BAB II

LANDASAN TEORI

2.1. Pemancar FM

Pemancar adalah suatu alat yang berfungsi membawa dan memproses sinyal informasi untuk ditransmisikan. Metode yang dipakai adalah menumpangkan sinyal informasi pada sinyal yang mempunyai frekuensi yang lebih tinggi, memperkuat sinyal kemudian memancarkannya ke udara. Sinyal frekuensi tinggi tersebut berfungsi membawa sinyal informasi dari pesawat pengirim ke pesawat penerima, oleh karena itu disebut sinyal pembawa (*carrier*). Proses penumpangan sinyal informasi pada sinyal pembawa terjadi pada piranti yang disebut dengan *modulator*. Prosesnya disebut proses modulasi dan hasil keluarannya disebut sinyal termodulasi.

Pemancar VHF (*Very High Frequency*) adalah pemancar yang dikategorikan kedalam kelompok frekuensi antara 144 – 148 MHz dengan menggunakan metode modulasi frekuensi (HT).

2.1.1. Modulasi Frekuensi

Modulasi adalah proses berubahnya salah satu parameter suatu sinyal pembawa yang disebabkan oleh sinyal pemodulasi. Pada sistem modulasi frekuensi parameter yang berubah adalah frekuensi sinyal pembawa.

Untuk menganalisa spektrum gelombang FM, maka persamaan harus duraikan terlebih dahulu dengan langkah-langkah sebagai berikut :

$$e(t) = E_c[\sin \varpi_c t \cos(m \sin \varpi_m t) + \cos \varpi_c t \sin(m \sin \varpi_m t)]$$

dimana

$$\cos(m \sin \varpi_m t) = J_0(m) + 2\sum_{n=1} J_{2n}(m) \cos 2\pi \varpi_m t$$

$$\sin(m \sin \varpi_m t) = 2\sum_{n=0} J_{2n+1}(m) \sin(2n+1)\varpi_m t$$

bentuk $J_n(m)$ merupakan suatu fungsi Bessel bentuk pertama dengan orde-n. Maka persamaan gelombang FM menjadi :

$$e(t) = E \ \sin \varpi_c t[J_0(m) + 2J_2(m) \cos 2\varpi_m t + 2J_4(m) \cos 2\varpi_m t]$$
$$+ E_c \cos \varpi_c t[2J_1(m) \sin \varpi_m t + 2J_3(m) \sin \varpi_m t + ...]$$

atau dapat ditulis dalam bentuk :

$$e(t) = J_0(m) E_c \sin \varpi_c t$$

$$+J_1(m)E_c[\sin(\varpi_c+\varpi_m)t-\sin(\varpi_c-\varpi_m)t]$$
$$+J_2(m)E_c[\sin(\varpi_c+2\varpi_m)t-\sin(\varpi_c-2\varpi_m)t]$$
$$+J_3(m)E_c[\sin(\varpi_c+3\varpi_m)t-\sin(\varpi_c-3\varpi_m)t]$$
$$+J_4(m)E_c[\sin(\varpi_c+4\varpi_m)t-\sin(\varpi_c-4\varpi_m)t]$$

2.1.2. Deviasi Frekuensi

Deviasi frekuensi adalah pergeseran sinyal pembawa dari frekuensi asal menuju frekuensi yang lebih tinggi atau rendah. Secara matematis, deviasi frekuensi didefinisikan sebagai :

$$\Delta_f = KE_m \cos\varpi_m t$$

Keterangan :

Δ_f = deviai frekuensi (Hz)

K = konstanta ddeviasi (Hz/V)

E_m = amplitudo maksimum sinyal pemodulasi (Volt)

t = waktu (detik)

2.1.3. Indeks Modulasi

Indeks modulasi adalah perbandingan deviasi frekuensi pembawa dengan frekuensi pemodulasi.

$$m = \frac{\Delta_f}{f_m}$$

Keterangan :

M = indeks modulasi

Δ_f = deviasi frekuensi (Hz)

f_m = frekuensi sinyal pemodulasi (Hz)

