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

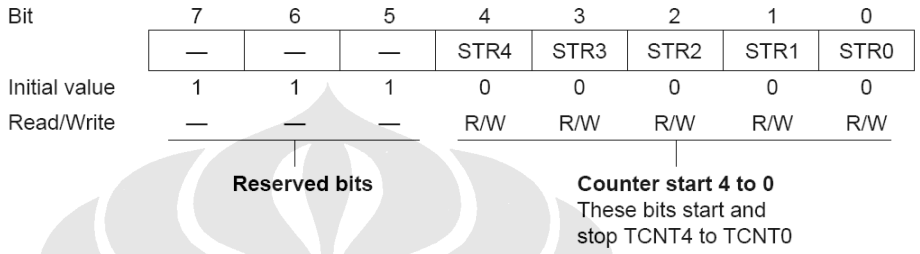
Selanjutnya akan dijelaskan mengenai register-register pada ITU, cara pengoperasian PWM dengan ITU, dan pemrograman ITU pada mikrokontroler.

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.



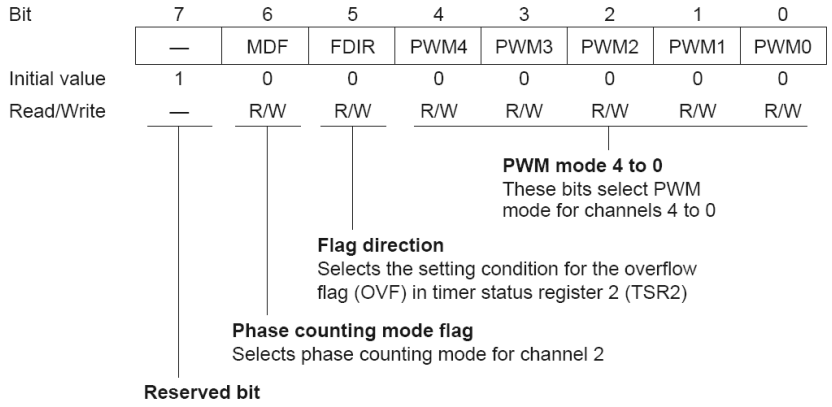
Gambar 2.1 Register TSTR

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.



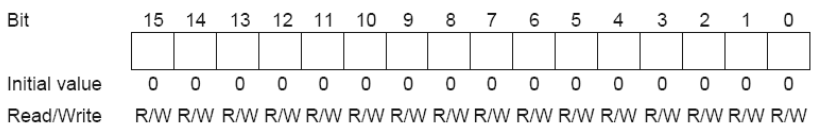
Gambar 2.2 Register TMDR

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	

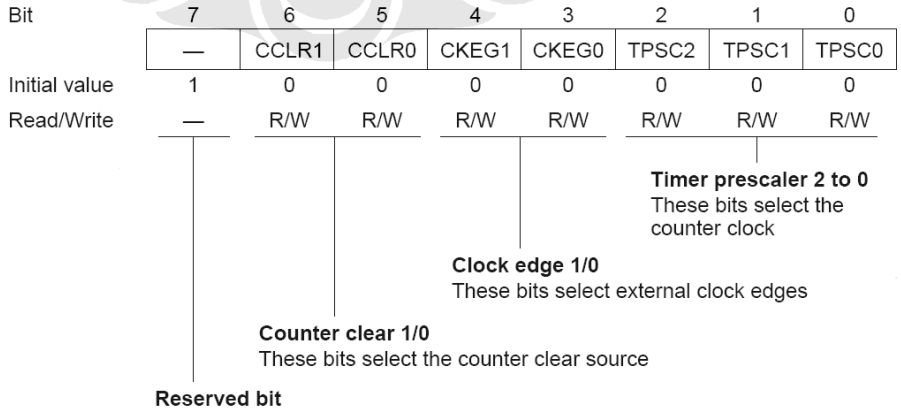
Pada skripsi ini digunakan GRA dan GRB compare match pada ITU2. General register berfungsi sebagai output compare register, nilai general register tersebut secara konstan dibandingkan dengan nilai TCNT. Ketika dua nilai tersebut sesuai (compare match), flag IMFA atau IMFB menjadi bernilai 1 pada TSR.

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.

