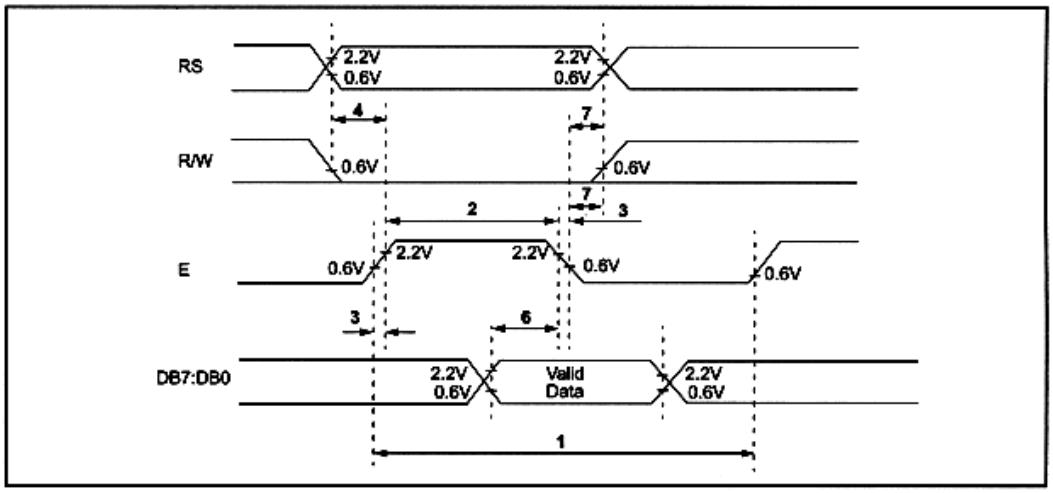
Gambar 21. Interface data 4 bit, pengiriman data high nibble pada port

3. Interface data 4 bit, pengiriman data low nibble pada port

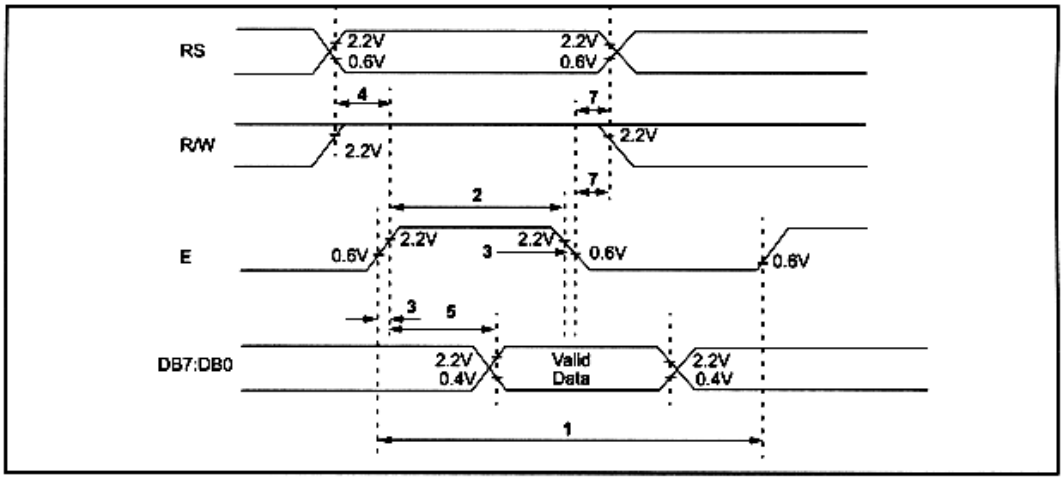


Gambar 22. Interface data 4 bit, pengiriman data low nibble pada port

LCD ini juga mempunyai tiga sinyal kontrol, diantaranya: Enable (E), Read/Write (R_W), dan register select (RS). Untuk menampilkan suatu huruf atau angka, data yang dikirim harus merupakan kode ASCII dari huruf dan angka tersebut.
Bagan pewaktu interface untuk penulisan dan pembacaan data dapat dilihat seperti pada Gambar 23-24.



