UNIVERSITAS INDONESIA

PERANCANGAN KENDALI PID UNTUK MOTOR DC MENGGUNAKAN MIKROKONTROLER H8/3052

SKRIPSI

TOHA KUSUMA
04 04 03 715 Y

FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK ELEKTRO
DEPOK
DESEMBER 2008

UNIVERSITAS INDONESIA

PERANCANGAN KENDALI PID UNTUK MOTOR DC MENGGUNAKAN MIKROKONTROLER H8/3052

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana

TOHA KUSUMA
04 04 03 715 Y

FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK ELEKTRO
DEPOK
DESEMBER 2008

LEMBAR PENGESAHAN

Skripsi dengan judul :

PERANCANGAN KENDALI PID UNTUK MOTOR DC MENGGUNAKAN MIKROKONTROLER H8/3052

Dibuat untuk melengkapi sebagian persyaratan menjadi Sarjana Teknik pada Program Studi Teknik Elektro, Departemen Teknik Elektro, Fakultas Teknik Universitas Indonesia dan disetujui untuk diajukan dalam presentasi skripsi.

Depok, 12 Desember 2008

Dosen Pembimbing,

(Dr. Ir.Wahidin Wahab, M.Sc)

NIP.

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya sendiri,

dan semua sumber baik yang dikutip maupun dirujuk

telah saya nyatakan dengan benar.

Nama	**: Toha Kusuma**
NPM	**: 04 04 03 715 Y**
Tanda Tangan	**:**
Tanggal	**: 12 Desember 2008**

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh :

Nama	: Toha Kusuma
NPM	: 040403715Y
Program Studi	: Teknik Elektro
Judul Skripsi	: Perancangan Kendali PID untuk Motor DC Menggunakan Mikrokontroler H8/3052

Telah berhasil dipertahankan di hadapan Dewan Penguji dan diterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Elektro, Fakultas Teknik, Universitas Indonesia.

DEWAN PENGUJI

Pembimbing :

Penguji :

Penguji :

Ditetapkan di :

Tanggal :

UCAPAN TERIMA KASIH

Puji syukur saya panjatkan kepada Allah SWT, karena atas berkat dan rahmat-Nya, saya dapat menyelesaikan skripsi ini. Penyusunan skripsi ini dilakukan dalam rangka memenuhi salah satu syarat untuk mencapai gelar Sarjana Teknik Departemen Teknik Elektro pada Fakultas Teknik Universitas Indonesia.

Saya menyadari bahwa tanpa bantuan dan bimbingan dari berbagai pihak, baik dari masa perkuliahan sampai pada penyusunan skripsi ini sangatlah sulit bagi saya untuk menyelesaikan skripsi ini. Untuk karena itu, saya mengucapkan terima kasih kepada :

(1) Dr. Ir. Wahidin Wahab, M.Sc selaku dosen pembimbing yang telah menyediakan waktu, tenaga dan pikiran di dalam mengarahkan saya dalam penyusunan skripsi ini;

(2) Dr. Abdul Muis, ST, M.Eng selaku dosen pembimbing kedua yang juga telah menyediakan waktu, fasilitas, tenaga, dan pikiran di dalam mengarahkan saya dalam penyusunan skripsi ini;

(3) Orang tua dan keluarga saya yang telah memberikan bantuan dukungan moral maupun material; dan

(4) Anggi Purwanto dan Hartanto Raharjo yang sudah memberikan bantuan dalam penyusunan Bab II dan III skripsi ini.

Akhir kata, semoga Allah SWT berkenan membalas semua kebaikan pihak-pihak yang membantu selesainya penyusunan skripsi ini. Semoga skripsi ini bermanfaat bagi pengembangan ilmu pengetahuan.

Depok, 12 Desember 2008

Penulis

HALAMAN PERNYATAAN PERSEUJUAN PUBLIKASI

TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Universitas Indonesia, saya yang bertanda tangan dibawah ini:

Nama : Toha Kusuma

NPM : 040403715Y

Program Studi : Teknik Elektro

Departemen : Teknik Elektro

Fakultas : Teknik

Jenis karya : Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Universitas Indonesia **Hak Bebas Royalti Noneksklusif *(Non-exclusive Royalty-Free Rights)*** atas karya ilmiah saya yang berjudul:

Perancangan Kendali PID untuk Motor DC Menggunakan Mikrokontroler H8/3052

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/format-kan, mengelolanya dalam bentuk pangkalan data (database), merawat, dan mempublikasikan tugas akhir saya tanpa perlu meminta izin dari saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di: Depok

Pada tanggal: 12 Desember 2008

Yang menyatakan

(Toha Kusuma)

ABSTRAK

Nama : Toha Kusuma

Program Studi : Teknik Elektro

Judul : Perancangan Kendali PID untuk Motor DC Menggunakan Mikrokontroler H8/3052

Pengendalian posisi dan kecepatan motor DC sangat penting untuk kendaraan pada umumnya dan robotik pada khususnya. Pada studi ini mempresentasikan pengendalian motor DC dengan algoritma kendali PID menggunakan mikrokontroler H8/3052. Pengendalian posisi dan kecepatan dengan menggunakan sinyal PWM yang dihasilkan mikrokontroler. Untuk mengatur perputaran motor digunakan rangkaian optoisolator dan H-bridge. Sinyal umpan balik dihasilkan dari rotary encoder EC16B berupa umpan balik posisi sudut dan pada mikrokontroler didiferensialkan menjadi kecepatan sudut. Perbedaan nilai antara setpoint dengan nilai encoder akan menghasilkan sinyal error. Program pengendali pada mikrokontroler selanjutnya akan menangani sinyal error tersebut untuk dikendalikan.

Kata kunci :

Motor DC, mikrokontroler H8/3052, kendali PID

ABSTRACT

Name : Toha Kusuma

Study Program: Electrical Engineering

Title : Robust DC Motor Control with Microcotroler H8/3052

DC motor speed and position controls are fundamental in vehicles in general and robotics in particular. This study presents the DC motor control with PID control using microcontroller H8/3052. Microcontroller uses PWM signals to control the position and speed of DC motor. For driving the motor, the optoisolator and H-Bridge circuits are used. Feedback signal is generated by rotary encoder EC16B that generates position feedback and in microcontroller those feedback will be differrentiated to be the angle velocity. The differences between setpoint number with the feedback from encoder will generate the error signal. Then the program on microcontroller will handle this error to be controlled.

Keywords:

DC motor, microcontroller H8/3052, PID control

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

DAFTAR SINGKATAN

A/D	*Analog to Digital*
ADC	*Analog to Digital Converter*
CPU	*Central Processing Unit*
D/A	*Digital to Analog*
DC	*Direct Current*
I/O	*Input/Output*
ITU	*Integrated Timer Unit*
LSB	*Least Significant Bit*
MCU	*Microcontroller Unit*
MSB	*Most Significance Bit*
NMI	*Non Maskable Interrupt*
PC	*Personal Computer*
PWM	*Pulse Width Modulation*
SCI	*Serial Communication Interface*
TCLK	*Timer Clock*

DAFTAR ISTILAH

Bit	digit dalam sistem bilangan biner, dapat bernilai 0 atau 1
Byte	seri 8-bit
Duty cycle	perbandingan antara panjang sinyal on dengan panjang satu gelombang pada sinyal PWM
Flag	satu atau lebih bit yang digunakan untuk menyimpan nilai biner atau kode yang menunjukkan kondisi tertentu
Least Significant Bit	bit dengan nilai pangkat terendah dalam satuan bit
Mikrokontroler	*single purpose processing unit* yang didesain untuk pengendalian kecil, kadang *real time*
Motor	aktuator yang menggunakan tegangan input sebagai pengendali
Pulse Width Modulation	tipe gelombang segi empat yang dimodulasi terhadap lebar sinyal
Interrupt	sinyal asinkron yang dikirimkan ke prosesor yang menunjukkan perlunya penanganan dari prosesor

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Motor DC merupakan aktuator yang sangat lazim digunakan. Ada berbagai macam alasan mengapa motor DC sangat populer digunakan. Salahsatunya adalah sistem tenaga listrik DC masih umum digunakan pada industri, automobil , dan robotika. Dan meskipun tidak ada sumber tenaga listrik DC, rangkaian penyearah dan chopper dapat digunakan untuk menghasilkan sumber listrik DC yang diinginkan. Motor DC juga digunakan karena kebutuhan akan variasi kecepatan motor yang lebar.

Dalam dunia industri, pengendalian posisi dan kecepatan motor DC sangat penting. Misalnya pada industri plastik. Pada proses penggulungan plastik, kecepatan penggulungan plastik harus disesuaikan dengan kecepatan mesin pengirim plastik dan juga disesuaikan dengan jari-jari gulungan. Jika tidak maka hasil gulungan plastik tidak rapi atau kusut.

Pada robotika pengendalian posisi dan kecepatan motor DC juga sangat penting misalnya dalam Kontes Robot Indonesia (KRI) dan Kontes Robot Cerdas Indonesia (KRCI). Robot harus dapat bergerak cepat dan tepat, meskipun terdapat berbagai halangan ataupun gangguan. Karena itu pergerakan robot memerlukan pengaturan posisi dan kecepatan motor yang baik agar tujuan yang diinginkan dapat tercapai.

Karena itulah kendali PID diperlukan disini yaitu untuk mengendalikan posisi dan kecepatan motor DC. Pengendali PID merupakan pengendali yang umum digunakan dalam berbagai macam proses industri. Popularitas pengendali PID disebabkan khususnya karena performansinya yang baik dalam jangkauan yang lebar dari berbagai kondisi operasi dan khususnya dalam kesederhanaan fungsi PID, yang memungkinkan engineer untuk mengoperasikannya secara simpel dan langsung. Untuk mengimplementasikan pengendali PID, tiga parameter harus ditentukan pada proses yang dikendalikan yang meliputi proportional gain, integral gain, dan derivative gain.

1.2 Tujuan Penulisan

Tujuan penulisan skripsi ini yaitu untuk merancang suatu pengendali motor DC dengan kendali PID berbasis mikrokontroler H8/3052 dengan PC sebagai pemberi *set point*, pengukur data, dan penyimpan data.

1.3 Pembatasan Masalah

Penulisan skripsi ini dibatasi pada pengendalian posisi dan kecepatan motor DC menggunakan feedback encoder dengan hasil yang didapatkan memenuhi kriteria yang diinginkan. Pengendalian dilakukan dengan sistem pengendali PID. Pengendali tersebut diharapkan dapat diaplikasikan untuk semua range posisi atau kecepatan. Pengendalian tersebut diharapkan menghasilkan sebuah sistem yang mempunyai persen *overshoot* kecil, *settling time* yang cepat, dan nilai *steady-state error* mendekati nol.

1.4 Metode Penelitian

Metode penelitian yang digunakan dalam penyusunan skripsi ini meliputi:

1. Pendekatan studi pustaka, yaitu dengan melakukan studi literatur dari buku-buku pustaka, referensi yang ada di internet, dan *manual book* atau *datasheet* dari suatu piranti.
2. Pendekatan diskusi dengan pembimbing skripsi.
3. Perancangan perangkat keras dan perangkat lunak.
4. Pengujicobaan.

1.5 Sistematika Penulisan

Agar pembahasan masalah pada skripsi lebih sistematis, maka skripsi ini dibagi menjadi beberapa bab.

Bab Pertama, Pendahuluan, meliputi latar belakang, tujuan penulisan, pembatasan masalah, metode penelitian, dan sistematika penulisan. Bab Kedua, membahas mengenai mikrokontroler H8/3052, dan fitur-fitur pendukung mikrokontroler meliputi ITU, port I/O, SCI, dan Interrupt Controller. Bab Ketiga, menjelaskan tentang perancangan kendali PID motor DC yang terdiri atas perancangan motor DC, perancangan blok kendali, perancangan kendali PID, perancangan perangkat lunak, serta perancangan perangkat keras. Bab Keempat menuliskan pengujian dan analisa dari percobaan yang dilakukan. Bab Kelima adalah kesimpulan dari skripsi.

BAB 2

MIKROKONTROLER H8/3052

Mikrokontroler yang digunakan dalam skripsi ini adalah H8/3052F, yaitu seri mikrokontroler (MCU) yang mengintegrasikan fungsi-fungsi sistem pendukung dengan sebuah inti CPU H8/300 yang mempunyai arsitektur asli Hitachi. Mikrokontroler H8/3052 memiliki fitur yang cukup lengkap seperti dideskripsikan pada Tabel 2.1 berikut.

Tabel 2.1 Fitur H8/3052F

Fitur	Deskripsi
CPU	a. Mempunyai 16 general register 16-bit (juga dapat digunakan sebagai 8 register 32-bit) b. Operasi kecepatan tinggi dengan pewaktuan maksimum 25MHz, operasi penjumlahan/ pengurangan 80ns,perkalian/pembagian 560ns, 16 Mbyte ruang pengalamatan c. Berbagai fitur instruksi :data transfer, aritmetika, logika, perkalian/pembagian bilangan bertanda dan tidak bertanda, akumulator bit, dan manipulasi bit.
Memori	Memori flash 512kbytes, RAM 8kbytes
Interrupt Controller	7 Pin interrupt eksternal: NMI, $\overline{IRQ}_0$ sampai dengan $\overline{IRQ}_5$, 30 intrerrupt internal, 3 level prioritas interrupt yang dapat dipilih

Tabel 2.1 Fitur H8/3052 (lanjutan)

Fitur	Deskripsi
16-bit Integrated Timer Unit (ITU)	a. 5 kanal timer 16-bit, yang dapat memproses sampai 12 pulsa output atau 10 pulsa input b. Timer counter 16-bit pada setiap kanal (kanal 0 – 4) c. Operasi dapat disinkronisasi d. Mode PWM dapat beroperasi pada tiap kanal e. Mode *Phase counting* dapat beroperasi pada kanal 2 f. Buffering pada kanal 3 dan 4 g. Mode *Reset Syncronized* PWM pada kanal 3 dan 4 h. Mode *Complementary* PWM pada kanal 3 dan 4
Watch Dog Timer (WDT), 1 kanal	a. Sinyal reset dapat dihasilkan dengan overflow b. Dapat digunakan sebagai interval timer
Serial Communication Interface (SCI), 2 kanal	a. Pemilihan mode sinkron atau asinkron b. Full duplex: dapat mengirim dan menerima secara simultan c. Terdapat dua kanal independen (SCI0 dan SCI1)
Konversi A/D	Resolusi 10 bit, 8 kanal, mode single dan scan, range konversi tegangan analog yang beragam, dan fungsi *sampling* dan *hold*.
Konversi D/A	Resolusi 8-bit, 2 kanal
I/O	70 pin yang dapat berfungsi sebagai pin input atau output, dan 9 pin yang berfungsi sebagai pin input saja.

2.1 Integrated Timer Unit (ITU)

ITU merupakan timer terintegrasi pada mikrokontroler H8/3052. H8/3052 mempunyai ITU 16-bit yang sudah terpasang tetap, kanal timer 16-bit. Beberapa fitur ITU dapat dijelaskan sebagai berikut:

a. Dapat memproses hingga 12 pulsa output dan 10 pulsa input.
b. Mempunyai 10 *General Register* (GRs dua tiap kanal) yang dapat berfungsi sebagai *output compare* atau *input capture*.
c. Terdapat 8 jenis sumber *counter clock* untuk tiap kanal yaitu *clock* internal ϕ, $\phi/2$, $\phi/4$, $\phi/8$ *clock* eksternal: TCLKA, TCLKB, TCLKC, TCLKD.
d. Lima jenis operasi pada tiap kanalnya yaitu, *waveform output by compare match, input capture, counter clearing, synchronization,* dan mode PWM.
e. Terdapat sebuah mode tambahan pada kanal 2 yaitu mode *Phase Counting* dan tiga buah mode tambahan pada kanal 3 dan 4, yaitu mode *Reset-synchronized* PWM, mode *Complementary* PWM dan *Buffering*.
f. Mempunyai 15 sumber interrupt.
g. Akses dengan kecepatan tinggi via bus internal 16 bit

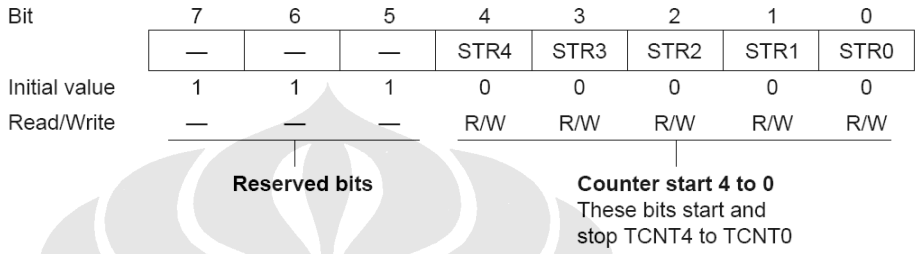
Selanjutnya akan dijelaskan mengenai register-register pada ITU, cara pengoperasian PWM dengan ITU, dan pemrograman ITU pada mikrokontroler.