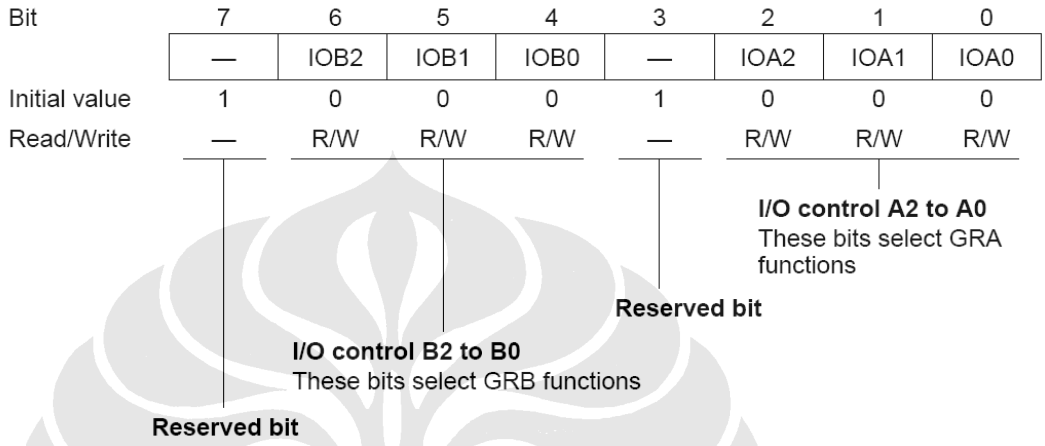
f. *Timer I/O Control Register* (TIOR)

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)

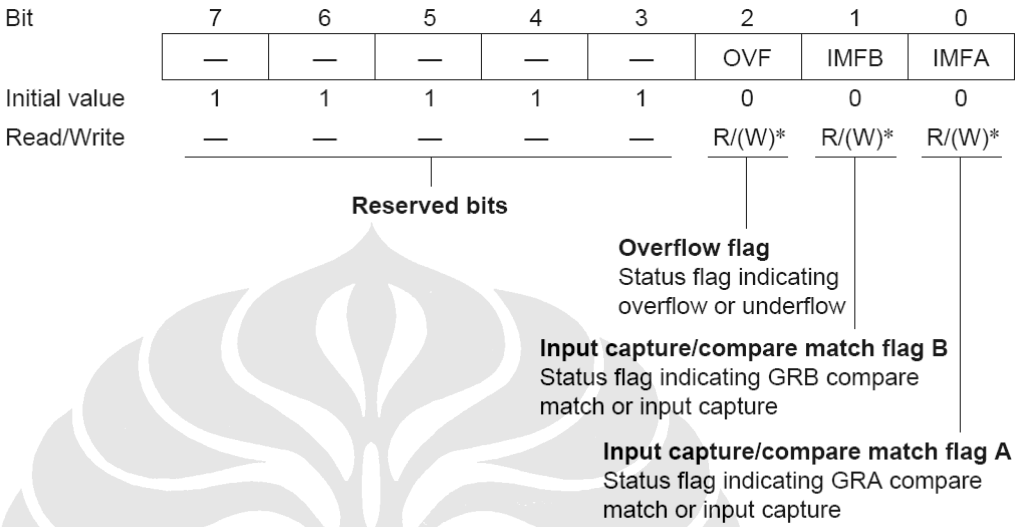
TSR adalah register 8-bit yang menunjukkan status input capture, compare match, maupun status overflow. ITU mempunyai 5 buah TSR, masing-masing satu buah di tiap kanalnya, seperti ditunjukkan pada Tabel 2.10.

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.