Gambar 23. Pewaktu Interface Penulisan Data



Gambar 24. Pewaktu Interface Pembacaan Data

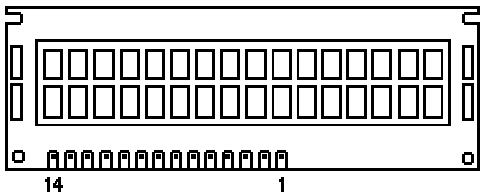
Daftar instruksi pada Tabel 12. merupakan perintah untuk LCD dalam pengoperasiannya :

Tabel 12. Daftar Perintah untuk LCD

KODE HEX	INSTRUKSI PERINTAH LCD
1	Clear display screen
2	Return home
4	Decrement cursor (shift cursor to left)
5	Increment cursor (shift cursor to right)
6	Shift display right
7	Shift display left
8	Display OFF, Cursor OFF
A	Display OFF, Cursor ON
C	Display ON, Cursor OFF
E	Display OFF, Cursor blinking
F	Display ON, Cursor blinking
10	Shift cursor position to left
14	Shift cursor position to right
18	Shift the entire display to left
1C	Shift the entire display to right
80	Force cursor to beginning of 1st line
C0	Force cursor to beginning of 2nd line
38	2 lines and 5 x 7 matrix

7.2. Inisialisasi LCD

Diagram nomber pin untuk panel LCD yang ditampilkan dari tampak depan dengan pin 14 paling sebelah kiri dan pin 1 pada sebelah kanan.



Gambar 25. Pin pada panel LCD

Pada prosedur ini harus dilakukan inisialisasai terhadap jenis LCD yang digunakan, prosedur tersebut adalah sebagai berikut :

A. Untuk melakukan inisialisasi LCD dikirimkan bebarapa perintah inisialisasi. Sebelum mengirimkan perintah Anda harus melakukan pengecekan apakah LCD sudah siap

menerima data. Cara pengecekanya dilakukan dengan rutin program siap() Gambar 26.(a) yauitu dengan mengirim bit-bit control R/W logika satu dan RS logika nol. Selanjutnya Anda menunggu jawaban dari LCD yang akan dikirimkannya melalui bit P0.7.

B. Bit RS mempunyai dua keadaan yaitu nol dan satu. Jika Anda ingin memberikan perintah maka harus diberi logika nol dan jika ingin mmengirimkan data maka harus diberi logika satu.

C. Bit R/W juga mempunyai dua keadaan yaitu logika nol dan satu. Jika Anda ingin membaca kondisi LCD pada bit BF (*busy flag*), maka Anda harus mengirimkan logika 1 dan jawabannya akan dikrimkan melalui bit P0.7. Bila Anda akan melakukan penulisan data ke LCD maka bit R/W ini harus diberi logika nol.

D. Apabila pada bit P0.7 ada pada logika satu, hal ini menandakan bahwa LCD sedang dalam keadaan sibuk sehingga tidak dapat menerima data atau perintah. Tetapi apabila bit P0.7 berlogika nol maka menandakan LCD dalam kondisi siap menerima data atau perintah.

E. Setelah selesai proses pengecekan bit P0.7, dan LCD telah memberi tanda sudah siap menerima data atau perintah maka langkah selanjutanya adalah menjalankan rutin program perintah() yang dapat dilihat pada Gambar 26(b). Pengiriman perintah menggunakan jalur data D0-D7, sehingga seluruh data-data inisialisasi dikirim melalui jalur ini.

```
void siap()
{
 Each = 0;
 RS = 0;
 RatauW = 1;
do
  {
  Each = 0;
  Each = 1;
  }
while (antos == 1);
  Each = 0;
}
```

(a)

```
void perintah()
{
siap();
P0 = unpad;
RS = 0;
RatauW = 0;
Each = 1;
Each = 0;
}
```

(b)

Gambar 26 : a. Rutin pengecekan kesiapan LCD Matrik untuk menerima data.
b. Rutin untuk mengirimkan perintah ke LCD

```
Void inisialisasi()
{
unsigned char init[4] = {0x38,0xC,0x6,0x1};
char x;
for (x = 0 ; x < 4 ; x ++)
 {
 unpad = init[x];
 perintah();
 }
}
```

Gambar 26. Rutin pengiriman data-data Inisialisasi LCD

F. Setelah pengecekan kesipana LCD dilakukan, selanjutnya Anda melakukan pengiriman data-data inisilisasi yaitu pengiriman data 38h yang merupakan perintah untuk menjadikan LCD ini bekerja dengan matrik 5 x 7 dot per karakter. Kemudian dilakukan pengecekan kembali kesiapan LCD untuk menerima data insialisasi berikutnya. Data berikutnya adalah data fh yang merupakan perintah untuk menampikan kursor pada LCD dan membuat kursor berkedip. Data inisialisasi ke tiga adalah data 6h yang merupakan perintah untuk membuat kursor bergeser dari kiri ke kanan. Sedangkan data lainnya adalah data 1h yang merupakan perintah untuk membersihkan layar LCD dan menenpatkan kursor dialamat memori paling awal (80h). Rutin program pengiriman data-data inisialilasi LCD ini ditulis dalam rutin program inisialisai() Gambar 26.