2.1.1 Register-Register ITU

ITU memiliki register-register yang digunakan dalam operasi timer. Pada mikrokontroler H8/3052, terdapat 13 buah register yang mendukung operasi timer. Pada skripsi ini hanya digunakan 8 buah register yaitu TSTR, TMDR, TCNT, GR, TCR, TIOR, TSR, dan TIER.

a. *Timer Start Register* (TSTR)

TSTR adalah register baca/tulis 8-bit yang memulai dan memberhentikan *timer counter* (TCNT) pada kanal 0 sampai dengan kanal 4. Konfigurasi TSTR dapat dilihat pada Gambar 2.1.



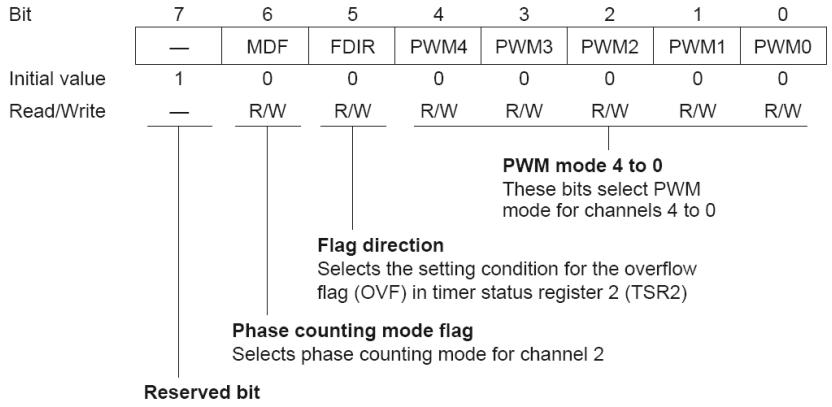
Gambar 2.1 Register TSTR

Nilai bit pada STR_0 sampai dengan STR_4 menentukan mulai atau berhentinya TCNT pada kanal yang dituju. Jika bit tersebut bernilai 1 maka TCNT mulai menghitung dan jika bit tersebut bernilai 0 maka timer berhenti menghitung. Pada skripsi ini, digunakan timer pada kanal 2 sehingga untuk memulai timer bekerja maka bit STR_2 diset nilainya menjadi 1. Dengan demikian pada program mikrokontroler dituliskan sebagai berikut.

```
ITU.TSTR.BIT.STR2 = 1;          /* Start counting TCNT2*/
```

b. *Timer Mode Register* (TMDR)

TMDR adalah register baca/tulis 8-bit yang memilih mode PWM untuk kanal 0 sampai dengan kanal 4. Mode PWM ini yang nantinya menentukan kanal yang dipakai pada mode PWM dan pin mana yang menjadi output PWM. TMDR juga memilih mode *phase counting* dan mengatur kondisi flag *overflow* (OVF). Konfigurasi register TMDR dapat dilihat pada Gambar 2.2.



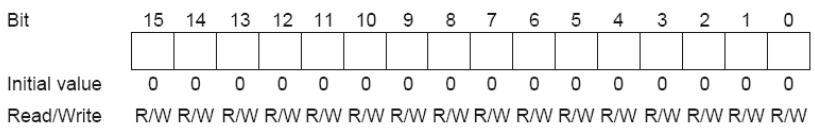
Gambar 2.2 Register TMDR

Mode PWM ditentukan dengan menset nilai bit mode PWM yang akan dipilih yaitu PWM_0 sampai dengan PWM_4 menjadi bernilai 1. Pada skripsi ini digunakan operasi PWM mode 2 yang berarti bahwa kanal 2 beroperasi dengan mode PWM dan pin $TIOCA_2$ menjadi pin output PWM. Untuk beroperasi dengan PWM mode 2, maka bit PWM_2 diset nilainya menjadi 1. Dengan demikian, pada program mikrokontroler dituliskan sebagai berikut.

```
ITU.TMDR.BIT.PWM2 = 1;          /*Select channel 2 in PWM mode */
```

c. *Timer Counter* (TCNT)

TCNT adalah register *counter* 16-bit. Register counter ini digunakan untuk pewaktuan pada mikrokontroler. Dengan ukuran register 16-bit maka TCNT dapat menghitung dari 0x0000 sampai dengan 0xffff. Konfigurasi register TCNT dapat dilihat pada Gambar 2.3.



Gambar 2.3 Konfigurasi Register TCNT

ITU mempunyai 5 buah TCNT, masing-masing satu TCNT untuk tiap kanal yaitu $TCNT_0$ sampai dengan $TCNT_4$. Jenis register TCNT dan fungsi yang dapat dilakukan dapat dilihat pada Tabel 2.2.

Tabel 2.2 Jenis dan Fungsi Register TCNT

Channel	Abbreviation	Function
0	TCNT0	Up-counter
1	TCNT1	
2	TCNT2	Phase counting mode: up/down-counter Other modes: up-counter
3	TCNT3	Complementary PWM mode: up/down-counter
4	TCNT4	Other modes: up-counter

TCNT mempunyai fungsi yaitu up-counter atau down-counter sesuai dengan mode PWM yang digunakan. Fungsi up-counter yaitu menghitung dari nilai kecil ke besar dari 0x0000 sampai dengan 0xffff. Sedangkan down-counter yaitu menghitung dari besar ke kecil dari 0xffff sampai dengan 0x0000. Pada skripsi ini digunakan TCNT2 dengan PWM mode 2, sehingga fungsi yang dijalankan yaitu up-counter. Nilai TCNT di-clear-kan dengan GRA output compare match.

d. *General Register* (GRA, GRB)

General register adalah register baca/tulis 16-bit yang dapat berfungsi baik sebagai *output compare register* atau *input capture register*. ITU mempunyai 10 *general register* yaitu masing-masing dua pada tiap kanalnya.

Tabel 2.3 Jenis dan Fungsi General Register

Channel	Abbreviation	Function
0	GRA0, GRB0	Output compare/input capture register
1	GRA1, GRB1	
2	GRA2, GRB2	
3	GRA3, GRB3	Output compare/input capture register; can be buffered by buffer registers BRA and BRB
4	GRA4, GRB4	

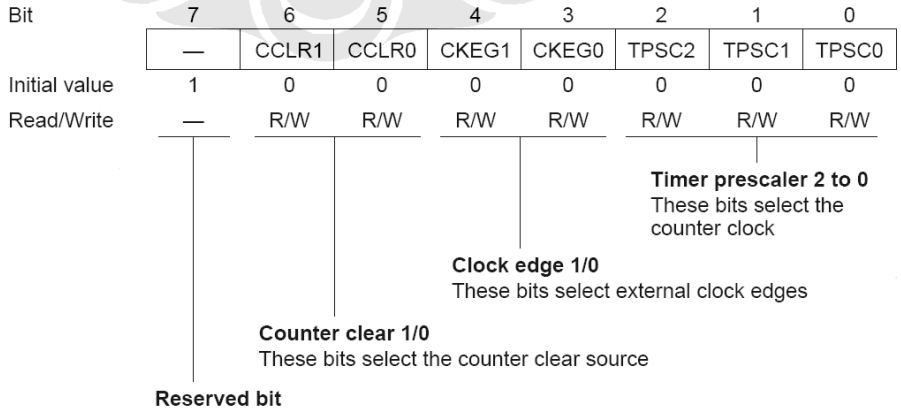
Pada skripsi ini digunakan GRA dan GRB compare match pada ITU2. General register berfungsi sebagai output compare register, nilai general register tersebut secara konstan dibandingkan dengan nilai TCNT. Ketika dua nilai tersebut sesuai (compare match), flag IMFA atau IMFB menjadi bernilai 1 pada TSR.

e. *Timer Control Register* (TCR)

TCR adalah register yang berfungsi untuk mengendalikan timer counter. Tiap TCR merupakan register 8-bit baca tulis yang dapat memilih sumber timer counter *clock*, memilih *clock* edge eksternal, dan memilih bagaimana counter di-clear-kan. ITU mempunyai 5 buah TCR, masing-masing satu buah pada tiap kanalnya.

Tabel 2.4 Jenis dan Fungsi Register TCR

Channel	Abbreviation	Function
0	TCR0	TCR controls the timer counter. The TCRs in all channels are functionally identical. When phase counting mode is selected in channel 2, the settings of bits CKEG1 and CKEG0 and TPSC2 to TPSC0 in TCR2 are ignored.
1	TCR1	
2	TCR2	
3	TCR3	
4	TCR4	



Gambar 2.4 Register TCR

Tabel 2.5 Bit Counter Clear $CCLR_{0,1}$

Bit 6: CCLR1	Bit 5: CCLR0	Description
0	0	TCNT is not cleared (Initial value)
	1	TCNT is cleared by GRA compare match or input capture*1
1	0	TCNT is cleared by GRB compare match or input capture*1
	1	Synchronous clear: TCNT is cleared in synchronization with other synchronized timers*2

Jika TCNT tidak di-clear-kan, maka TCNT akan kembali bernilai 0 ketika terjadi overflow yaitu setelah TCNT bernilai 0xffff maka TCNT kembali bernilai 0x0000. Jika TCNT di-clear-kan saat GRA/GRB compare match/input capture, maka saat terjadi GRA/GRB compare match atau input capture, nilai TCNT menjadi kembali ke 0x0000. TCNT di-clear-kan dengan compare match ketika general register berfungsi sebagai output compare register, dan di-clear-kan dengan input capture ketika general register berfungsi sebagai input capture register.

Bit clock edge CKEG1 dan CKEG0 berfungsi memilih edge input clock eksternal ketika sumber clock eksternal digunakan. Bit timer prescaler TPSC0 sampai dengan TPSC2 digunakan untuk memilih sumber counter clock.

Tabel 2.6 Setting Time Prescaler

Bit 2: TPSC2	Bit 1: TPSC1	Bit 0: TPSC0	Description
0	0	0	Internal clock: ϕ (Initial value)
		1	Internal clock: $\phi/2$
	1	0	Internal clock: $\phi/4$
		1	Internal clock: $\phi/8$
1	0	0	External clock A: TCLKA input
		1	External clock B: TCLKB input
	1	0	External clock C: TCLKC input
		1	External clock D: TCLKD input

Nilai bit pada TPSC2 menentukan apakah timer yang digunakan adalah timer eksternal atau internal. Nilai bit TPSC1 dan TPSC0 menentukan clock yang digunakan sesuai dengan yang tertera pada Tabel 2.6. Adapun nilai ϕ untuk H8/3052 adalah 25MHz.

Misalkan digunakan internal clock 25MHz dan TCNT di-clear-kan dengan GRA compare match, maka pada pemrograman dituliskan sebagai berikut.

```
ITU2.TCR.BIT.TPSC = 0; /*Select internal clock = 25Mhz*/

ITU2.TCR.BIT.CCLR = 1; /* Clear TCNT by GRA output compare match */
```

Nilai bit TPSC=0 berarti bahwa bit $TPSC_2$=0, $TPSC_1$=0, dan $TPSC_0$=0. Sedangkan nilai bit CCLR=1 berarti bahwa nilai $CCLR_1$=0 dan $CCLR_0$=1.

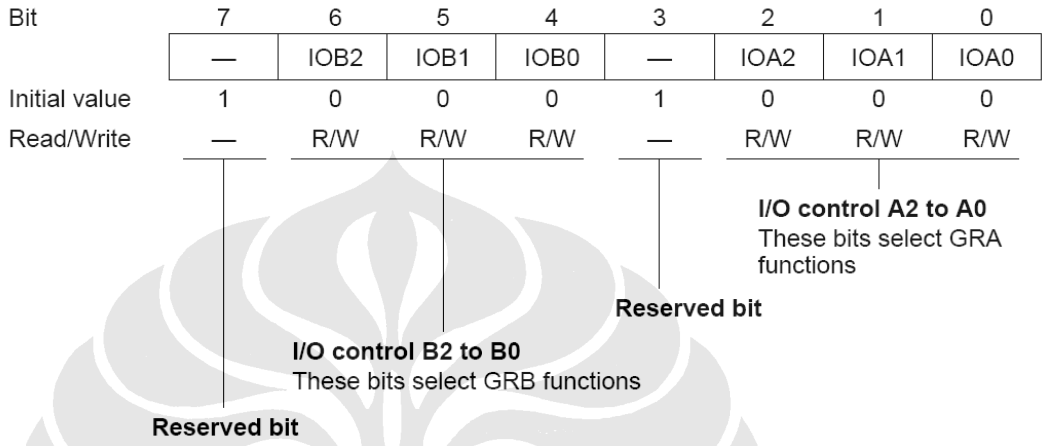
f. *Timer I/O Control Register* (TIOR)

TIOR adalah register 8-bit yang berfungsi untuk mengendalikan general register. Tiap TIOR adalah register baca tulis 8-bit yang memilih fungsi output compare atau input capture general register dan juga menentukan fungsi pin TIOCA dan TIOCB. Jika fungsi *output compare* yang dipilih, TIOR juga memilih tipe dari output. Jika fungsi *input capture* yang dipilih, TIOR juga memilih bentuk perubahan sinyal dari sinyal *input capture*.

Tabel 2.7 Jenis dan Fungsi Register TIOR

Channel	Abbreviation	Function
0	TIOR0	TIOR controls the general registers. Some functions differ in PWM mode. TIOR3 and TIOR4 settings are ignored when complementary PWM mode or reset-synchronized PWM mode is selected in channels 3 and 4.
1	TIOR1	
2	TIOR2	
3	TIOR3	
4	TIOR4	

ITU mempunyai 5 buah TIOR seperti tertera pada Tabel 2.7, masing-masing satu buah di setiap kanalnya. Gambar 2.5 menunjukkan konfigurasi dari register TIOR.



Gambar 2.5 Register TIOR

Bit IOB_0 sampai dengan IOB_2 memilih fungsi GRB. Sedangkan bit IOA_0 sampai dengan IOA_2 memilih fungsi GRA. Pemilihan bit-bit IOB dan IOA ditunjukkan pada Tabel 2.8 dan Tabel 2.9.