2.1.4. Lebar pita gelombang FM

Lebar pita aktual yang dibutuhkan untuk melewatkan seluruh pita sisi yang berarti adalah dua kali hasil perkalian dari frekuensi sinyal pemodulasi tertinggi dan banyaknya pita sisi yang berarti (*Significant sideband*) yang ditentukan dari tabel fungsi Bessel. Secara matematis, aturan untuk menentukan lebar pita untuk termodulasi sudut dengan menggunakan tabel fungsi Bessel adalah :

$$B = 2(nXF_m)$$

dimana :

B = lebar pita (Hz)

N = banyaknya pita sisi berarti yang ditentukan dari tabel fungsi Bessel

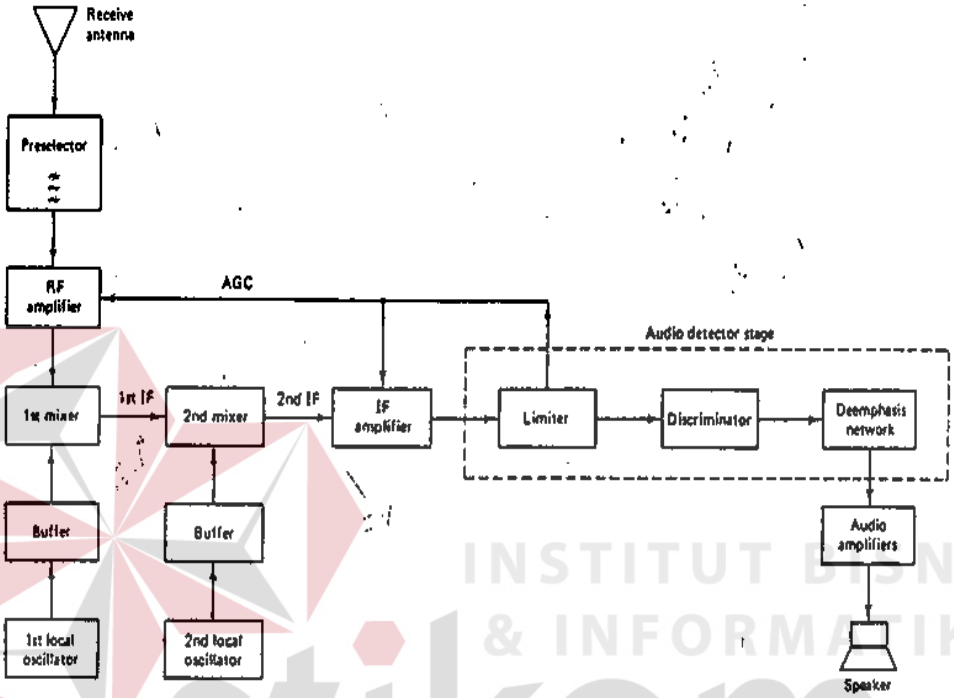
F_m = frekuensi sinyal pemodulasi tertinggi (Hz)

2.2. Penerima FM

Gambar 2.0. menunjukkan diagram blok sederhana dari penerima FM *SuperHeterodyne* konversi ganda. Seperti terlihat bahwa penerima FM sejenis dengan penerima AM. Penguat RF, pencampur dan tingkat frekuensi antara (IF *stages*) adalah sama dengan penerima AM, tetapi penerima FM biasanya membutuhkan penguatan frekuensi lebih besar. Detektor puncak yang biasanya dipakai dalam penerima AM digantikan oleh pembatas (*limiter*), diskriminator frekuensi dan jaringan *deemphasis*. Pembatas dan jaringan *deemphasis* memberikan penambahan perbandingan S/N yang akan diumpankan ke tingkat demodulator. Frekuensi tinggi IF pertama relatif tinggi (biasanya 10,7 MHz) yang berfungsi untuk menolak frekuensi bayangan (*image frequency*). Sedangkan tingkat IF kedua memiliki frekuensi yang lebih rendah (biasanya 45 5 KHz, berbentuk keramik *filter*) yang dipakai untuk mengurangi kemungkinan terjadinya osilasi dari penguat.