7.3. Pengiriman alamat tujuan penulisan karakter

A. LCD yang digunakan dalam contoh ini adalah merupakan LCD Matrik 2 baris x 16 kolom yang setiap kolom memiliki alamat masing-masing yang sudah ditentukan oleh pabrik pembuatnya dengan pengalamatannya dapat dilihat pada Tabel 13.

Tabel 13. Pengalamatan LCD Matrik 2 baris x 16 kolom

	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**
Baris ke 1	**80**	**81**	**82**	**83**	**84**	**85**	**86**	**87**	**88**	**89**	**8A**	**8B**	**8C**	**8D**	**8E**	**8F**
Baris ke 2	**C0**	**C1**	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**	**C9**	**CA**	**CB**	**CC**	**CD**	**CE**	**CF**

B. Alamat tujuan penulisan karakter yang akan Anda kirim ke LCD harus dikirim sebagai rutin perintah bukan sebagai data ASCII. Bila pada pengiriman karakter yang banyak dan penempatannya berada pada alamat memori yang berurutan dari kiri ke kanan, maka pengiriman alamat tujuan penulisan karakter cukup satu kali saja yaitu dikirim alamat yang paling kiri. Jadi ketika karakter berikutnya dikirim maka LCD langsung menaikkan satu alamat tujuan penulisan karakternya.

7.4. Mengirim karakter ASCII

A. Data yang akan Anda tampilkan di LCD ini adalah data dalam bentuk karakter ASCII.. Pengiriman data ini harus menggunakan rutin program tersendiri yang berbeda dengan rutin program perintah. Rutin program untuk mengirim data ASCII dapat Anda lihat pada rutin program tampil() pada Gambar 27.

```
void tampil()
{
siap();
P0 = unpad;
RS = 1;
RatauW = 0;
Each = 1;
Each = 0;
}
```

Gambar 27. Rutin program untuk mengirim data ASCII

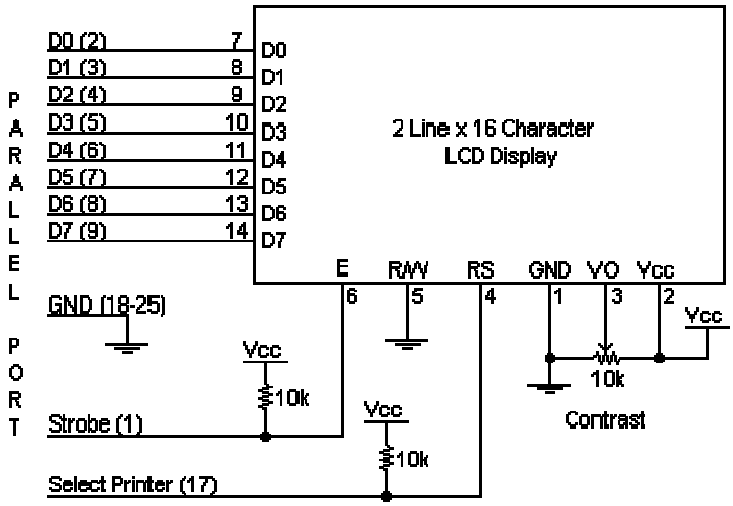
B. Apabila Anda akan melakukan pengiriman 16 karater sekaligus maka sebaiknya datanya disimpan pada Array baru dengan menggunakan perintah *for* untuk melakukan perulangan atau *looping* pengiriman ke-16 data tersebut.

Bila Anda akan melakukan penggantian tampilan pada LCD dengan tampilan yang berbeda dengan tampilan sebelumnya, maka sebaiknya sebelum datanya dikirim diawali dengan perintah untuk membersihkan LCD dan memindahkan kursor ke alamat tujuan karakter kiri atas (80h).

7.5. Anatarmuka LCD ke Port Parallel LPT PC

Setelah Anda mengenal arsitektur, aturan inisialisasi dan aturan pengiriman data ke LCD, sekarang Anda akan diperkenalkan dengan cara membuat program port yang dihubungkan ke LPT. LCD 2 baris x 16 karakter ini sangat banyak digunakan orang saat ini

karana cukup sederhana cara kerjannya dan relatif murah. Port LPT yang akan digunakan untuk mengantarmuka LCD ini adalah Standard Parallel Port (SPP).