Tabel 2.8 Tabel Setting Bit I/O Control B_{0-2}

Bit 6: IOB2	Bit 5: IOB1	Bit 4: IOB0	Description	
0	0	0	GRB is an output compare register	No output at compare match (Initial value)
		1		0 output at GRB compare match*1
	1	0		1 output at GRB compare match*1
		1		Output toggles at GRB compare match (1 output in channel 2)*1, *2
1	0	0	GRB is an input capture register	GRB captures rising edge of input
		1		GRB captures falling edge of input
	1	0		GRB captures both edges of input
		1		

Tabel 2.9 Tabel Setting Bit I/O Control A_{0-2}

Bit 2: IOA2	Bit 1: IOA1	Bit 0: IOA0	Description	
0	0	0	GRA is an output compare register	No output at compare match (Initial value)
		1		0 output at GRA compare match*1
	1	0		1 output at GRA compare match*1
		1		Output toggles at GRA compare match (1 output in channel 2)*1, *2
1	0	0	GRA is an input capture register	GRA captures rising edge of input
		1		GRA captures falling edge of input
	1	0		GRA captures both edges of input
		1		

Pada skripsi ini digunakan output compare match yaitu output = 0 pada GRB compare match, dan output = 1 pada GRA compare match. Dengan demikian pada program dituliskan sebagai berikut.

```
ITU2.TIOR.BIT.IOB = 1; /* Output = 0 at GRB compare match */

ITU2.TIOR.BIT.IOA = 2; /* Output = 1 at GRA compare match */
```

g. *Timer Status Register* (TSR)

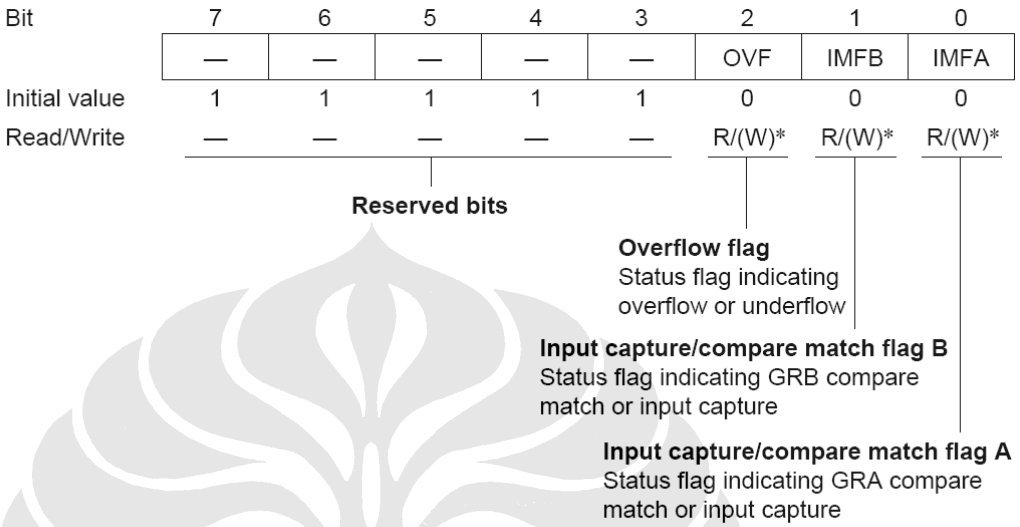
TSR adalah register 8-bit yang menunjukkan status input capture, compare match, maupun status overflow. ITU mempunyai 5 buah TSR, masing-masing satu buah di tiap kanalnya, seperti ditunjukkan pada Tabel 2.10.

Tabel 2.10 Jenis dan Fungsi Register TSR

Channel	Abbreviation	Function
0	TSR0	Indicates input capture, compare match, and overflow status
1	TSR1	
2	TSR2	
3	TSR3	
4	TSR4	

Tiap TSR adalah register baca/tulis yang mengandung flag yang menandakan bahwa TCNT *overflow* atau *underflow* serta GRA dan GRB *compare match* atau *input capture*. Flag-flag ini merupakan sumber interrupt dan menghasilkan interrupt CPU jika di-*enable* sesuai dengan pengaturan bit pada TIER. Penulisan

bit hanya bisa dengan nilai 0 yaitu untuk meng-clear-kan flag. Konfigurasi bit TSR dapat dilihat pada Gambar 2.6.



Gambar 2.6 Konfigurasi Register TSR

Bit OVF merupakan flag yang menunjukkan terjadinya overflow atau underflow yaitu saat OVF bernilai satu. Bit IMFA dan IMFB merupakan flag yang menunjukkan terjadinya output compare atau input capture pada GRA atau GRB, yaitu saat bit tersebut bernilai 1.

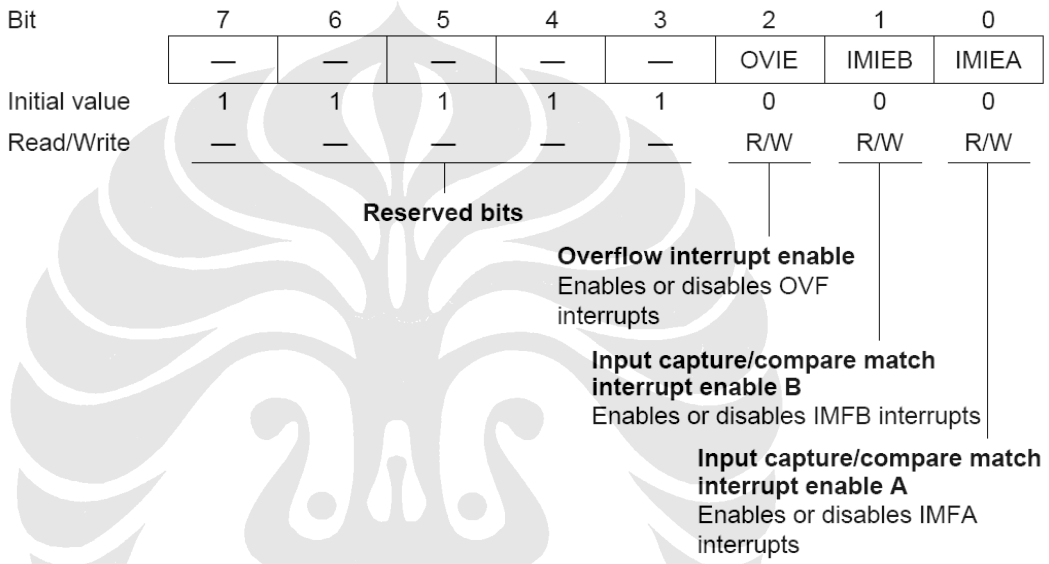
h. *Timer Interrupt Enable Register* (TIER)

TIER adalah register baca/tulis 8-bit yang memilih permintaan interrupt *overflow* dan permintaan interrupt *input capture* atau *output compare* dari general register. ITU mempunyai 5 buah TIER, masing-masing satu buah pada tiap kanalnya, seperti ditunjukkan pada Tabel 2.11.

Tabel 2.11 Jenis dan Fungsi Register TIER

Channel	Abbreviation	Function
0	TIER0	Enables or disables interrupt requests.
1	TIER1	
2	TIER2	
3	TIER3	
4	TIER4	

Gambar 2.7 menunjukkan setting konfigurasi bit TIER.



Gambar 2.7 Register TIER

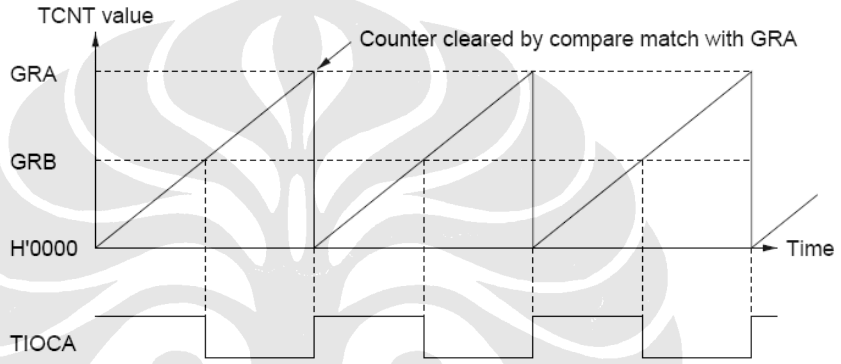
Seperti ditunjukkan pada Gambar 2.7, bit OVIE merupakan bit yang men-enable atau disable interrupt OVF. Bit IMIEB digunakan untuk meng-enable atau disable interrupt IMFB. Sedangkan bit IMIEA digunakan untuk meng-enable atau disable interrupt IMFA.

Misalnya digunakan interrupt IMFA yaitu saat terjadi output compare match pada GRA. Untuk meng-enable interrupt IMFA tersebut, maka pada program dituliskan sebagai berikut.

```
ITU2.TIER.BIT.IMIEA = 1 ; /* enable IMIA interrupt */
```

2.1.2 Operasi PWM

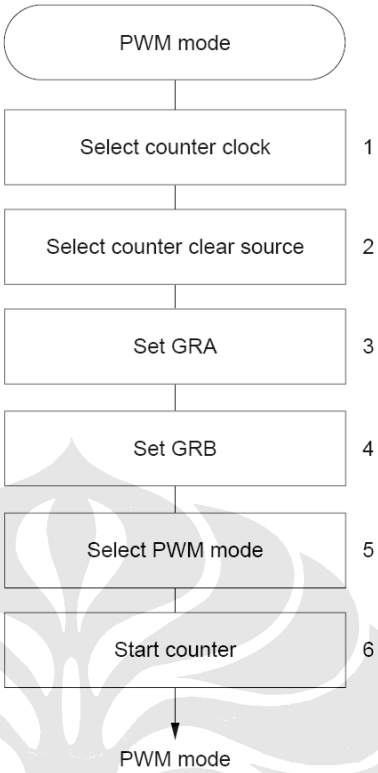
Pada operasi mode PWM, GRA dan GRB bekerja secara berpasangan dan gelombang PWM adalah output dari pin TIOCA. Output PWM menjadi 1 saat compare match dengan GRA dan menjadi 0 saat compare match dengan GRB. *Duty cycle* suatu gelombang PWM merupakan perbandingan antara nilai GRA dan GRB. Hal ini seperti ditunjukkan pada Gambar 2.8.



Gambar 2.8 Operasi PWM dengan TCNT Di-clear-kan dengan GRA Compare Match

Pada Gambar 2.8 ditunjukkan bahwa nilai TCNT berubah menjadi nol ketika terjadi GRA compare match. Hal ini terjadi karena TCNT di-clear-kan dengan GRA compare match. Nilai output PWM yaitu TIOCA, berubah menjadi 0 saat terjadi GRB compare match dan berubah menjadi 1 saat terjadi GRA compare match. Sedangkan duty cycle merupakan perbandingan lamanya sinyal on dengan panjang satu gelombang pada opersai PWM. Hal ini diwakili dengan perbandingan antara nilai GRB dengan nilai GRA tersebut.

Pada skripsi ini, digunakan operasi PWM mode 2 untuk menghasilkan output PWM yang digunakan untuk mengatur kecepatan motor. Contoh operasi mode PWM ditunjukkan pada diagram alir Gambar 2.9.



Gambar 2.9 Operasi PWM

Diagram alir pada Gambar 2.9 dapat dijilaskan sebagai berikut.

1. Pemilihan sumber counter clock yaitu dengan menset bit TPSC2 sampai dengan TPSC0 pada TCR.
2. Menset bit CCLR1 dan CCLR0 pada TCR untuk mengatur counter clear source.
3. Menset nilai GRA, yaitu waktu dimana sinyal PWM berubah menjadi 1.
4. Menset nilai GRB, yaitu waktu dimana sinyal PWM berubah menjadi 0.
5. Menset bit PWM pada TMDR untuk memilih mode PWM.
6. Menset bit STR pada TSTR ke 1 untuk memulai timer counter.

Operasi PWM tersebut diterjemahkan ke dalam bahasa pemrograman dengan contoh program sebagai berikut.

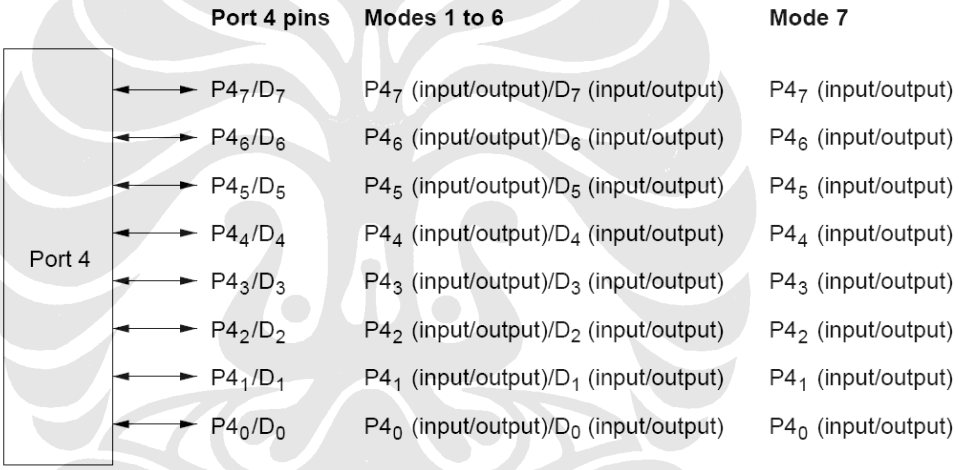
```
void initPWMsingle(void) {
      MSTCR.BIT._ITU = 0;           /* ITU operates normally */
      ITU.TOCR.BIT.XTGD = 1;    /* Disable external trigerring */
      ITU.TSTR.BIT.STR2 = 0;    /* Stop Counting TCNT in channel 2 */
      ITU2.TCR.BIT.TPSC = 3;    /*Select internal clock = 25Mhz*/
      ITU2.TCR.BIT.CCLR = 1;    /* Clear TCNT by GRA output compare match */
      ITU2.GRA = 0xF424;        /* GRA=62500 with pulse cycle approx 20 ms */
      ITU2.GRB = 0x0000;        /* Duty cycle 0%*/
      ITU2.TIOR.BIT.IOB = 1;    /* Output = 0 at GRB compare match */
      ITU2.TIOR.BIT.IOA = 2;    /* Output = 1 at GRA compare match */
      ITU.TMDR.BIT.PWM2 = 1;    /*Select channel 2 in PWM mode */
      ITU2.TIER.BIT.IMIEA = 1 ;     /* enable IMIA interrupt */
}
void main (void) {
      set_imask_ccr(1);             /* Disable all of the interrupts        */
      initPWMsingle();              /* Inisialisasi PWM                */
      set_imask_ccr(0);             /* Enable interrupts                    */

      ITU.TSTR.BIT.STR2 = 1;        /* Start counting TCNT2*/
      while(1) {
      }
}
```

2.2 Port Input/Output (I/O)

H8/3052 mempunyai 10 port input/output (port 1, 2, 3, 4, 5, 6, 8, 9, A, B) dan 1 port input (port 7). Setiap port memiliki sebuah *data direction register* (DDR) untuk pemilihan input atau output, dan sebuah *data register* untuk menyimpan data output. Pada skripsi ini port yang digunakan yaitu port 4 sebagai input untuk encoder dan port A untuk output PWM. Pada pembahasan selanjutnya hanya akan mengarah pada port 4.

Port 4 adalah port input/output 8-bit dengan konfigurasi pin seperti Gambar 2.10. Fungsi-fungsi pin berbeda berdasarkan pada mode operasi yang digunakan.



Gambar 2.10 Port 4

2.2.1 Konfigurasi Register I/O

Konfigurasi register port 4 dirangkum dalam Tabel 2.12 berikut.

Tabel 2.12 Register Port 4

Address*	Name	Abbreviation	R/W	Initial Value
H'FFC5	Port 4 data direction register	P4DDR	W	H'00
H'FFC7	Port 4 data register	P4DR	R/W	H'00
H'FFDA	Port 4 input pull-up MOS control register	P4PCR	R/W	H'00

Note: * Lower 16 bits of the address.

a. Port 4 Data Direction Register

P4DDR adalah regiter tulis 8-bit yang dapat memilih input atau output untuk tiap pin di port 4.

Bit	7	6	5	4	3	2	1	0
	$P4_7DDR$	$P4_6DDR$	$P4_5DDR$	$P4_4DDR$	$P4_3DDR$	$P4_2DDR$	$P4_1DDR$	$P4_0DDR$
Initial value	0	0	0	0	0	0	0	0
Read/Write	W	W	W	W	W	W	W	W

Port 4 data direction 7 to 0
These bits select input or output for port 4 pins

Gambar 2.11 P4DDR

- Mode 1 sampai dengan mode 6

Ketika semua area dijadikan sebagai area akses 8-bit, maka yang dipilih adalah mode bus 8-bit, dan port 4 berfungsi sebagai input/output biasa. Sebuah pin di port 4 menjadi output jika nilai P4DDR diset ke 1, dan menjadi port input jika diset ke 0.