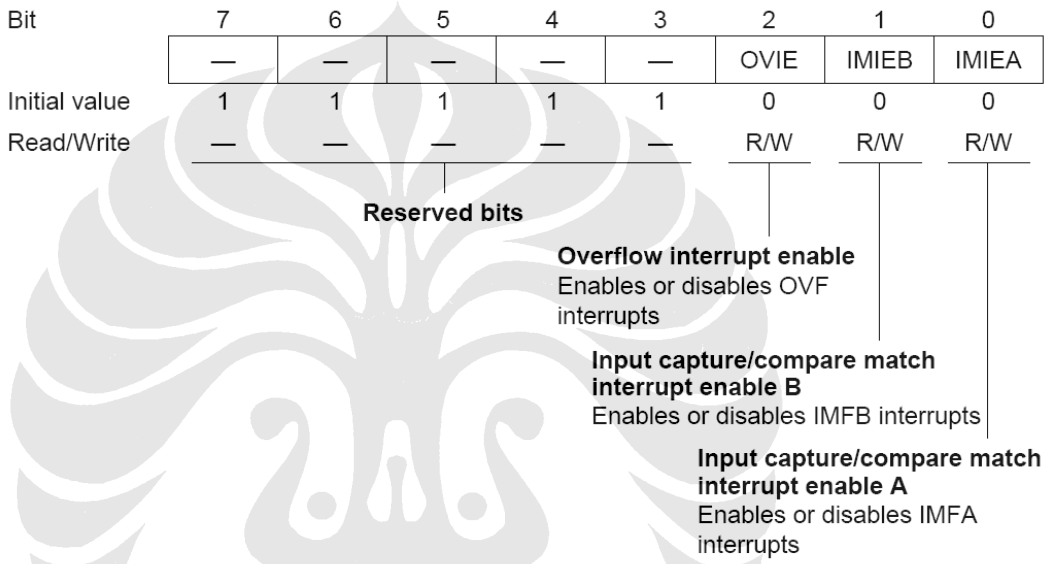
h. *Timer Interrupt Enable Register* (TIER)

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

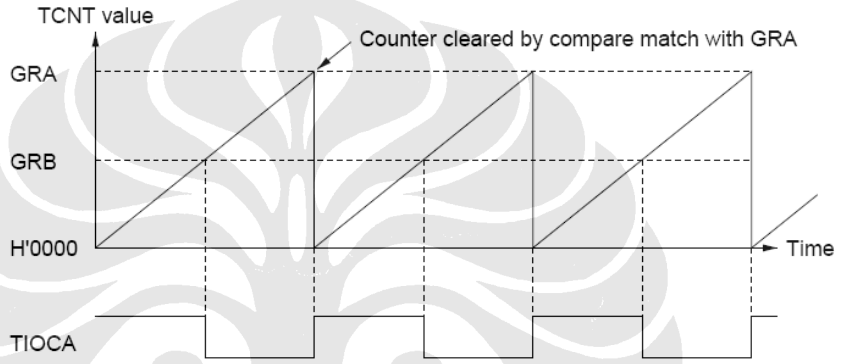
Seperti ditunjukkan pada Gambar 2.7, bit OVIE merupakan bit yang men-enable atau disable interrupt OVF. Bit IMIEB digunakan untuk meng-enable atau disable interrupt IMFB. Sedangkan bit IMIEA digunakan untuk meng-enable atau disable interrupt IMFA.

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

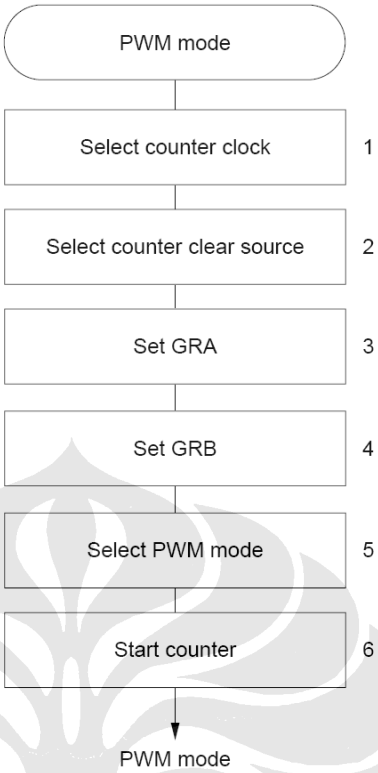
Pada operasi mode PWM, GRA dan GRB bekerja secara berpasangan dan gelombang PWM adalah output dari pin TIOCA. Output PWM menjadi 1 saat compare match dengan GRA dan menjadi 0 saat compare match dengan GRB. *Duty cycle* suatu gelombang PWM merupakan perbandingan antara nilai GRA dan GRB. Hal ini seperti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