Gambar 28. Skematik rangkaian antarmuka dengan port parallel

Rangakainnnya sangat sederhana dengan menghubungkan panel LCD yaitu *Enable* dan *Register Select* ke Port Kontrol LPT. Umumnya Port parallel mempunyai resitor untuk internal pull-up, tetapi beberapa mungkin tidak dilengkapi dengan resistor ini. Anda disarankan lebih baik melengkapi saja dengan dua buah resitor 10K untuk eksternal pull up agar rangkain dapat lebih portable untuk berbagai jenis komputer yang mungkin tidak memiliki resistor sebagai internal pull up.

R/W pada panel LCD Anda hubungkan pada write mode. Cara ini tidak akan mengakibatkan konflik bus pada baris data. Akan tetapi kita tidak bisa membaca balik internal LCD Busy Flag yang memberitahukan kita apakah LCD telah menerima dan menyelesaikan proses eksekusi instruksi terakhir. Hal ini bisa diatasi dengan memberikan delay pada program yang dibuat. Berikut ini adalah contoh lengkap Pemrograman port untuk menggunakan LCD.

```
/*  Pemrograman Port LCD                                          */
/*  Register select harus dihubungkan ke select Printer pada Pin 17   */
/*  Enable harus dihubungkan ke Strobe pada (PIN1)                */
/*  DATA 0:7 dihubungkan ke  DATA 0:7 pada pin 2- pin 9            */

#include <dos.h>
#include <string.h>

#define alamatportdasar 0x378  /* Memasukkan alamat dasar port LPT yang dipakai

#define DATA alamatportdasar+0
#define STATUS alamatportdasar+1
#define KONTROL alamatportdasar+2

void main(void)
{
```

```
 char string[] = {" Inst. Elektronika FI UNPAD    "};
 char init[20];
 int cacahan;
 int len;
 init[0] = 0x0F; /* Inisialisasi  Tampilan  */
 init[1] = 0x01; /* Membersihkan Tampilan  */
 init[2] = 0x38; /* Dual Line / 8 Bits */

outportb(KONTROL, inportb(KONTROL) & 0xDF); /* Reset Control Port */
outportb(KONTROL, inportb(KONTROL) | 0x08); /* Set Register Select */

 for (cacahan = 0; cacahan <= 2; cacahan++)
  {
   outportb(DATA, init[count]);
   outportb(KONTROL,inportb(KONTROL) | 0x01); /* Set Strobe (Enable)*/
   delay(20);                          /* Penggunaan delay untuk menunggu proses oleh LCD */
   outportb(KONTROL,inportb(KONTROL) & 0xFE); /* Reset Strobe (Enable)*/
   delay(20);                          /* Penggunaan delay untuk menunggu proses oleh LCD */
  }

 outportb(KONTROL, inportb(KONTROL) & 0xF7);  /* Reset Register Select */
 len = strlen(string);

 for (cacahan = 0; cacahan < len; cacahan++)
  {
   outportb(DATA, string[count]);
   outportb(KONTROL,inportb(KONTROL) | 0x01); /* Set Strobe */
   delay(2);
   outportb(KONTROL,inportb(KONTROL) & 0xFE); /* Reset Strobe */
   delay(2);
  }
}
```

Seperti telah dijelaskan pada tahapan-tahapan insialisasi LCD sebelumnya, untuk dapat menampilkan karakter pada LCD Anda harus mengirim beberpa instruksi. Hal ini dikerjakan pada *for* pertama. Instruksi ini dikirimkan ke LCD pada *Instruction Register* yang dikontrol oleh *Register Select* pada Pin 4. Ketika pin 4 rendah instruksi register diambil dan ketika tinggi register data harus diambil. Instruksi ini kita hubungkan ke Port Parallel *Select Printer* yang kadang diinversi oleh hardware, sehingga Anda perlu menuliskan logika '1' ke bit 3 pada Control Register agar *Select Printer* menjadi rendah.