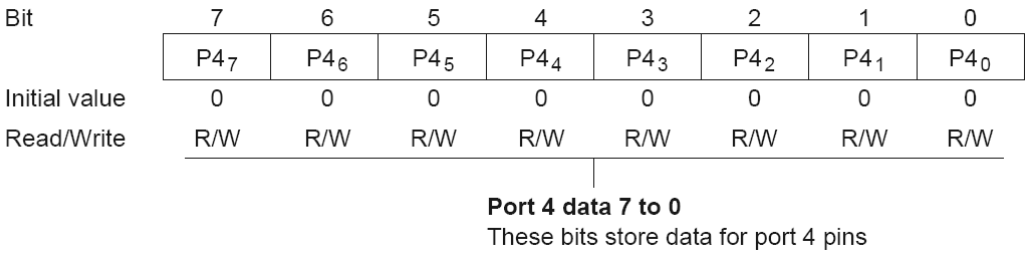
Ketika sedikitnya satu area dijadikan sebagai area akses 16-bit, maka yang dipilih adalah mode bus 16-bit, dan port 4 berfungsi sebagai bus data.

- Mode 7

Port 4 berfungsi sebagai port input/output. Sebuah pin di port 4 menjadi output jika bit P4DDR yang berkaitan diset ke 1, dan menjadi input jika diset ke 0.

b. *Port 4 Data Register* (P4DR)

P4DR adalah register baca/tulis yang menyimpan data output untuk pin $P4_7$ sampai dengan $P4_0$.



Gambar 2.12 P4DR

2.2.2 Pemrograman Port I/O

Contoh pemrograman port I/O pada mikrokontroler H8/3052 adalah sebagai berikut.

```
void initinout(void) {
     P4.DDR = 0x00;      /* Port 4 sebagai input */
}
void main (void) {
     A_lama = P4.DR.BIT.B5;        /* Pembacaan encoder bit A */
     B_lama = P4.DR.BIT.B4;        /* Pembacaan encoder bit B */
     while(1) {
          A_baru = P4.DR.BIT.B5;
          B_baru = P4.DR.BIT.B4;
     }
}
```

Program tersebut merupakan program pembacaan encoder. Sinyal encoder yang terdiri atas 2 sinyal yaitu A dan B digunakan sebagai input. Port 4 diset menjadi input yaitu dengan menset ilai P4.DDR=0. Pembacaan nilai encoder bit A yaitu sesuai dengan nilai pada port 4 pin 5. Sedangkan pembacaan nilai encoder bit B yaitu sesuai dengan nilai pada port 4 pin 4. Sinyal encoder ini selanjutnya akan diolah pada program untuk menentukan besar sudut perputaan motor DC dan juga digunakan untuk menghitung kecepatan motor DC.

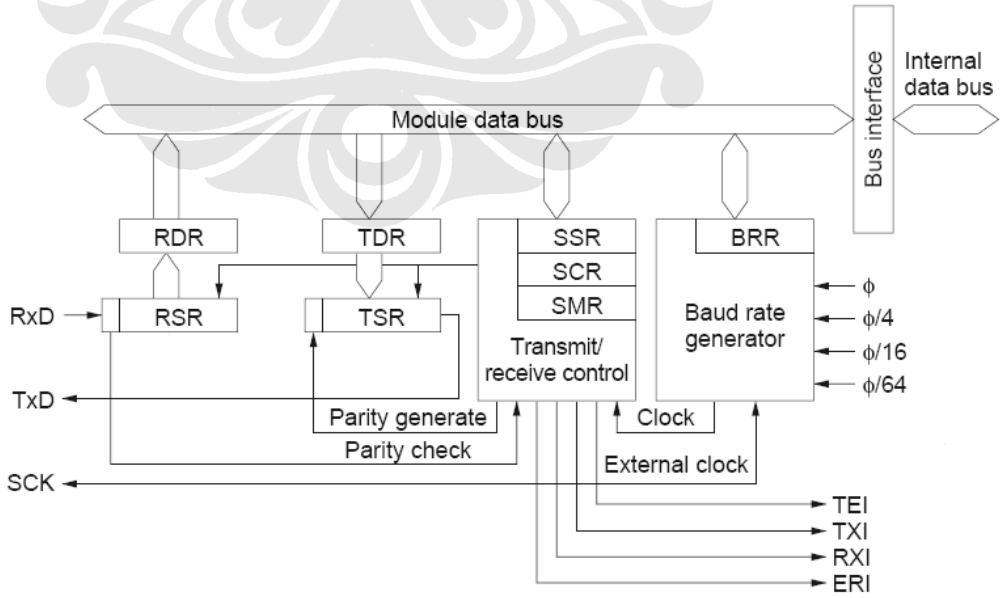
2.3 Serial Communication Interface (SCI)

Mikrokontroler H8/3052 mempunyai dua kanal SCI yang digunakan untuk komunikasi serial. Konfigurasi pin serial dan fungsinya diperlihatkan pada Tabel 2.13 berikut.

Tabel 2.13 Pin SCI dan Fungsinya

Channel	Name	Abbreviation	I/O	Function
0	Serial clock pin	SCK_0	Input/output	SCI_0 clock input/output
	Receive data pin	RxD_0	Input	SCI_0 receive data input
	Transmit data pin	TxD_0	Output	SCI_0 transmit data output
1	Serial clock pin	SCK_1	Input/output	SCI_1 clock input/output
	Receive data pin	RxD_1	Input	SCI_1 receive data input
	Transmit data pin	TxD_1	Output	SCI_1 transmit data output

Tiap kanal SCI mempunyai tiga buah pin yaitu SCK, RxD, dan TxD. Pin SCK digunakan sebagai clock input/output. Pin RxD digunakan sebagai input penerimaan data SCI. Pin TxD digunakan sebagai output pengiriman data SCI. Blok diagram SCI ditunjukkan seperti pada Gambar 2.13.



Gambar 2.13 Blok Diagram SCI

Blok diagram pada Gambar 2.13 menunjukkan hubungan antara register-register pada SCI. TDR dan TSR merupakan register yang digunakan dalam pengiriman data. RSR dan RDR merupakan register yang digunakan dalam penerimaan data serial. SSR, SCR, dan SMR merupakan register yang digunakan untuk pengendalian penerimaan dan pengiriman data serial. Sedangkan BRR digunakan sebagai baud rate generator yang mengatur nilai baud rate pada komunikasi serial. Selanjutnya akan dibahas mengenai deskripsi masing-masing register pada SCI tersebut.

2.3.1 Register-Register SCI

SCI mempunyai 8 jenis register untuk memilih mode sinkron atau asinkron, menentukan format data dan *bit rate*, dan mengendalikan bagian pengiriman dan penerimaan.

a. *Receive Shift Register* (RSR)

RSR adalah register yang menerima data serial. SCI mengisi input data serial pada pin RxD ke RSR sesuai dengan urutan penerimaan, LSB (bit 0) terlebih dahulu sehingga data dikonversi ke data paralel. Ketika satu byte telah diterima, akan secara otomatis ditransfer ke RDR. CPU tidak dapat langsung membaca atau menulis RSR.

b. *Receive Data Register* (RDR)

RDR adalah register yang menyimpan data serial yang diterima. Ketika SCI selesai menerima 1 byte data serial, kemudian data serial yang diterima ditransfer ke RDR untuk penyimpanan. RSR kemudian siap untuk menerima data yang selanjutnya. Mekanisme *double buffering* ini memungkinkan data untuk diterima secara kontinu.

c. *Transmit Shift Register* (TSR)

TSR adalah register yang mengirimkan data serial. SCI mengisi data serial dari TDR ke TSR, kemudian mengirim data secara serial dari pin TxD, LSB (bit 0) terlebih dahulu. Setelah mengirim 1 byte data, SCI secara otomatis mengisi lagi data dari TDR ke TSR dan mulai mengirimkannya. Jika flag TDRE diset 1 pada SSR, SCI tidak mengisi konten TDR ke TSR.

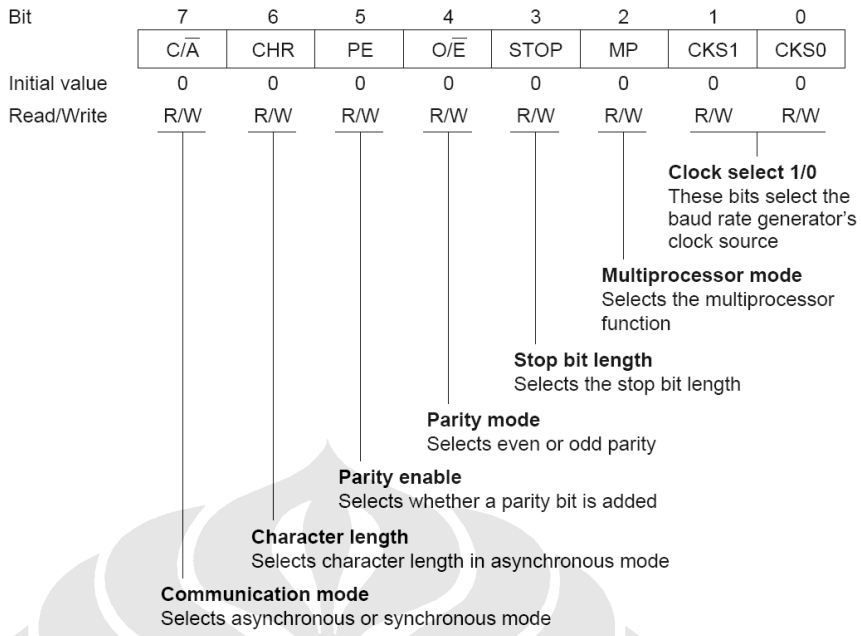
d. *Transmit Data Register* (TDR)

TDR adalah register 8-bit yang menyimpan data untuk pengiriman serial. Ketika SCI mendeteksi bahwa TSR kosong, SCI akan memindahkan data yang tertulis di TDR dari TDR ke TSR dan memulai pengiriman serial. Pengiriman data serial secara kontinu dimungkinkan dengan menuliskan data yang akan dikirim selanjutnya pada TDR selama pengiriman serial dari TSR.

e. *Serial Mode Register* (SMR)

SMR adalah register 8-bit yang menentukan format komunikasi serial dan memilih sumber *clock* untuk *baud rate generator*. Konfigurasi bit pada SMR seperti ditunjukkan pada Gambar 2.14.

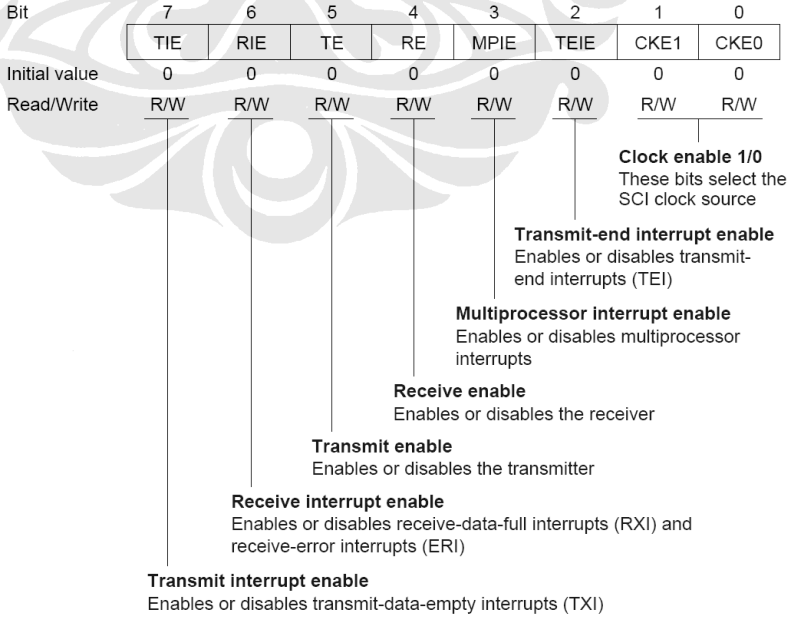
Bit C/A digunakan untuk menentukan mode operasi yaitu sinkron atau asinkron. Bit CHR digunakan untuk memilih panjang karakter pada mode asinkron. Bit PE digunakan untuk memilih penambahan parity atau tidak. Bit O/E menetukan jenis parity yang ditambahkan. Bit STOP digunakan untuk menentukan panjang bit stop. Bit MP digunakan untuk memilih fungsi multiprosesor. Dan 2 bit terakhir yaitu CKS1 dan CKS0 digunakan untuk memilih sumber clock dari baud rate generator.



Gambar 2.14 Serial Mode Register

f. *Serial Control Register* (SCR)

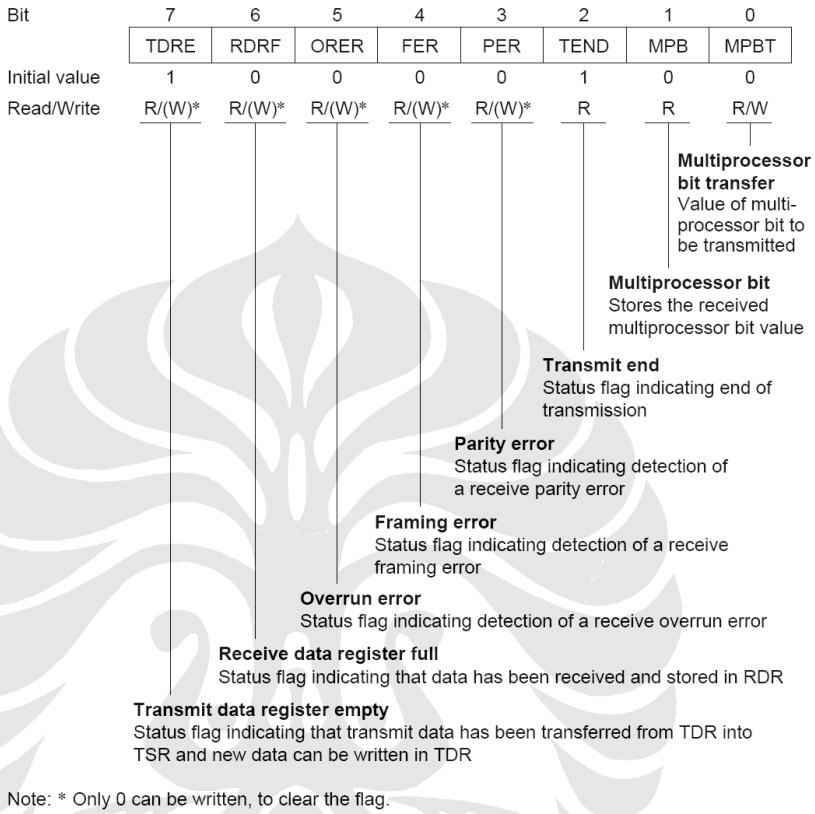
SCR merupakan register 8-bit yang berperan dalam pengaktifan pengirim dan penerima SCI, pewaktuan keluaran pada mode asinkron, interupsi, dan sumber pewaktuan transmisi/penerimaan.



Gambar 2.15 Serial Control Register

g. Serial Status Register (SSR)

SSR merupakan register 8-bit yang mengandung nilai bit multiprosesor, dan status flag-flag yang mengindikasikan status operasi SCI.



Gambar 2.16 Serial Status Register

h. *Bit rate* Register (BRR)

BRR adalah register 8-bit bersama dengan bit CKS1 dan CKS0 pada SMR untuk memilih sumber pewaktuan penghasil baud rate, dan menentukan *bit rate* komunikasi serial. Pada tiap-tiap kanal SCI terdapat kontrol penghasil baud rate yang independen sehingga nilai yang berbeda dapat dituliskan pada 2 kanal yang berbeda.