Untuk dapat mendeteksi sinyal FM, diperlukan suatu rangkaian yang keluarannya berubah secara linier, sesuai dengan frekuensi dari sinyal masukan. Beberapa rangkaian yang sering dipergunakan adalah : Detektor kecuraman (*Slope*

Detector), Diskriminator Foster-Seeley. Detektor perbandingan (*Ratio Detector*), Detektor Reaktif (*Quadrature Detector*) dan Diskriminator loop fasa terkunci (*Phase Lock Loop Demodulator*).



Gambar 2.0. Blok diagram penerima FM

2.3. Komunikasi Data

Tujuan dari komunikasi adalah mengirimkan data atau pesan dari suatu tempat ke tempat lain dengan jarak yang mungkin berjauhan dan untuk memastikan data atau pesan dapat diterima dengan baik serta dimengerti. Komunikasi data didefinisikan sebagai suatu bentuk komunikasi antara suatu

piranti dengan piranti lain seperti terminal ke komputer atau komputer ke komputer.

2.4. Macam hubungan dalam Sistem Komunikasi

Dalam sistem komunikasi antara dua buah peralatan terdapat tiga macam operasi yang dapat digunakan yaitu : s*implex, half duplex, dan full duplex*. Hubungan *simplex* digunakan untuk mengirimkan data satu arah saja. Pada h*alf duplex*, komunikasi terjadi antara dua peralatan yang terhubung dengan pertukaran data secara bergantian. Sedangkan *full duplex*, pertukaran informasi dapat dilakukan secara bersamaan sekaligus.

Data yang dikirimkan oleh suatu terminal adalah informasi yang telah dikodekan dalam bentuk tertentu. Informasi ini terbentuk dari simbol-simbol yang disebut karakter. Agar dapat ditransmisikan, karakter tersebut harus dikodekan menjadi sekelompok bit. Pengkodean ini dilakukan oleh komputer yaitu standar ACSII (*American Standard Code for Information Interchange*).

2.5. Kecepatan pengiriman data

Kecepatan pengiriman data adalah aspek yang harus diperhatikan dalam komunikasi data. Untuk suatu bentuk gelombang biner, laju bit sama dengan laju pengiriman sinyal dan dinyatakan dalam bit per detik. Jika Γ adalah waktu yang diperlukan untuk memancarkan 1 bit, maka laju pengiriman sinyal r adalah :

$$r = \frac{1}{\Gamma}$$

Bila suatu bentuk gelombang dari kode biner dengan periode 2Γ dikirim melalui saluran transmisi yang mempunyai lebar jalur frekuensi (B), maka frekuensi sinyal tersebut dinyatakan dengan :

$$B \geq \frac{r}{2}$$

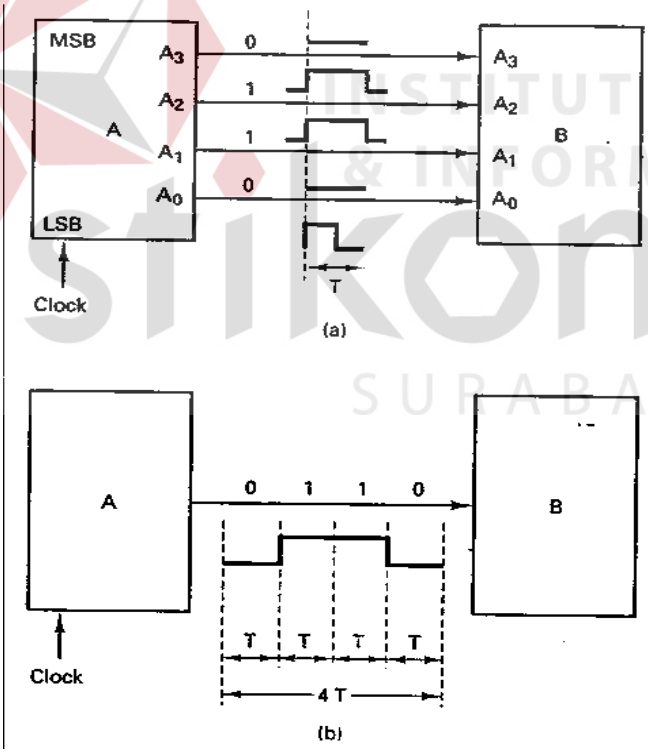
Selanjutnya ini dikenal sebagai kriteria *nyquist*, yang berarti untuk laju pengiriman sinyal r, lebar jalur tersempit yang dapat digunakan adalah :