Agar Anda dapat mengirimkan instruksi ke modul LCD, maka *Register Select* harus dibuat rendah. Hal ini dapat dilakukan dengan mengeset bit 3 dari Register Kontrol menjadi '1'. Tentu saja kita tidak ingin register ini diset ulang pada saat pengirimin bit lain pada Port Registre Kontrol. Cara menaggulanginya adalah dengan membaca Port kontrol dan melakukan operasi OR dengan 0x80, berikut contoh programnnya

```
outportb(KONTROL, inportb(KONTROL) | 0x08);
```

Operasi ini hanya akan mengset bit 3 pada Register Kontrol. Setelah Anda menempatkan data byte pada baris data, Anda harus mengirimkan sinyal enable agar modul LCD membaca

data. Data tersebut kemudian dikirm ke modul LCD dengan pewaktuan (timing) dari transisi tinggi ke rendah. *Strobe* diinversi secara hardware yaitu dengan mengeset bit 0 pada Register Kontrol sehingga Anda mendapatkan transisi tinggi ke rendah pada baris *Strobe.* Kemudian kita menunggu delay dan kembali ke level tinggi untuk pengiriman byte berikutnya.

Setelah Anda menginisilisasi dan mengirimkan text ke modul LCD dengan karakter yang dikirim melalui *Data Port* pada LCD, Anda perlu menjadikan bit 3 nol. Untuk itu kita bisa menggunkan perintah

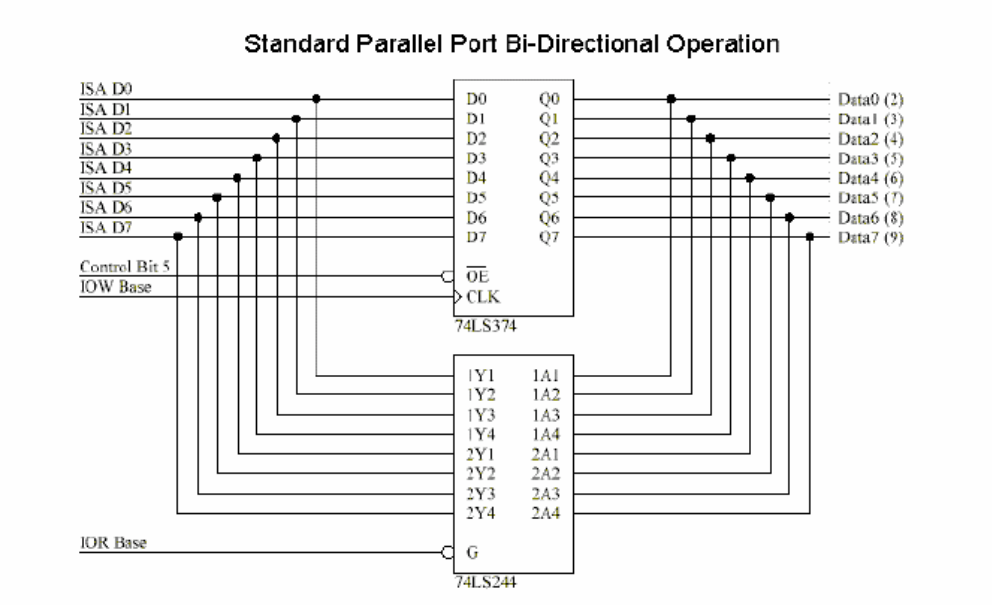
```
outportb(CONTROL, inportb(CONTROL) & 0xF7);
```

Langkah berikutnya adalah mengatur satu loop *for* untuk membaca byte dari string dan mengirimnya ke panel LCD. Langkah ini diulang sepanjang string.

Dilay yang Anda berikan harus cocok dengan LCD yang digunkan. Jika ternyata panel tidak melakukan inisialisasi maka Anda dapat menaikkan delay. Sebaliknya jika panel melompatkan karakter atau mengulang-ulang karakter mungkin Anda gagal dalam koneksi *Enable.* Anda bisa cek koneksi *Enable* ke *Strobe.*

8. PORT PARALEL DWI-ARAH

Gambar 29. skematik di bawah menujukkan penyederhanaan Regewster Data dari Port parallel. Awalnya memang port Parallel digunakan dalam bentuk kartu dengan menggunakan komponen digital 74LS. Saat ini semuannya dibuat dalam satu ASIC, tetapi teori pengoperasiannya tetap sama.