Pengaturan BRR dihitung sebagai berikut:

$$\mathbf{N} = \frac{\emptyset}{\mathbf{64}\times\mathbf{2^{2n-1}}\times\mathbf{B}} \times \mathbf{10^6} - \mathbf{1} \qquad (2.1)$$

Dengan B = *Bit rate* (bit/sekon)

N = BRR seting untuk *baud rate generator* (0≤N≤255)

$\emptyset$ = Freuensi *clock* sistem (MHz)

n = Sumber *clock baud rate generator* (n = 0, 1, 2, 3)

Nilai n dapat dilihat pada Tabel 2.14 berikut:

Tabel 2.14 Setting Nilai SMR

		SMR Settings	
n	**Clock Source**	**CKS1**	**CKS0**
0	ϕ	0	0
1	ϕ/4	0	1
2	ϕ/16	1	0
3	ϕ/64	1	1

Nilai error *bit rate* pada mode asinkron dapat dihitung berdasarkan persamaan berikut:

$$\boldsymbol{Error}(\%) = \left\{\frac{\emptyset \times \mathbf{10^6}}{(\boldsymbol{N}+\mathbf{1})\times\boldsymbol{B}\times\mathbf{64}\times\mathbf{2^{2n-1}}} - \mathbf{1}\right\} \times \mathbf{100} \qquad (2.2)$$

Pada komunikasi serial yang dilakukan, digunakan sumber *clock* φ (25MHz) dan *bit rate* 9600bps. Sehingga diperoleh nilai n = 0 dan B = 9600, maka nilai setting BRR (N) dapat berdasarkan perhitungan:

$$\mathrm{N} = \frac{25}{64 \times 2^{2(0)-1} \times 9600} \times 10^6 - 1 = 80{,}38 \cong 80 = 0x50$$

2.3.2 Pemrograman SCI

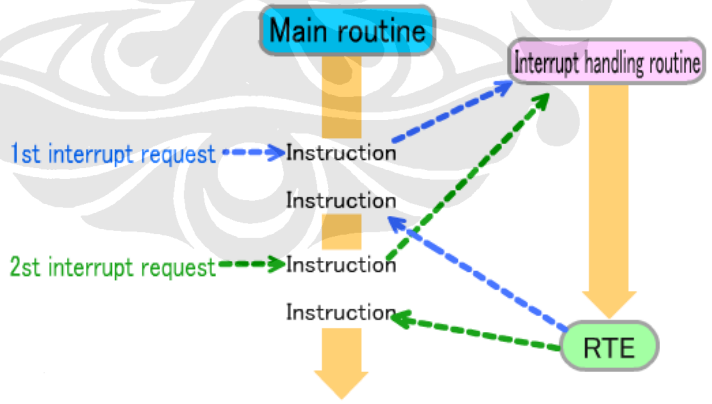
Contoh pemrograman SCI pada mikrokontroler H8/3052 adalah sebagai berikut.

```
void sci_init(void) {
     MSTCR.BIT._SCI0 = 0;      //Enable SCI0 Module
     SCI0.SCR.BYTE = 0x00;     // Disable SCI Interrupts
     SCI0.SMR.BYTE = 0x20;     // Comm. format : Async, 8-bit data, Even Parity
     //added, 1 stop bits
                               // Disable MP, On-cip clock source = 25 Mhz
     SCI0.BRR = 0x13;          // Baud rate initialized to 38400bps
     wait(1);                  // Delay wait until at least 1 bit interval elapsed
     SCI0.SSR.BYTE &= 0x80 ;   /* clear receive flags */
     SCI0.SCR.BYTE = 0x70 ; /* enable receive interrupt,transmit and receive enable */
}
void sci_write( unsigned int data ) {
      while(!(SCI0.SSR.BYTE & 0x80));        // while transmit register empty wait
      SCI0.TDR = data;                       // Transfer data into TDR register.
      SCI0.SSR.BIT.TDRE = 0;
}
unsigned char sci_read( void ){
     unsigned char code;
     while( !(SCI0.SSR.BYTE & 0x78) );
     code = SCI0.RDR;
     SCI0.SSR.BYTE = SCI0.SSR.BYTE & ~0x78;
     return(code);
}
void main (void) {
     set_imask_ccr(1);              /* Disable all of the interrupts             */
      sci_init();                   /* Inisialisasi serial                       */
      set_imask_ccr(0);             /* Enable interrupts                         */
      while(1) {
                sci_write(A_baru);
           sci_write(B_baru);
      }
}
```

2.4 Interrupt Controller

Semua sistem yang mengaplikasikan mikrokomputer pada saat tertentu harus dapat merespon permintaan proses yang mungkin dapat terjadi kapan saja seperti sinyal input keybord pada serial atau input dari sensor. Meskipun memungkinkan untuk satu program secara berurutan mencari dan merespon terhadap semua permintaan proses, tetapi hal itu akan membutuhkan waktu yang sangat lama dan memperlambat respon kecepatannya. Untuk menangani masalah ini, dibutuhkan sebuah mekanisme yang memungkinkan untuk merespon suatu program (*interrupt handling routine*) dan hanya dijalankan ketika permintaan proses tersebut muncul.

Saat terjadi permintaan interrupt, maka program dipegang oleh interrupt handling routine untuk dijalankan program interrupt yang diinginkan. Jika program interrupt telah mencapai akhir interrupt atau mencapai RTE, maka program akan kembali ke main routine. Hal ini seperti ditunjukkan pada Gambar 2.17.

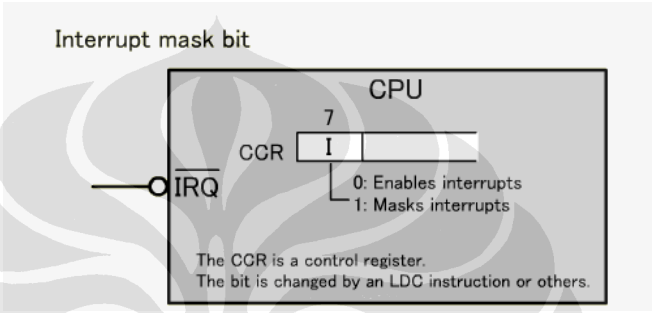


Gambar 2.17 Proses Interrupt

Dua kondisi yang harus dipenuhi untuk memungkinkan interrupt dihasilkan yaitu pertama permintaan interrupt telah muncul dan kedua interrupt tersebut harus di-*enable*.

Interrupt controller pada H8/3052 mempunyai beberapa fitur sebagai berikut

- *Interrupt Priority register* (IPR) untuk mengatur prioritas interrupt
- Tiga level masking dengan bit I dan UI dari pada CCR (Condition Code Register)
- Independent Vector Address, semua interrupt telah memiliki alamat vektor independen sehingga ISR tidak perlu mengidentifikasi lagi sumber interrupt.



Gambar 2.18 Interrupt Mask Bit

Untuk meng-*enable* atau disable interrupt yaitu dikontrol dengan MSB pada CCR (bit I) atau interrupt mask bit. Ketika bit I bernilai 1, interrupt di-masking atau di-disable, dan operasi interrupt tidak dijalankan meskipun jika terjadi permintaan interrupt. Ketika bit I bernilai 0, interrupt di-*enable* dan operasi interrupt dijalankan jika muncul permintaan interrupt. Bit I menutup interrupt, kecuali untuk NMI (Non Maskable Interrupt). NMI tidak dapat di-disable meskipun bit I diset ke 1.

2.4.1 Sumber Interrupt

Sumber interrupt meliputi interrupt eksternal (NMI, IRQ_0 hingga IRQ_5) dan 30 interrupt internal. Skripsi ini hanya menggunakan interrupt internal yaitu timer interrupt dan SCI interrupt.

a. Interrupt Internal

Tiga puluh interrupt eksternal permintaan interrupt-nya berasal dari modul *on-cip* pendukung.

- Tiap modul *on-cip* pendukung mempunyai flag status yang mengindikasikan status interrupt dan mempunyai bit *enable* untuk meng-*enable* atau men-disable interrupt.
- Level prioritas interrupt diatur dari IRA dan IPRB

b. Interrupt Exception Vector Table

Seperti telah dijelaskan sebelumnya bahwa semua interrupt telah memiliki alamat vektor independen sehingga ISR tidak perlu mengidentifikasi lagi sumber interrupt. Tabel vektor interrupt dapat dilihat pada Lampiran. Tabel tersebut menunjukkan sumber interrupt, alamat vektornya, dan default urutan prioritasnya.

2.4.2 Pemrograman Interrupt

Contoh pemrograman interrupt pada mikrokontroler H8/3052 adalah sebagai berikut.

```
//interrupt timer
__interrupt(vect=32) void INT_IMIA2(void) {
    ITU2.TSR.BIT.IMFA &= 0;
    t++;
if (t==50){
     detik++;
     t=0;
     }
}
//interrupt penerimaan data serial
__interrupt(vect=53) void INT_RXI0(void) {
cmd=sci_read();
switch(cmd) {
     case '+' : ITU2.GRB+=100;break;     //menambah duty cycle PWM
     case '-' : ITU2.GRB-=100;break;     //mengurangi duty cycle PWM
     case 'x' : PA.DR.BIT.B5 ^= 1;break; //bolak-balik motor (pin CS)
}
```

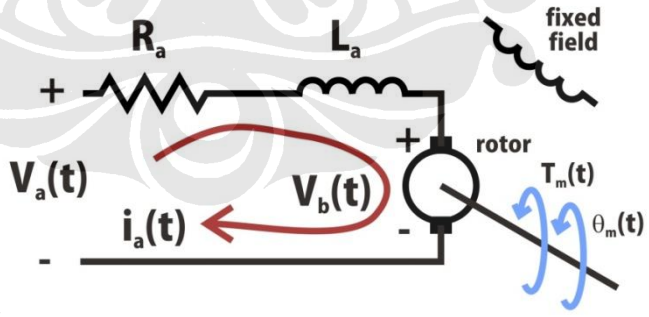
BAB 3

PERANCANGAN KENDALI MOTOR DC

Perancangan kendali motor DC dalam skripsi ini meliputi perancangan motor DC, perancangan blok kendali, perancangan kendali PID, perancangan perangkat lunak, dan perancangan perangkat keras.

3.1 Perancangan Motor DC

Motor DC adalah aktuator listrik yang menggunakan tegangan input sebagai variabel pengendali. Pengendalian motor DC ada dua jenis yaitu pengendalian medan *(field-controlled)* dan pengendalian jangkar *(armature controlled)*. Motor DC yang digunakan pada skripsi ini adalah motor DC dengan pengendalian jangkar. Rangkaian pengganti untuk motor DC dengan pengendalian jangkar dapat dilihat pada Gambar 3.1.



Gambar 3.1 Rangkaian Ganti Motor DC

Motor DC dengan pengendalian jangkar menggunakan arus jangkar i_a sebagai variabel pengendali. Medan stator dapat ditimbulkan dengan kumparan berarus atau dengan magnet permanen.

Ketika arus medan konstan mengalir pada kumparan medan, torsi motor yaitu

$$\boldsymbol{T_m(s) = \left(K_1 K_f I_f\right) I_a(s) = K_m I_a(s)} \quad (3.1)$$

Jika menggunakan magnet permanen, maka

$$\boldsymbol{T_m(s) = K_m I_a(s)} \quad (3.2)$$

Dimana K_m adalah fungsi permeabilitas dari bahan magnet. Arus jangkar berhubungan dengan tegangan input pada rangkaian jangkar sesuai dengan persamaan

$$\boldsymbol{V_a(s) = (R_{a+} L_a s) I_a(s) + V_b(s)} \quad (3.3)$$

Dimana $V_b(s)$ adalah tegangan *back electromotive-force* (emf) yang sebanding dengan kecepatan motor. Karena itu, kita mendapatkan persamaan

$$\boldsymbol{V_b(s) = K_b \omega(s)} \quad (3.4)$$

Dan arus jangkar

$$\boldsymbol{I_a(s) = \frac{V_a(s) - K_b \omega(s)}{(R_{a+} L_a s)}} \quad (3.5)$$

Torsi pada motor adalah sama dengan torsi yang dikirimkan ke beban.

$$\boldsymbol{T_m(s) = T_L(s) + T_d(s)} \quad (3.6)$$

Dimana $T_L(s)$ adalah torsi beban dan $T_d(s)$ adalah torsi gangguan yang biasanya diabaikan. Bagaimanapun juga, torsi gangguan harus diperhatikan pada sistem yang bersubjek pada gaya eksternal seperti hembusan angin yang keras pada antena. Torsi beban untuk benda berputar dituliskan sebagai

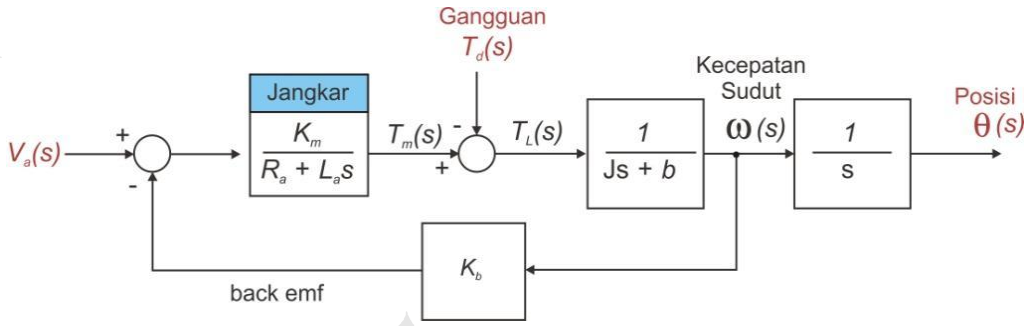
$$\boldsymbol{T_L(s) = Js^2 \theta(s) + bs\theta(s)} \quad (3.7)$$

Persamaan (3.7) dan (3.8) merepresentasikan torsi beban

$$\boldsymbol{T_L(s) = Js^2 \theta(s) + bs\theta(s) = T_m(s) - T_d(s)} \quad (3.8)$$

Secara skematis, hubungan-hubungan antara motor DC dengan

pengendalian jangkar ditunjukkan pada Gambar 3.2.



Gambar 3.2 Blok Diagram Fungsi Alih Motor DC

Pada skripsi ini, digunakan persamaan 3.9 untuk mengetahui fungsi alih dari motor. Nilai ζ dan ω_n didapatkan dari grafik step response yang dilakukan dalam uji coba.

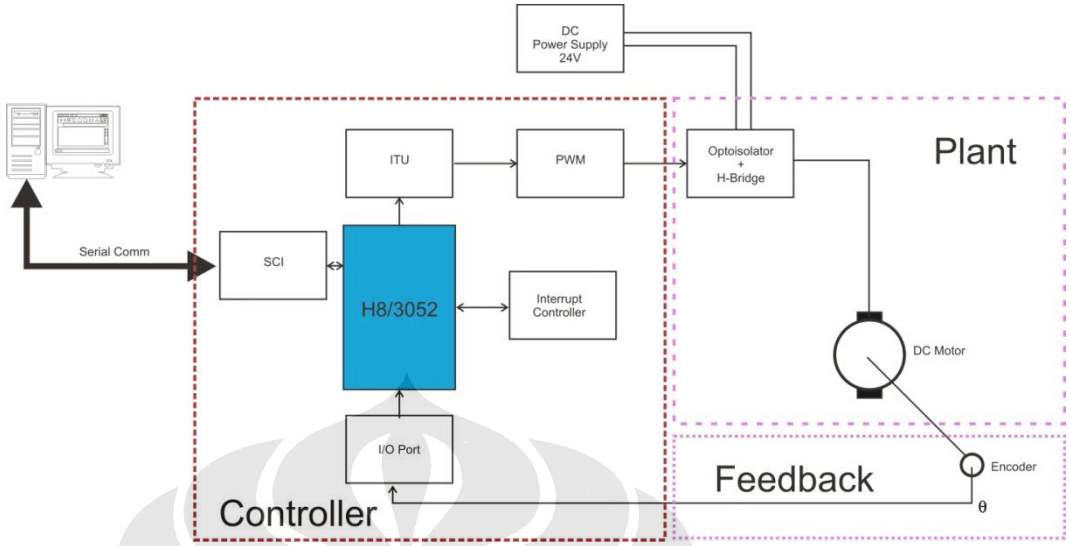
$$\frac{\omega(s)}{V_a(s)} = \frac{1/k_E}{1 + \frac{2s\zeta}{\omega_n} + s^2/\omega_n{}^2} \qquad (3.9)$$

Dari persamaan fungsi alih yang telah didapat, kemudian digunakan untuk mendesain pengendali PID pada motor DC.