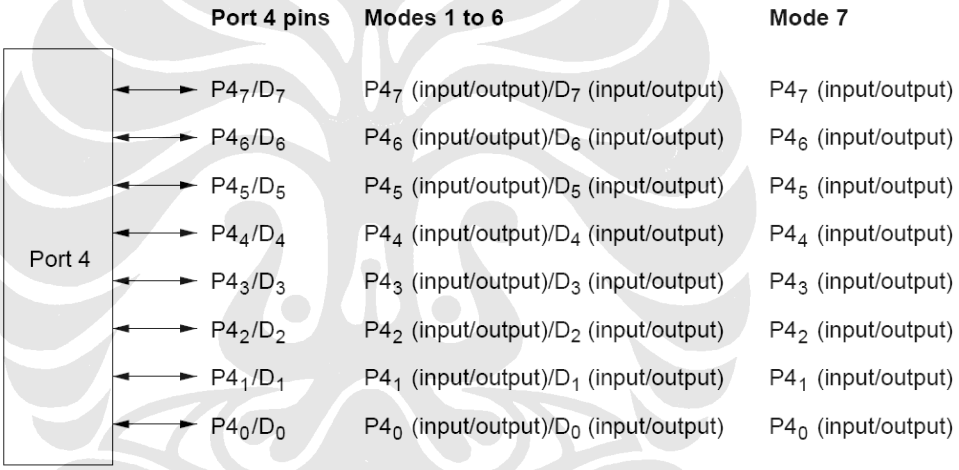
```
void initPWMsingle(void) {
      MSTCR.BIT._ITU = 0;           /* ITU operates normally */
      ITU.TOCR.BIT.XTGD = 1;    /* Disable external trigerring */
      ITU.TSTR.BIT.STR2 = 0;    /* Stop Counting TCNT in channel 2 */
      ITU2.TCR.BIT.TPSC = 3;    /*Select internal clock = 25Mhz*/
      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.

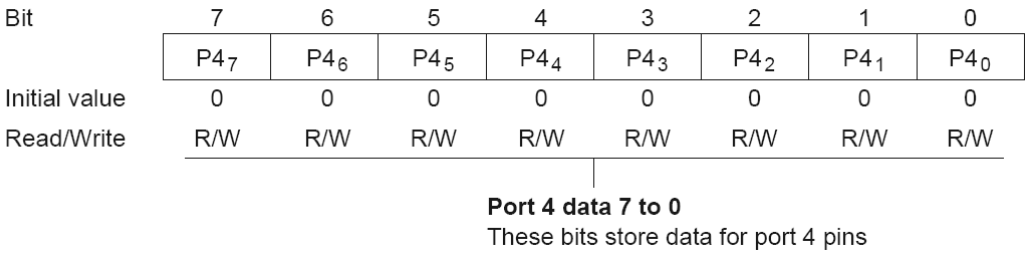
Ketika sedikitnya satu area dijadikan sebagai area akses 16-bit, maka yang dipilih adalah mode bus 16-bit, dan port 4 berfungsi sebagai bus data.

- 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.