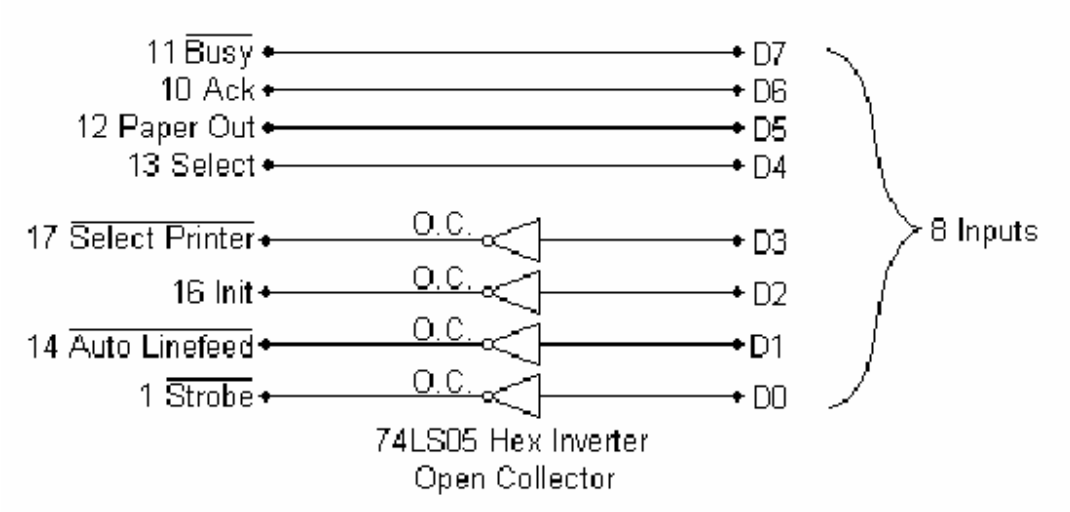


Gambar 29. Rangkaian Port dwi-arah

Port yang buak dwi-arah dibuat dengan keluaran enable 74LS374 dipaksa secara permanent dalam logika rendah, sehingga port data selalu hanya berupa keluaran. Jika Anda membaca register data pada port parallel, datanya berasala dari 74LS374 yang juga terhubung pin-pin data. Seandanya Anda mampu mendrive '374 , Anda akan dapat mengunhakannya sebagai port dwi-arah atau hanya sebagai input saja. Untuk mendrive ini Anda dapat melakukannya dengan menambahkan transisitor pada setiap pin input.

8.1. Mengunakan Port Parallel untuk Input 8 Bit

Ketika kita menggunakan suatu perangkat konversi data dari analog ke digital atau ADC untuk mengakusisi suatu data maka kita sangat membutuhkan fiditas data denagn mode port dwi-arah. Akan tetapi jika port parallel yang ada tidak mendukung untuk mode dwi-arah, Anda jangan kawatir. Anda dapat maksimum memasukkan 9 bit suatu waktu tertentu. Untuk melakukan hal ini guanakan 5 kanal input dari Port Status dan 4 kanal input (open collector) port Control.



Gambar 30. Rangkaian Port 8 input

Anda harus memilih Input port parallel sedemikan rupa sehingga memudahkan dalam pemrograman. Penempatan pin-pinnya seperti pada Gambar 30.
Berdasarkan desain koneksi seperti di atas, sekarang kiat membuat programnya. Hal pertama yang harus dilakukan adalah mengirimkan xxxx0100 ke port Control Port. Hal ini akan menyiapkan pin port control dalam logika tinggi, sehingga pin-pin ini siap untuk sebagai penginput data.

```
outportb(CONTROL, inportb(CONTROL) & 0xF0 | 0x04);
```

Selanjutnya kita dapat membaca data most significant nibble dari port status. Karena kita mengingkan hanya data MSnibble, manupulasi logika digital yang dapat kita lakukan adalah dengan melakukan operasi AND pada hasilnya dengan 0xF0, dengan demikian LSnibble-nya nol. Busy bersifat terinversi secara hardware, tetapi gak perlu dipermasalahkan karena kita sudah melakukan operasi AND di atas.