3.2 Perancangan Blok Kendali

Perancangan blok kendali motor DC yang terdiri atas plant, feedback, controller, dan pengambil data dapat digambarkan sebagai seperti pada Gambar 3.3 .

PC memberikan nilai setpoint berupa kecepatan ke mikrokontroler H8/3052. Dengan algoritma kendali PID yang telah dibuat, akan menghasilkan input untuk plant berupa nilai duty cycle. Aktuator pada plant yaitu motor DC merespon input tersebut dengan menghasilkan perubahan posisi. Perubahan posisi tersebut diukur menggunakan encoder. Encoder kemudian memberikan feedback ke mikrokontroler.



Gambar 3.3 Skema Perancangan Kendali PID pada Motor DC

Mikrokontroler mengolah nilai perubahan posisi tersebut menjadi kecepatan, kemudian membandingkan nilai kecepatan dari feedback encoder dengan kecepatan dari setpoint. Perbedaan nilai tersebut menghasilkan nilai error yang kemudian diolah pada program kontrol untuk menjadi input pada plant kembali. Nilai-nilai pengukuran yang didapat direkam dan ditampilkan pada komputer untuk selanjutnya dapat dianalisa.

3.3 Perancangan Kendali PID

Salah satu bentuk pengendali yang secara luas digunakan pada pengendalian proses industri adalah pengendali PID. Pengendali ini mempunyai fungsi alih sebagai berikut.

$$G_c(s) = K_P + \frac{K_I}{s} + K_D s \qquad (3.10)$$

Dengan K_P = proportional gain

K_I = integral gain

K_D = differential gain

Persamaan untuk output pada domain waktu adalah

$$u(t) = K_P e(t) + K_I \int e(t)dt + K_D \frac{de(t)}{dt} \quad (3.11)$$

Jika kita setting KD = 0, maka kita mempunyai pengendali proportional plus integral (PI).

$$G_c(s) = K_P + \frac{K_I}{s} \quad (3.12)$$

Ketika KI=0, didapatkan

$$G_c(s) = K_P + K_D s \quad (3.13)$$

yang disebut sebagai pengendali proportional plus derivative (PD).

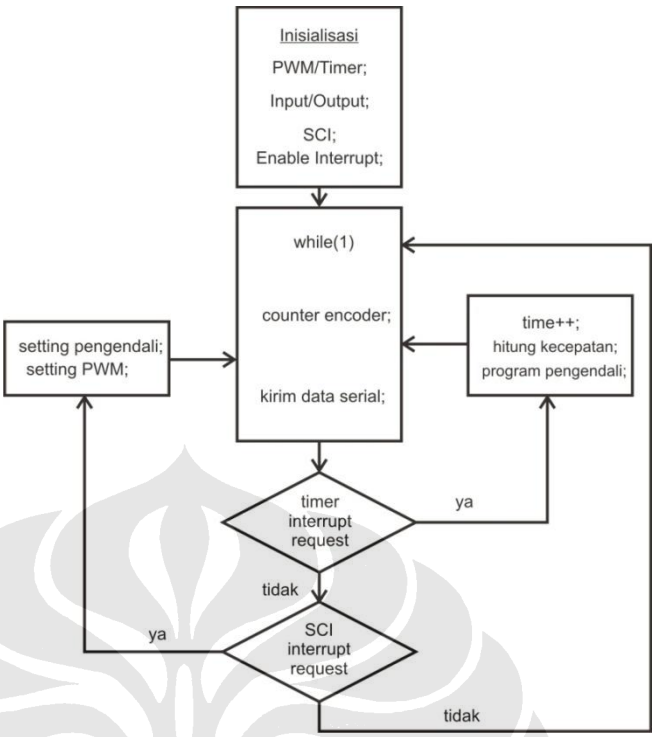
Program pengendali tersebut kemudian diterapkan pada pemrograman C pada mikrokontroler.

3.4 Perancangan Perangkat Lunak

Perancangan perangkat lunak terdiri atas dua bagian yaitu perancangan perangkat lunak pada mikrokontroler dan perancangan perangkat lunak pada PC.

Pada perancangan perangkat lunak mikrokontroler, penulisan program dengan menggunakan bahasa C dengan menggunakan software HEW (*High-performanced Embedded Workshop*) dari Renesas. Kemudian program C yang telah dibuat kemudian di-build hingga menjadi file dalam .mot. File .mot inilah yang di-*burn* ke mikrokontroler dengan menggunakan program H8Writer.

Untuk program yang dijalankan pada mikrokontroler, seperti ditunjukkan pada diagram alir pada Gambar 3.4.



Gambar 3.4 Diagram Alir Program pada Mikrokontroler

Pada awal program dimulai dengan inisialisasi program yang terdiri atas inisialisasi PWM/Timer, inisialisasi input/output, inisialisasi SCI, dan inisialisasi interrupt. Pada program utama dilakukan program penghitungan posisi motor dengan menggunakan sinyal dari encoder serta program pengiriman data serial yang terdiri atas data posisi, waktu, setpoin, dan kecepatan motor.

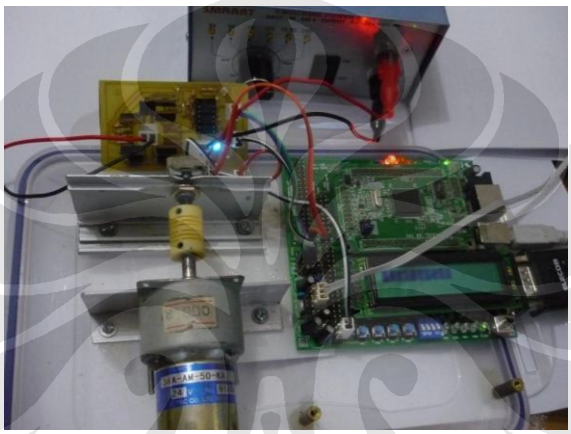
Pada saat terjadi permintaan interrupt timer, mikrokontroler melaksanakan program penanganan interrupt yang terdiri atas penambahan nilai waktu, perhitungan kecepatan, dan aplikasi program pengendalian. Pada saat terjadi permintaan interrupt penerimaan SCI, mikrokontroler menjalankan program penanganan interrupt yang terdiri atas setting nilai duty cycle PWM, serta setting nilai Kp, Ki, dan Kd untuk pengendalian. Jika program penanganan interrupt selesai, maka program kembali ke program utama.

Perancangan perangkat lunak pada PC terdiri atas program pembacaan data

serial menggunakan gcc cygwin, pembuatan grafik data dengan gnuplot, dan tampilan secara realtime menggunakan aplikasi graph232.

3.5 Perancangan Perangkat Keras

Perancangan blok diagram pengendalian kemudian diaplikasikan ke perancangan perangkat keras. Masing-masing komponen dirangkai hingga menadi rangkaian perangkat keras seperti ditunjukkan pada Gambar 3.5.



Gambar 3.5 Rangkaian Perangkat Keras

Rangkaian perangkat keras yang digunakan yaitu terdiri atas:

a. Motor DC Tsukasa Electric TG-38A-AM-50-KA

Gambar 3.6 Motor DC Tsukasa Electric

Motor DC yang digunakan yaitu motor DC produksi Tsukasa Electric dengan nomor seri TG-38A-AM-50-KA. Motor ini mempunyai tegangan rating 24V.

b. *Evaluation Kit* AKI-H8/3052-LAN

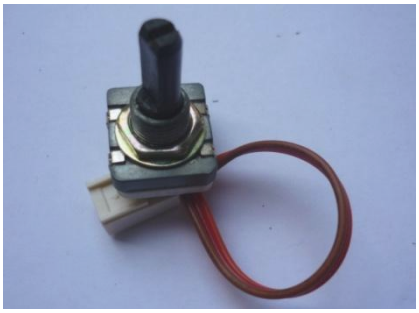
Evaluation kit AKI-H8/3052-LAN ini digunakan sebagai perangkat pengendali. Evaluation kit multifungsi ini memiliki beberapa dukungan perangkat keras tambahan yaitu LCD, 4 buah LED, kanal untuk motor servo , koneksi RS232, koneksi LAN, dan koneksi power USB, dan koneksi power 5V.

Gambar 3.7 AKI-H8/3052-LAN

LCD digunakan untuk debugging dan menampilkan data. Akan tetapi pada skripsi ini LCD tidak digunakan karena operasi LCD cukup menyita waktu yang lama. Kanal servo digunakan untuk output sinyal PWM serta outpun pin CS. Komunikasi serial dapat memilih menggunakan SCI0 atau SCI1. Untuk catu daya digunakan power USB dengan tegangan kurang lebih 5V.

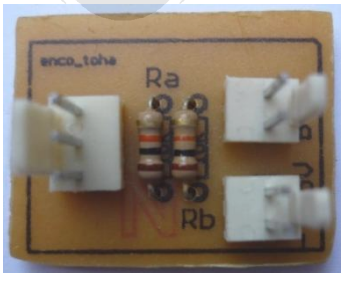
c. Rotary Encoder EC16B

Encoder yang digunakan yaitu EC16B yang dibuat oleh ALPS *Electricity Company* di Jepang. Bentuk hardware dari encoder EC16B dapat dilihat pada Gambar 3.8.

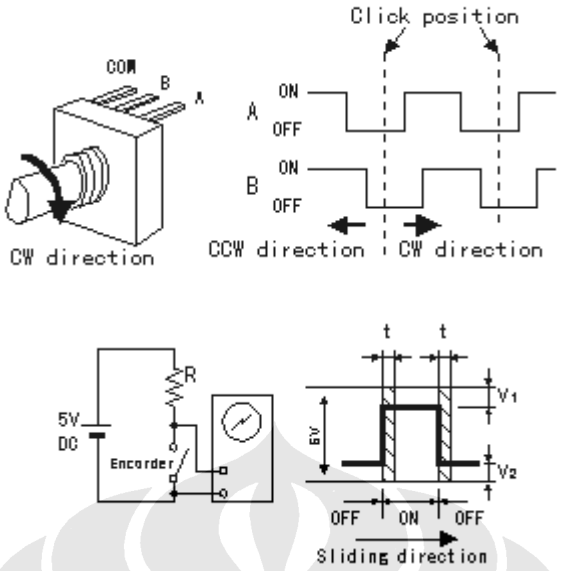


Gambar 3.8 Encoder EC16B

Encoder ini mempuyai 3 terminal, satu digunakan sebagai *common terminal* (COM), dan dua lainnya digunakan untuk output A dan B. Output dari terminal yaitu kondisi dimana terminal tersebut terhubung dengan *common terminal* atau tidak. Encoder EC16B ini tergolong sebagai encoder mekanik yaitu menggunakan sentuhan dalam menentukan output encoder. Dua terminal output mempunyai perbedaan waktu saat menyentuh *common terminal* ketika sumbu encoder berputar. Untuk interface encoder dengan mikrokontroler, digunakan rangkaian encoder seperti pada Gambar 3.9. Rangkaian ini terdiri atas dua buah resistor 10K. Rangkaian encoder ini menggunakan catu daya dari mikrokontroler yaitu sebesar 5V. Terdapat dua pin sebagai output dari encoder yaitu pin A dan pin B. Pin A dan pin B ini harus dijaga agar tidak menyentuh ground agar tidak terjadi kesalahan dalam perhitungan.



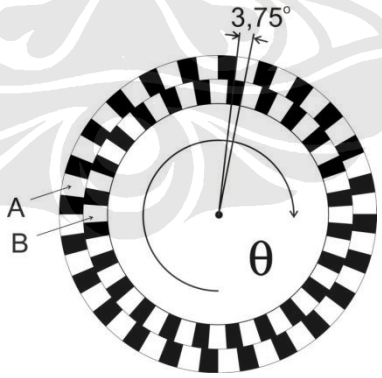
Gambar 3.9 Rangkaian Encoder



Gambar 3.10 Prinsip Kerja Encoder EC16B

Sumber: http://www.interq.or.jp/japan/se-inoue/e_ckt10.htm

Gambar 3.10 menunjukkan sinyal encoder yang dihasilkan pada pin A dan pin B saat diputar searah atau berlawanan arah jarum jam. Encoder ini dapat digunakan untuk menghitung putaran baik searah jarum jam maupun berlawanan arah jarum jam.



Gambar 3.11 Incremental Rotary Encoder

Encoder EC16B merupakan incremental rotary encoder dengan resolusi sebesar 3,75°. Resolusi encoder tersebut merupakan satuan terkecil perubahan posisi yang dapat diukur oleh encoder. Besar resolusi ini didapatkan dari besar putaran encoder dibagi dengan jumlah perubahan output biner dalam satu kali

putaran yaitu ditunjukkan pada persamaan 3.14.

$$Resolusi = \frac{Sudut\ dalam\ 1\ putaran}{Jumlah\ pulsa \times Jumlah\ perubahan\ biner\ per\ pulsa} \quad (3.14)$$

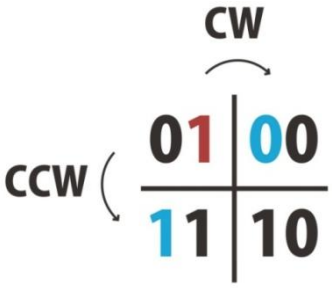
$$Resolusi = \frac{360^\circ}{24 \times 4} = 3,75^\circ \quad (3.15)$$

Sebuah incremental encoder menggunakan 2-bit output biner untuk menentukan posisi putaran motor. Encoder EC16B ini menggunakan sensor mekanik untuk mendapatkan output biner tersebut. Terdapat dua track yaitu track A dan track B sebagai 2-bit output encoder. Persentuhan antara dua pin yaitu pin A dan B encoder dengan pin common menghasilkan sinyal output encoder.

Perbedaan fasa antara track A dan B digunakan untuk menghitung putaran encoder. Kita dapat menggunakan Gray code untuk menentukan posisi dan arah dari perputaran motor. Gray code atau juga dikenal *reflected binary code* adalah sistem penomoran biner dimana dua nilai yang bersebelahan tepat hanya memiliki satu digit beda. Dinamakan Gray code karena ditemukan oleh Frank Gray. Pada awalnya Frank Gray memperkenalkan *reflected binary code* dalam paten aplikasinya tahun 1947. Dia memberikan nama berawal dari fakta bahwa kode ini "mungkin dibentuk dari kode biner yang konvensional dengan urutan proses yang terbalik".

Kode ini dikatakan reflected karena bisa dibentuk dengan cara dicerminkan. Ambil Gray code 0,1 kemudian dicerminkan dan dituliskan pada urutan setelahnya hingga menjadi 0, 1, 1, 0. Tambahkan 0 pada setengah pertama dan 1 pada setengah digit kedua setelah digit-digit tersebut menjadi 00, 10, 11, 01 (Gray code 2 digit). Untuk mendapatkan Gray code dengan jumlah digit ke-(n+1) yaitu dengan mencerminkan Gray code digit ke-n kemudian menambahkan digit 0 pada setengah pertama dan digit 1 pada setengah kedua.

Prinsip kerja Gray code dalam menentukan posisi dan arah perputaran motor dapat dijelaskan seperti pada **Error! Reference source not found.**.



Gambar 3.12 Prinsip Gray Code 2bit

Gambar itu dapat kita sederhanakan dengan algoritma sebagai berikut

if(bit_kanan_lama XOR bit_kiri_baru = = 1) CW++;

if(bit_kanan_lama XOR bit_kiri_baru = = 0) CCW++;

dengan bit kanan = bit pada track B, dan bit kiri = bit pada track A.