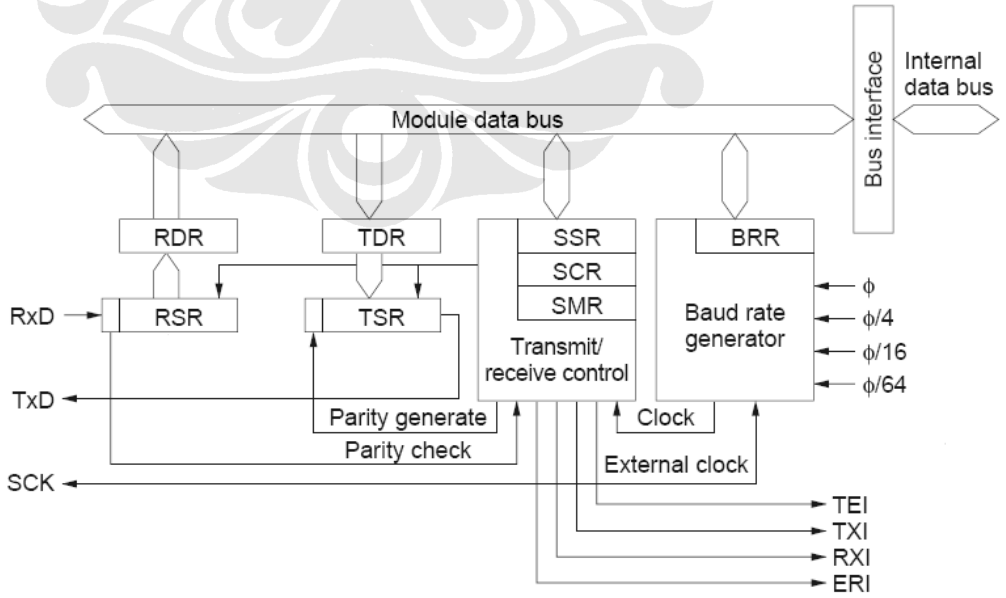
2.3 Serial Communication Interface (SCI)

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.

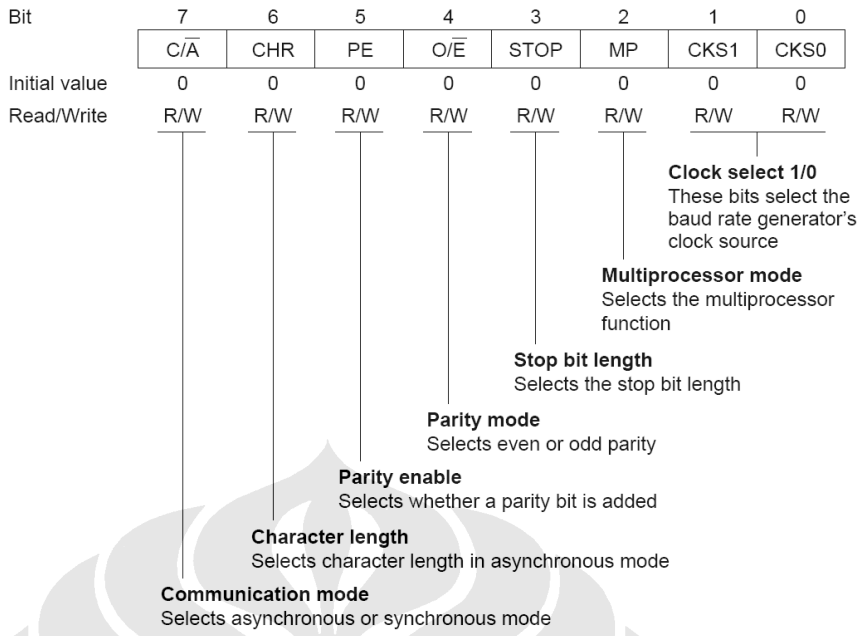
d. *Transmit Data Register* (TDR)

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.

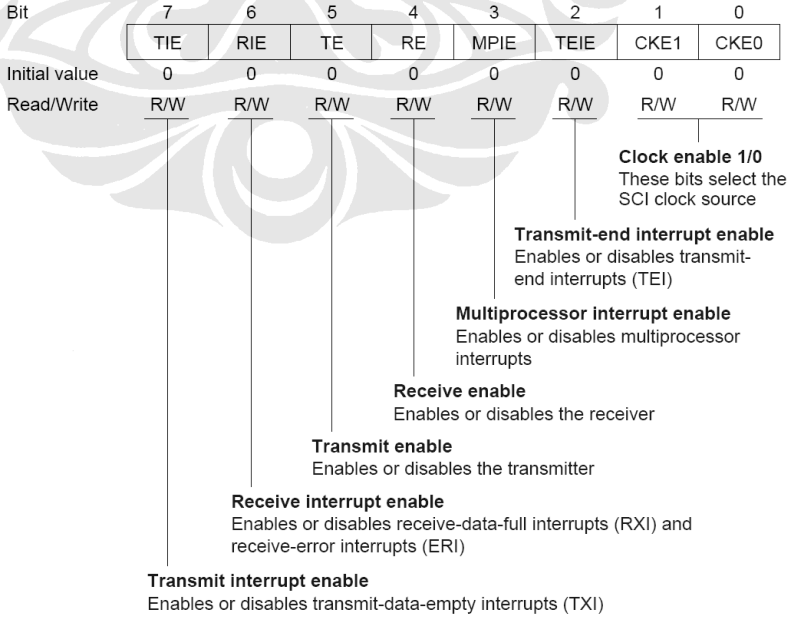
Bit C/A digunakan untuk menentukan mode operasi yaitu sinkron atau asinkron. Bit CHR digunakan untuk memilih panjang karakter pada mode asinkron. Bit PE digunakan untuk memilih penambahan parity atau tidak. Bit O/E menetukan jenis parity yang ditambahkan. Bit STOP digunakan untuk menentukan panjang bit 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)