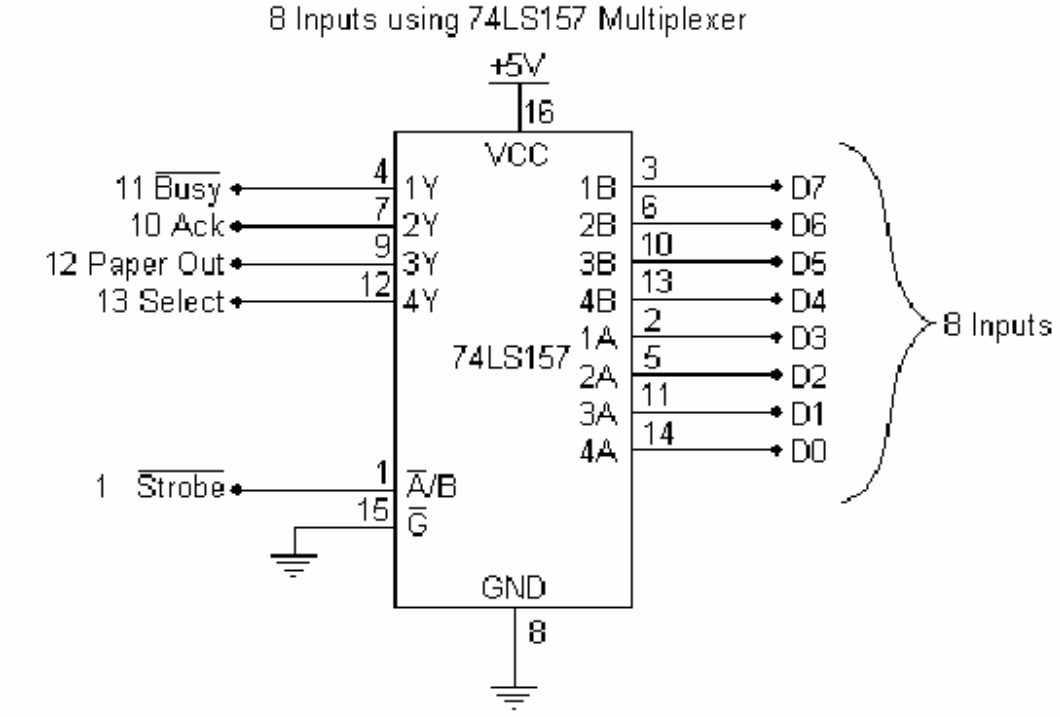
```
a = (inportb(STATUS) & 0xF0); /* Read MSnibble */
```

Sekarang kiat dapat membaca LSnibble yang berada pada port control. Karena ita tidak memerlukan MSnibblenya sehingga kita melakukan operasi AND pada hasilnya denga 0x0F untuk membuat nol MSnibble. Selanjunta adalah melakukan pemnggabungan kedua byte ini untuk mendapatkan 8 bit input. Hal ini dapat dilakukan dengan melakukan operasi OR terhadap kedua byte tersebut. Apakah data sudah falid? Kaena Bits 2 dan 7 adalah inversi, jadi data belum falid. Untuk itu kita dapat melakukan operasi OR lagi pada byte intu dengan 0x84, yang menjadikan kedua bit itu togel.

```
a = a |(inportb(CONTROL) & 0x0F); /* Read LSnibble */
a = a ^ 0x84; /* Toggle Bit 2 & 7 */
```

8.2. Mode Nibble

Mode Nibble digunakan untuk membaca data input 8 bit tanap menggunakan port yang dalam mode inverse, melaikan mengguakana kanal data. Mode Nibble menggunakan multiflekser 2 kanal ke 1 kanal untuk membaca satu nibble data pada satu pewaktuan. Kemudain selanjutnya terswitch ke nibel yang lainnya dan kita menjadi lebih mudah membacanya secara program. Program hanya perlu merekonstruksi kembali nible-nible itu menjadi data 8 bit atau byte. Kekurangan desain perangkat keras seperti ini adalah sedikit agak lambat. Sekarang kita hanya membutuhkan beberapa isntruksi I/O untuk membaca data itu, tetapi sebagai konsekwensinya kita harus menambah biaya untuk membeli IC eksternal berupa multiplekser.



Gambar 31. Rangkaian Port 8 input menggunakan 74LS157

Cara kerja multiplekser 74LS157 kanal 2 ke kanal 1 cukup sederhana. IC ini berprilaku seperti empat buah switch atau saklar. Ketyika input A/B rendah, input A yang terpilih , misalnya 1A dilewatkan ke 1Y, 2A dilewatkan ke 2Y dan seterusnya. Ketika A/B tinggi, input B yang diambil.
Output-Output Y dihubungkan ke port satatus pada port parallel, dengan desain seperti ini port ini merepresentasikan MSnibble dari register status.
Langkah-langkah dalam pengambilan data pada port tersebut diawali dengan penginisilisasian multiplexer untuk menswitch slah satu input A atau B. Selanjunya adalah membaca LSnibble pertama, kemudian kita membuta A/B rendah. Strobe mempunyai sifat terinversi secara hardware, sehingga kita harus mengeset Bit 0 pada port control untuk mendapatkan logika rendah pada Pin 1.

```
outportb(KONTROL, inportb(KONTROL) | 0x01); /* Select Low Nibble (A)*/
```