Pada gambar di atas, posisi 01 kita anggap sebagai posisi lama. Jika berputar searah jarum jam (CW), maka posisi baru adalah 00. Dengan algoritma tersebut, maka:

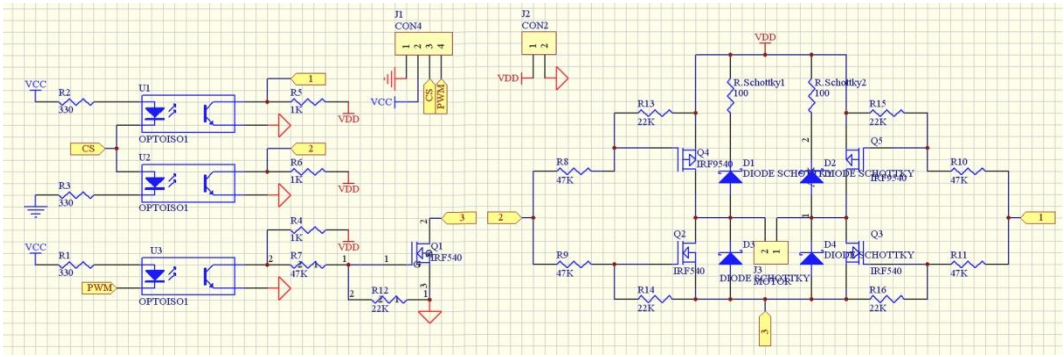
bit_kanan_lama XOR bit_kiri_baru = 1 XOR 0 = 1, maka CW++;

Sedangkan jika berputar berlawanan arah jarum jam, maka posisi baru adalah 11.

bit_kanan_lama XOR bit_kiri_baru = 1 XOR 1 = 0, maka CCW++;

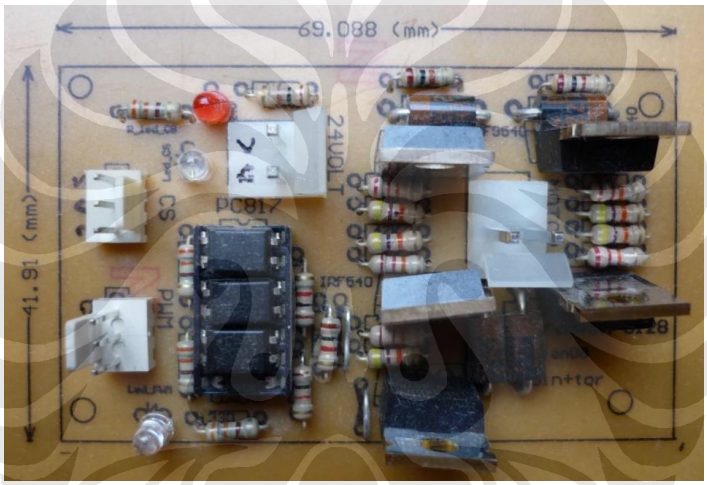
d. Rangkaian *driver* motor

Rangkaian *driver* motor di sini merupakan gabungan antara rangkaian optoisolator dan H-bridge. Skematik dan hardware rangkaian driver motor dapat dilihat pada Gambar 3.13 dan Gambar 3.14.



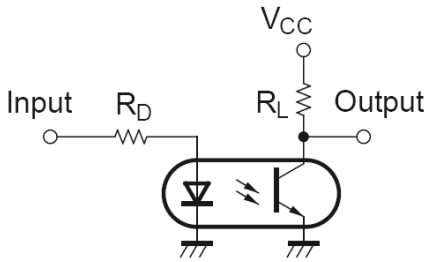
Gambar 3.13 Rangkaian skematik *Driver* Motor: Optoisolator dan H-Bridge

Sumber: Dokumen Tim Robot UI 2008



Gambar 3.14 Optoisolator dan Driver H-Bridge Motor DC

Rangkaian optoisolator digunakan untuk memisahkan sumber tegangan mikrokontroler dengan sumber tegangan motor, dan juga menghasilkan output PWM dan $\overline{\text{PWM}}$. Sedangkan input dari optoisolator ada dua yaitu PWM dan CS. CS inilah yang menentukan output optoisolator apakah PWM atau $\overline{\text{PWM}}$. Komponen utama optoisolator ini adalah PC817 SHARP.



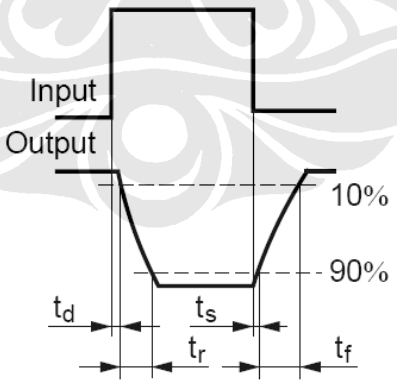
Gambar 3.15 Prinsip Kerja PC817

Sumber: http://pdf1.alldatasheet.net/datasheet-pdf/view/43371/SHARP/PC817.html

Prinsip kerja optoisolator pada Gambar 3.15 adalah sebagai berikut. Pada saat input bernilai HIGH, maka LED pada optoisolator akan menyala dan transistor pada optoisolator ON sehingga output dihubungkan dengan GROUND dan output tidak menyala.

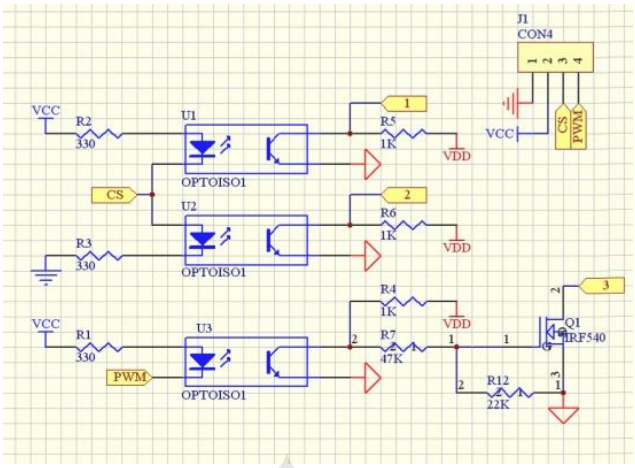
Sebaliknya saat input bernilai LOW, maka LED pada optoisolator tidak menyala dan transistor OFF. Akibatnya output mendekati nilai Vcc.

Dengan rangkaian seperti pada Gambar 3.15 maka hubungan antara input dan output dapat digambarkan seperti pada Gambar 3.16.



Gambar 3.16 Hubungan input dan output pada PC817

Sumber: http://pdf1.alldatasheet.net/datasheet-pdf/view/43371/SHARP/PC817.html

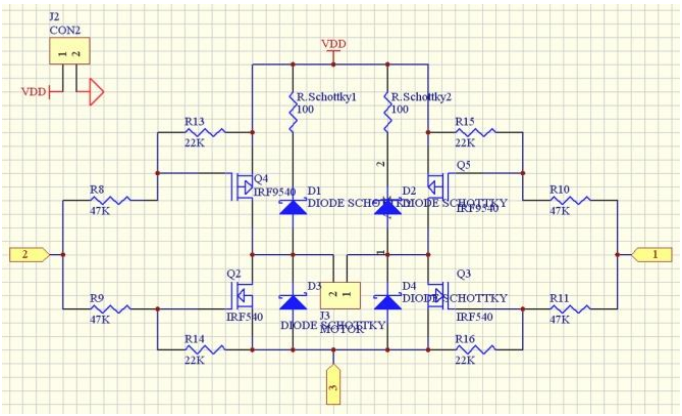


Gambar 3.17 Rangkaian Optoisolator

Jika diaplikasikan pada rangkaian optoisolator seperti pada Gambar 3.17, maka optoisolator dapat digunakan untuk menentukan arah motor. Nilai VCC kita anggap sebagai 1 dan nilai Ground kita anggap 0. Saat pin CS bernilai 1, makaterjadi perbedaan tegangan antara CS dengan Ground sehingga LED pada U3 akan menyala. Hal ini mengakibatkan transistor pada U3 ON dan pin 2 terhubung dengan Ground sehinga bernilai 0. Sebaliknya karena antara VCC dan CS tidak terdapat perbedaan tegangan maka LED pada U1 tidak menyala, transistor U1 OFF. Sehingga pin 1 tidak terhubung dengan Ground, dan tegangan pada pin 1 besarnya kurang lebih sama dengan VDD dan nilainya kita anggap 1. Sehingga saat pin CS bernilai 1, maka pin 1 bernilai 1 dan pin 2 benilai 0.

Hal ini berlaku sebaliknya, yaitu saat pin CS bernilai 0, maka pin 1 bernilai 0 dan pin 2 bernilai 1. Pengaturan inilah yang menyebabkan motor dapat bergerak searah jarum jam atau berlawanan arah dengan jarum jam.

Rangkaian H-Bridge sendiri digunakan untuk mengendalikan gerak motor maju atau mundur. Komponen utama yang digunakan yaitu transistor IRF9540N dan IRF540N. Skematik Protel rangkaian H-Bridge dapat dilihat pada Gambar 3.18.



Gambar 3.18 Rangkaian H-Bridge Motor

Pin 1 dan 2 pada rangkaian merupakan pin yang menentukan arah putaran motor. Nilai pin ini ditentukan dari output rangkaian optoisolator. Pin 3 merupakan pin PWM yang menentukan besarnya kecepatan motor. Pin V_{DD} merupakan sumber tegangan untuk motor yaitu sebesar 24V. Dioda schottky digunakan untuk menangkap back emf motor yang dihasilkan oleh kumparan motor saat power on dan off. Tegangan flyback ini bisa berapa kali lebih tinggi daripada tegangan supply. Jika tidak digunakan diode, maka transistor dapat terbakar. Resistor $R_{Schottky}$ sebesar 100Ω digunakan sebagai pembagi tegangan. Resistor ini hanya dipasang pada bagian antara dioda dengan sumber tegangan sedangkan di sisi diode dengan PWM tidak dipasang karena sudah terpasang seri.

e. Perangkat keras lainnya

Perangkat keras lainnya terdiri atas PC, peralatan serial, dan digital tachometer. PC digunakan untuk menampilkan dan merekam data yang diperoleh melalui komunikasi serial dari mikrokontroler. Tachometer digunakan untuk mengukur kecepatan motor DC untuk dibandingkan dengan pengukuran menggunakan encoder. Tachometer yang digunakan pada skripsi ini yaitu Dual Digital Tachometer DT-2268 Lutron. Sedangkan peralatan komunikasi serial terdiri atas konektor USB ke serial dan kabel serial to serial.

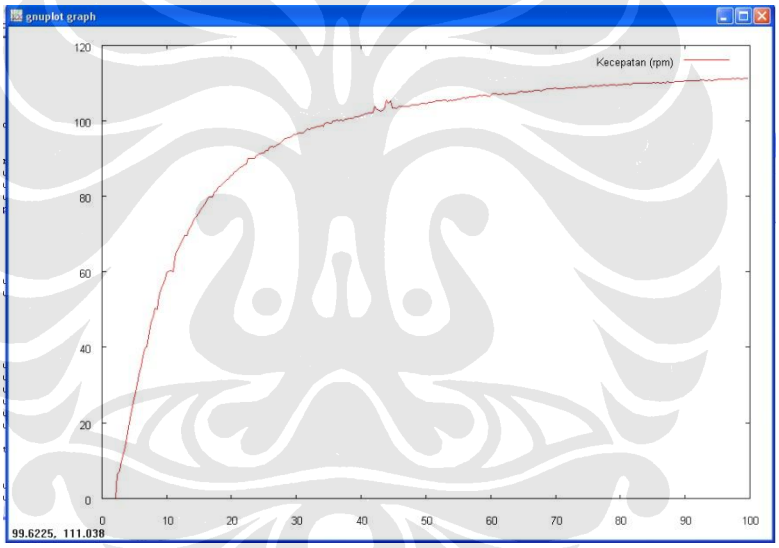
BAB 4

PENGUJIAN DAN ANALISA

Ada beberapa percobaan yang dilakukan.

4.1 Pengujian Fungsi Alih Tegangan (Duty Cycle) terhadap Motor

Pengujian ini dilakukan dengan memberikan input PWM pada motor kemudian diukur kecepatan putar motor DC dalam rpm dengan menggunakan tachometer digital. Didapatkan grafik dari hasil percobaan sebagai berikut.



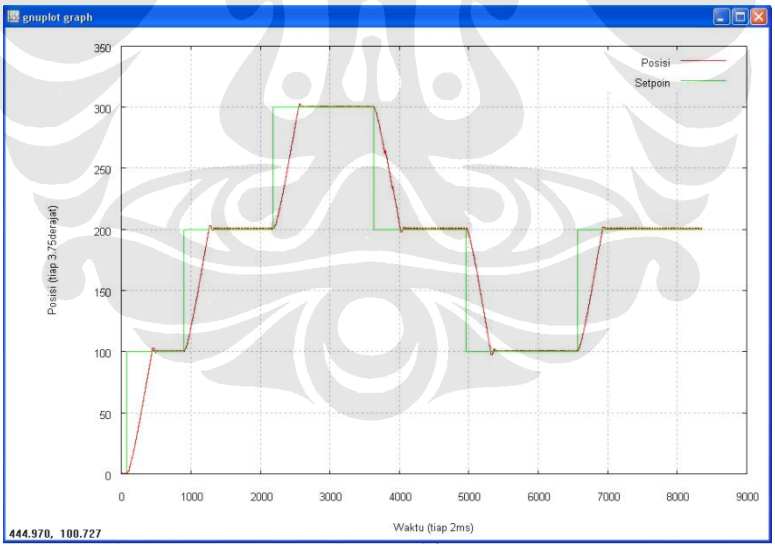
Gambar 4.1 Kurva Kecepatan terhadap Duty Cycle PWM

Kurva fungsi alih tersebut menunjukkan bahwa hubungan antara tegangan (duty cycle PWM) dan kecepatan putar motor tidak linier. Untuk itulah diperlukan desain pengendali yang dapat digunakan untuk semua range kecepatan dalam range tersebut.

4.2 Pengujian Pengendalian Posisi

Pada pengendalian posisi, pertama dilakukan setting pengendali dengan memberikan nilai Kp, Ki, dan Kd pada program mikrokontroler dari komputer dengan serial. Dengan nilai setpoint yang berubah-ubah, didapatkan grafik posisi motor seperti pada Gambar 4.2.

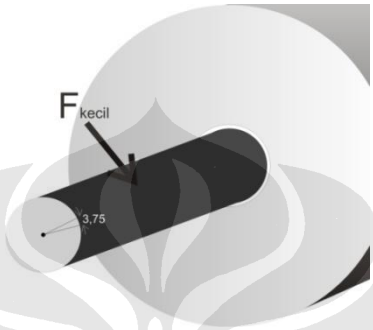
Ketika motor diberikan nilai setpoint posisi tertentu, motor akan menaikkan kecepatannya untuk menuju setpoin yang diinginkan. Ketika sudah mencapai posisi setpoint, motor akan berusaha mempertahankan untuk tetap berada pada posisi tersebut. Jika posisi motor melewati batas setpoint, maka motor akan sedikit berputar berlawanan dengan arah dengan sebelumnya untuk kembali mencapai posisi tersebut.



Gambar 4.2 Grafik Pengendalian Posisi Motor

Dapat dilihat bahwa nilai error yang didapatkan mendekati nol. Error hanya terjadi berkisar 1 tik encoder (3,75˚) yaitu sebesar resolusi encoder. Hal ini dikarenakan sifat motor yang tidak dapat benar-benar diam pada titik tertentu tetapi memang selalu bergeser sebesar 1 tik ketika diberi sedikit gaya.

Pergerakan pada motor ini disebut backlash. Backlash ini terjadi karena as pada motor DC tidak bisa terpasang tepat pada motor DC. Ketika terjadi error tersebut, pengendali langsung memberikan tanggapan dengan perubahan nilai PWM sehingga motor kembali pada posisi setpoin yang diinginkan.