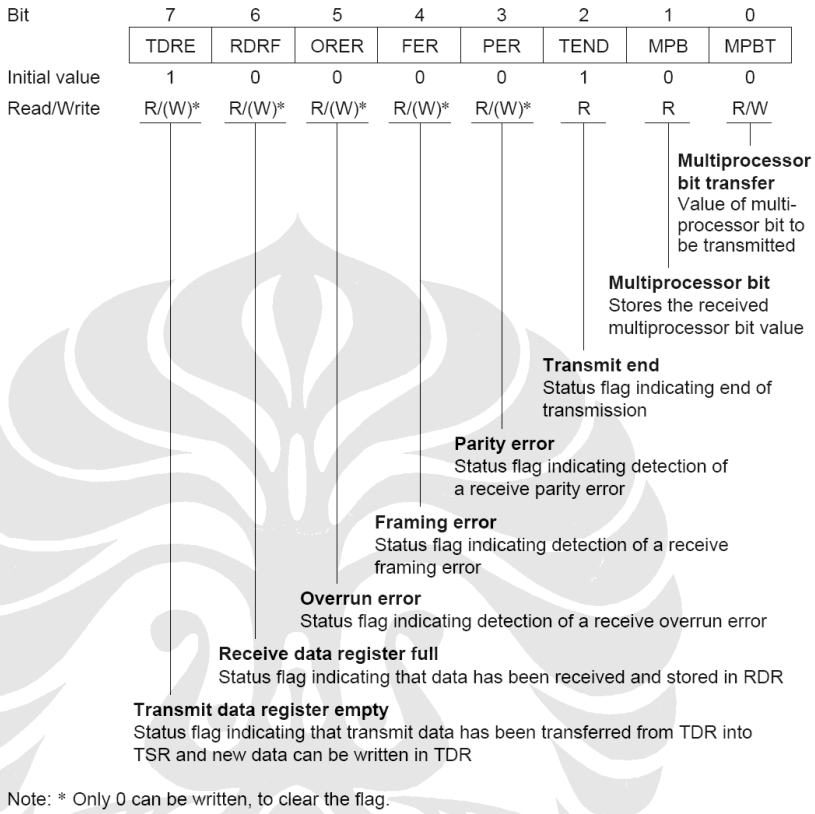
SCR merupakan register 8-bit yang berperan dalam pengaktifan pengirim dan penerima SCI, pewaktuan keluaran pada mode asinkron, interupsi, dan sumber pewaktuan transmisi/penerimaan.



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

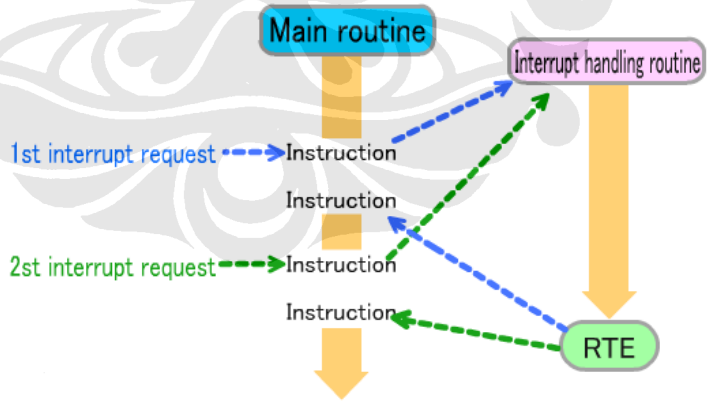
Contoh pemrograman SCI pada mikrokontroler H8/3052 adalah sebagai berikut.

```
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.

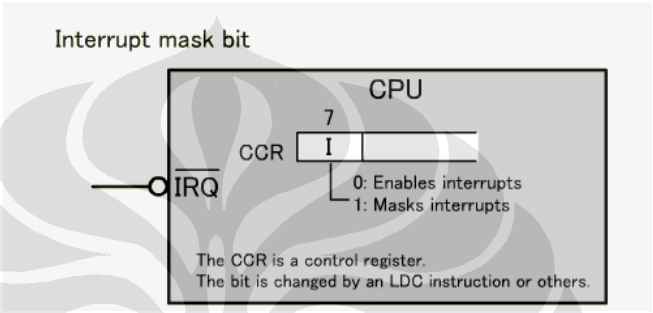


Gambar 2.17 Proses Interrupt

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)
}
```