Sesudah low nibble terpilih, kita membaca LSnibble dari Port Status. Kita perlu mengingat bahwa Busy Line bersofat inversi, sehingga kita perlu melakukan operasi AND dari hasilnya denagn 0xF0 untuk membuat nol pada LSnibble.

```
a = (inportb(STATUS) & 0xF0); /* Read Low Nibble */
```

Sekarang tinnga kita geser nibble yang baru dibaca ke LSnibble yang ada di variabel a,

```
a = a >> 4; /* Menggeser 4 Bit ke kanan */
```

Kita baru mendapatkan setengah dari data, sehinga perlu selanjunta mendapatkan MSnibble. Untuk itu kita harus meswitch multiplexer untuk medapatkan input B. Kemudian kita dapat membaca MSnibble dan mengabungkan kedau nible tersebut menjadikannya 8 bit input atau satu byte,

```
outportb(CONTROL, inportb(CONTROL) & 0xFE); /* memilih High Nibble (B)*/
a = a |(inportb(STATUS) & 0xF0); /* Membaca High Nibble */
byte = byte ^ 0x88;
```

Dua kanal yang lain yang bersifat inverse dibaca pada kanal Busy. Untuk itu kita perlu memberikan proses delay, jika terjadi kesalahan hasil pembacaan.

9. PEMROGRAMAN PORT PARALEL MENGGUNAKAN FUNGSI DOS DAN BIOS

9.1. Pemrograman Port Paralel Menggunakan Fungsi DOS dan BIOS

Kita juga dapat menggunakan perintah fungsi DOS untuk secara mudah mengirim karakter atau data ke port paralel. Perintah fungsi tersebut adalah prints( ) yang merupakan fungsi DOS ke 5. Cobalaha program berikut ini

```
# include <dos.h>

/* mengirim karakter ke port parallel atau printer* /

void prits(char *s)
```

```
{
while( *s ) bdos(0x5,*s++,0);
}
```

Jika ingin, Anda juga dapat menggunakan funsgi library dari BIOS dari pada menggunakan rutin DOS. Bentuk umum perintahnya dalah

int **biosprint**(int *mode*, int *value*, int *port*)

dengan mode merupakan penentuan bagaimana biosprint bekerja. Jika *mode* adalah nol, biosprint( ) mengirim *value* ke port parallel, Jika *mode* adalah 1, adalah menginisilasiasi port parallel dan jika *mode* adalah 2, adalah kembali ke status printer. Sedangkan port adalah spesifikasi port parallel yang digunakan yaitu 0 untuk LPT1 dan 1 untuk LPT2. Funsgi biosprint( ) ini ada di bios.h

9.2. Pemrograman Port Serial Menggunakan Fungsi DOS dan BIOS

Untuk mengakses port serial kita dapat mengguinakan fungsi DOS yang ke 3 untuk membaca data dan ke 4 untuk mengirinkam data. Berikut ini contoh program untruk mengakses port serila

```
# include <dos.h>
/* mengirim karakter ke port serial * /
void put_async(char ch)
{
        bdos(0x4,ch,0);
}
/* mebaca karakter dari port serial * /
void get_async( )
{
        return((char) bdos(0x3,0,0);
}
```

Fungsi ini secara otomatis menggunakan port serial *default*. Jika Anda melakukan pengantian port pada MODE command, fungsi ini secara otomatis menggunakan port yang baru. Kita juga dapat menggunakan rutin dari BIOS yaitu fungsi **bioscom**() dengan bentuk umumnya

int **bioscom**(int *mode*, char *val*, int *port*)

dengan mode menujukkan operasi dari bioscom() sebagai berikut:

Mode	Fungsi
0	Inisalisasi Port
1	Mentrasmisikan satu byte
2	Menerima satu byte

3 Mengembalikan status Port

Port serial yang akan diakses ditentukan oleh *port* dengan 0 menandakan COM1 dan 1 menandakan COM2 dan seterusnya. Fungsi ini ada di dalam **biso.h**

Daftar Pustaka

Herbert Schildt.1986. Advance C. Osborne McGrawHill, Berkeley California.ISBN 0-07-881208-9.

Joseph S. Byrd & Roberto O. Pettus. 1993. Micro Computer Sysytem: Architecture and programming. Prentice Hall.USA. ISBN:0-13-030685-1 hal 275-326

Joseph Williams. 1997. An introduction to Computing Infrastructure: Hardware and Operating Sysytem.Que Education &Training.USA. ISBN:1-57576-355-9. hal 135-150.

Herbert Schildt.1998. C++: The Complete Reference. 3rd edition. Tata McGrawHill, New Delhi.ISBN 0-07-463880-7.