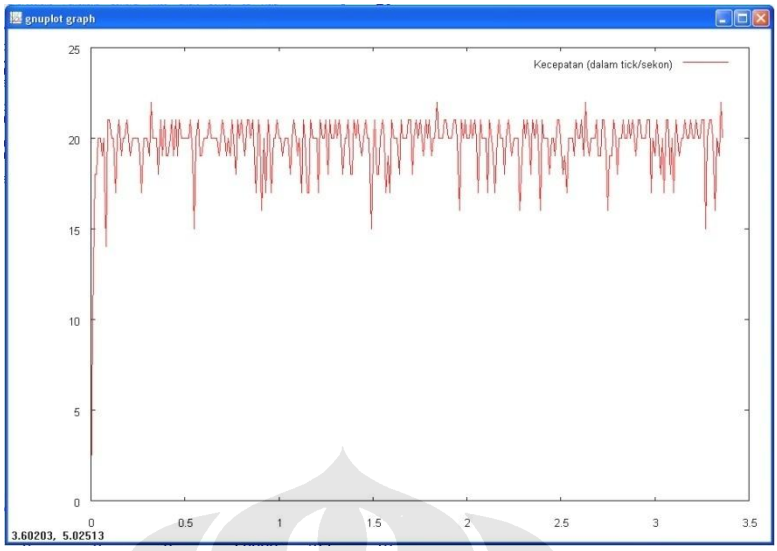
Gambar 4.3 Backslash Sebesar 3,75˚

4.3 Pengujian Kecepatan Motor

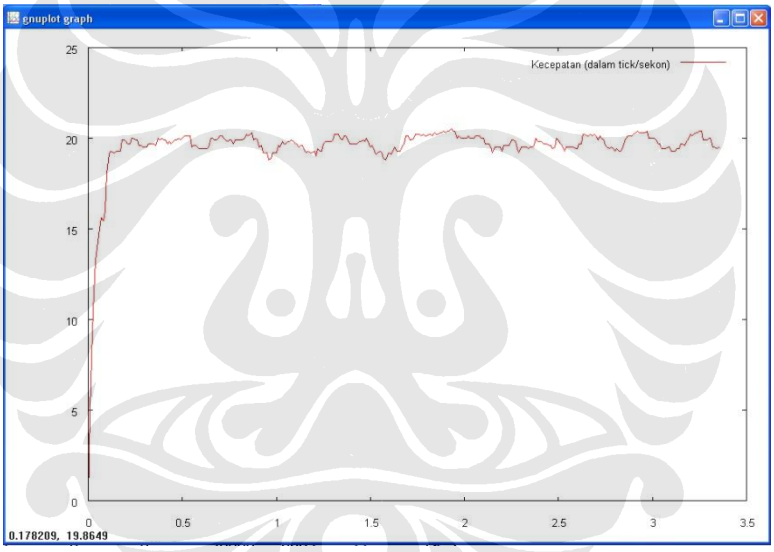
Pengujian pengendalian kecepatan motor dilakukan dengan memberikan sampling time sebesar 100ms. Tiap 100ms dijalankan program perhitungan kecepatan, dan program pengendalian. Pertama dilakukan pengujian step response pada motor tanpa pengendali. Grafik pada Gambar 4.4 menunjukkan adanya ripple pada kurva kecepatan motor. Hal ini dikarenakan metode yang digunakan dalam perhitungan kecepatan motor DC yaitu dengan perhitungan metode T

$$\omega = \frac{\Delta\theta}{\Delta t}$$

Perhitungan ini diperbaiki yaitu dengan digunakannya filter pada perhitungan kecepatan. Filter yang digunakan yaitu dengan menggunakan rata-rata tiap 10 sample. Setelah dilakukan proses filtering didapatkan step response tanpa pengendali seperti pada Gambar 4.5.



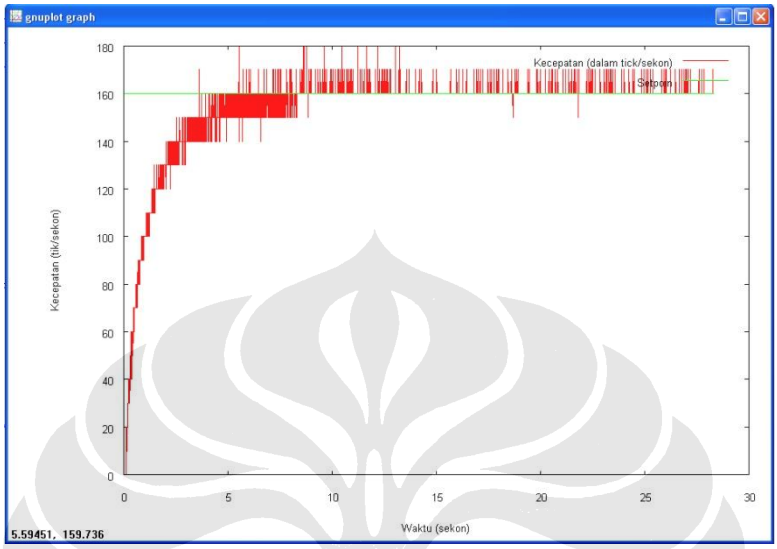
Gambar 4.4 Pengujian Step Response Tanpa Filter Kecepatan



Gambar 4.5 Pengujian Step Response dengan Filter Kecepatan

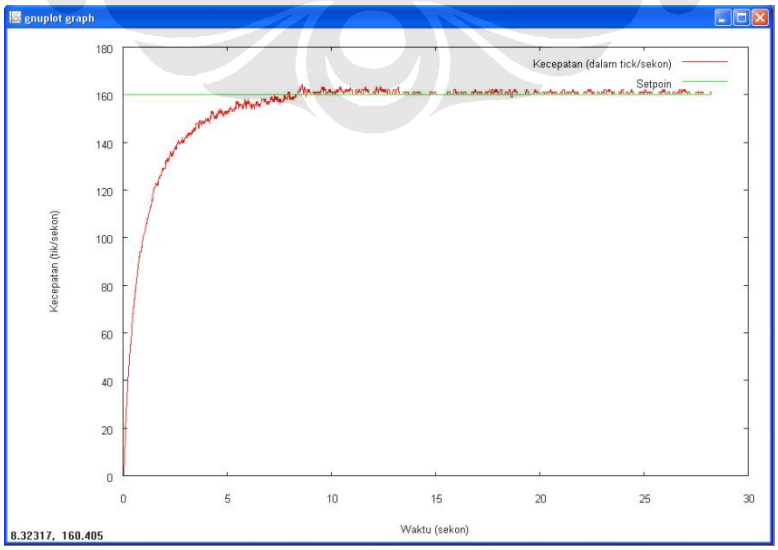
Pada pengujian tanpa pengendali, setelah dilakukan filtering kecepaan, didapatkan grafik kecepatan seperti Gambar 4.5. Nilai kecepatan ini tergantung pada duty cycle PWM. Jika kita ingin memberikan nilai setpoint kecepatan tertentu, maka kita harus mengubah sendiri nilai duty cycle secara manual agar didapatkan kecepatan yang diinginkan. Agar kecepatan dapat mengikuti setpoint yang diinginkan tanpa harus mengubah secara manual, maka kita memberikan pengendali pada motor tersebut.

Pada percobaan pertama dilakukan pengendalian PID tanpa beban. Pada percobaan ini diberikan nilai setpoint yaitu 160 tik/s. Hasilnya dapat dilihat seperti pada Gambar 4.6.



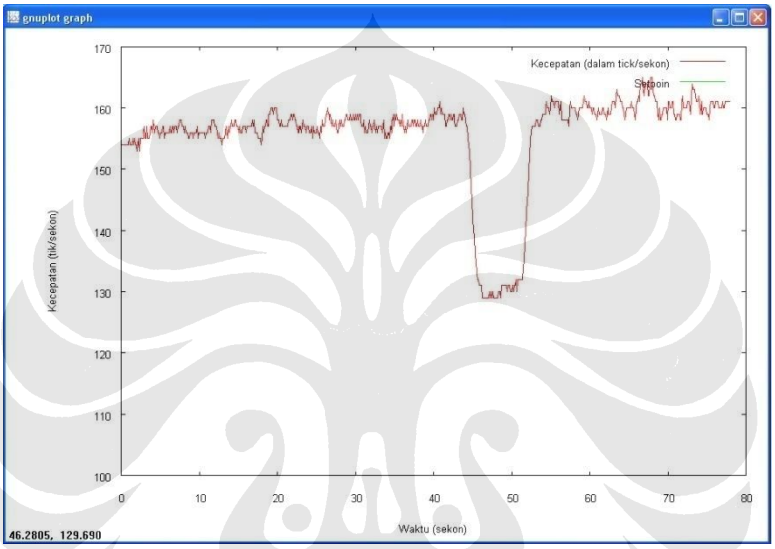
Gambar 4.6 Pengendalian Kecepatan Motor dengan PID Tanpa Filter Kecepatan

Seperti pada pengujian step response tanpa pengendali, diperlukan proses filtering agar didapatkan nilai kecepatan yang baik. Setelah dilakukan proses filter, maka didapatkan grafik kecepatan motor dengan pengendalian PID seperti pada Gambar 4.7.

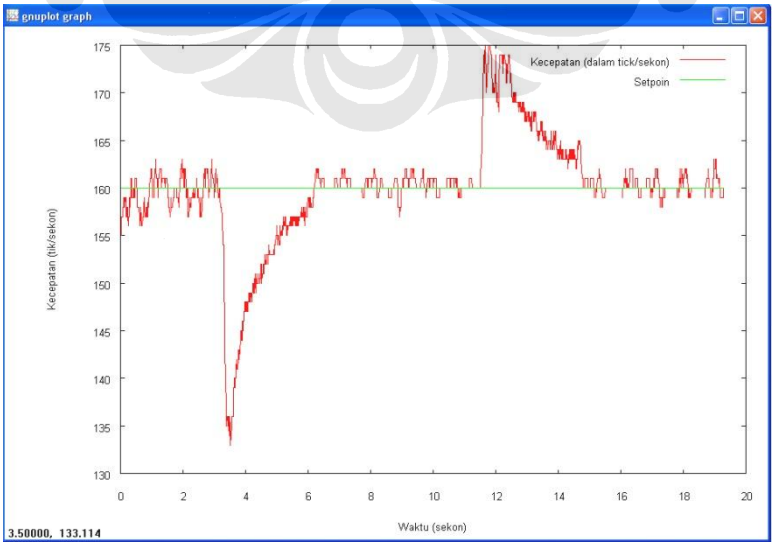


Gambar 4.7 Pengendalian Kecepatan Motor PID dengan Filter

Pada grafik yang ditunjukkan pada Gambar 4.7 menunjukkan bahwa dengan pengendalian PID, didapatkan nilai persen overshoot kecil mendekati nol, error steady state mendekati nol, dan setting time yang cepat yaitu 9 sekon. Hal ini menunjukkan bahwa pengendali PID dapat digunakan sebagai pengendali kecepatan motor untuk mendapatkan sistem yang memiliki respon yang cepat, dan nilai error steady state mendekati nol.



Gambar 4.8 Pemberian Beban pada Motor Tanpa Pengendali



Gambar 4.9 Pemberian Beban pada Motor dengan Pengendali PID

Pada percobaan selanjutnya, dilakukan percobaan pemberian beban pada motor. Pemberian beban pada motor tanpa pengendali menyebabkan motor tidak mencapai setpoint kecepatan yang diinginkan. Hal ini terlihat pada Gambar 4.8 yang menunjukkan saat diberi beban, kecepatan motor yaitu sebesar 130tik/s di bawah kecepatan setpoint 160tik/s. Kecepatan kembali menuju setpoint hanya setelah beban dilepaskan.

Berbeda dengan kecepatan motor tanpa pengendali, pada motor DC dengan pengendali PID, saat motor diberi beban, maka kecepatan motor turun, akan tetapi kempali menyesuaikan dengan kecepatan setpoint. Hal ini dapat dilihat pada Gambar 4.9. Begitu juga saat dilepaskan, kecepatan motor naik akan tetapi kemudian kembali ke kecepatan setpoint yang diinginkan. Dengan demikian, pengendali PID dapat mengendalikan kecepatan motor sesuai dengan nilai setpoint yang diinginkan meskipun diberi beban.

BAB 5

KESIMPULAN

Dari keseluruhan skripsi ini dapat disimpulkan beberapa hal yaitu:

1. Pada skripsi ini, mikrokontroler digunakan sebagai unit pengendali. PC digunakan sebagai pemonitor untuk member setpoint dan untuk menampilkan sinyal masukan dan keluaran motor yang dikendalikan dan sekaligus dapat merekam sinyal-sinyal tersebut untuk dianalisa.
2. Penggunaan *interrupt service routine* pada program mikrokontroller sangat penting untuk mempermudah pemrograman proses kontrol.
3. Pengendalian posisi motor dengan PID menghasilkan posisi motor yang sesuai dengan yang diinginkan.
4. Pada pengendalian kecepatan motor, sinyal keluaran kecepatan motor harus difilter untuk menghasilkan informasi kecepatan motor yang lebih baik.
5. Pada percobaan pengujian kecepatan motor dengan diberi beban, pengendali PID mampu memperbaiki nilai deviasi kecepatan akibat beban tambahan tersebut sehingga kecepatan motor dapat kembali ke kecepatan setpoint yang diinginkan.

DAFTAR REFERENSI

Kenji Tamaki, Kiyoshi Ohnishi, Kouhei Ohnishi, Kunio Miyachi. (1986). *Microprocessor-Based Robust Control of a DC Servo Motor.* IEEE Control Systems Magazine

___. (2001). *Hitachi Single Chip Microcomputer H8/3052F-ZTAT Hardware Manual.*

Estiko Rijanto. 2000. *Robust Control: Theory for Application.* ITB Press Bandung

Richard C. Dorf, Robert H. Bishop. 2005. *Modern Control Systems, Tenth Edition.* Pearson Educational International. New Jersey.

_____. *H8 Family Tutorial Course,* Renesas. 24 November 2008. http://resource.renesas.com/lib/eng/e_learning/h8_300henglish/index.html

John G. Bollinger, Neil A. Duffie. 1998. *Computer Control of Machines and Processes.* Addison Wesley Publishing Company.

LAMPIRAN

1. Tabel Vektor Interrupt

Sumber Interrupt	Keterangan	Asal	Nomor Vektor	IPR
NMI	Non Maskable Interrupt	Pin Eksternal	7	
IRQ0			12	IPRA7
IRQ1			13	IPRA6
IRQ2			14	IPRA5
IRQ3			15	
IRQ4			16	IPRA4
IRQ5			17	
WOVI	Interval Timer	Watchdog Timer	20	IPRA3
CMI	Compare Match	Refresh Controller	21	
IMIA0	Compare Match/ Input Capture A0	ITU Channel 0	24	IPRA2
IMIB0	Compare Match/ Input Capture B0		25	
OVI0	Overflow 0		26	
IMIA1	Compare Match/ Input Capture A1	ITU Channel 1	28	IPRA1
IMIB1	Compare Match/ Input Capture B1		29	
OVI1	Overflow 1		30	
IMIA2	Compare Match/ Input Capture A2	ITU Channel 2	32	IPRA0
IMIB2	Compare Match/ Input Capture B2		33	
OVI2	Overflow 2		34	
IMIA3	Compare Match/ Input Capture A3	ITU Channel 3	36	IPRB7
IMIB3	Compare Match/ Input Capture A3		37	
OVI3	Overflow 3		38	
IMIA4	Compare Match/ Input Capture A4	ITU Channel 4	40	IPRB6
IMIB4	Compare Match/ Input Capture A4		41	
OVI4	Overflow 4		42	
DEND0A		DMAC	44	IPRB5
DEND0B			45	
DEND1A			46	
DEND1B			47	
ERI0	Receive Error 0	SCI Channel 0	52	IPRB3
RXI0	Receive Data Full 0		53	
TXI0	Transmit Data Empty 0		54	
TEI0	Transmit End 0		55	
ERI1	Receive Error 1	SCI Channel 1	56	IPRB2
RXI1	Receive Data Full 1		57	
TXI1	Transmit Data Empty 1		58	
TEI1	Transmit End 1		59	
ADI	A/D End	A/D	60	IPRB1