$$B = \frac{r}{2}$$

2.6. Metode Pengiriman Data

2.6.1. Komunikasi data serial dan paralel

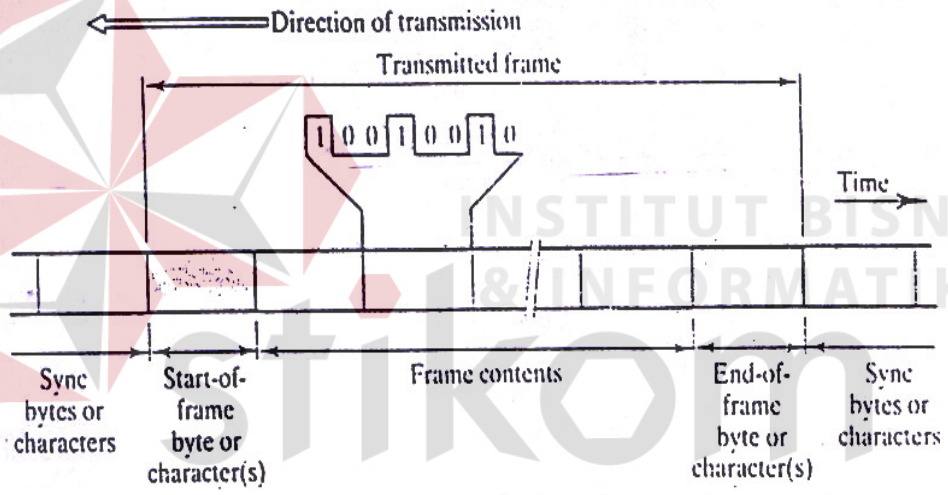
Dalam pengiriman data, bit-bit yang membentuk karakter ditransmisikan secara paralel atau serial. Pada sisstem serial, dibutuhkan hanya satu kanal saluran transmsisi. Setiap bit data dikirim berurutan satu demi satu. Sedang dalam sistem paralel, dibutuhkan kanal sejumlah bit pembentuk karakter. Jadi untuk standar pengkodean ACSII yang mengkodekan delapan bit setiap karakter, dibutuhkan delapan kanal untuk komunikasi data dengan sistem paralel ini. Delapan bit pembentuk karakter tersebut ditransmisikan sekaligus dalam satu periode waktu. Gambar 2.1. (a) menunjukkan sistem pengiriman data secara paralel, sedangkan untuk gambar 2.1. (b) adalah sistem pengiriman data dalam modus serial.



Gambar 2.1. Modus Transmisi (a) Paralel, (b) Serial

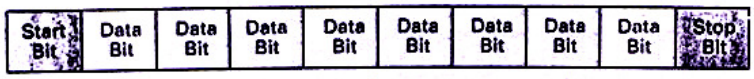
2.6.2. Komunikasi data sinkron dan asinkron

Cara pengiriman data ada dua macam yaitu sinkron dan asinkron. Pada modus sinkron karakter yang dikirimkan tergabung dalam satu blok. Blok data ini diapit dengan karakter kontrol yang menunjukkan awal dan akhir dari suatu blok. Karakter kontrol ini berupa karakter khusus berdasarkan protocol yang digunakan. Pada modus ini tidak ada penambahan bit *start stop* ke karakter. Gambar 2.2. dibawah ini menunjukkan pentransmisian data pada modus sinkron.



Gambar 2.2. Transmisi Sinkron

Sedangkan pada modus asinkron setiap karakter ditransmisikan secara terpisah astu dengan yang lainnya. Pada awal karakter ditambahkan *start bit* dan diakhir karakter ditambahkan *stop bit*. Besarnya bit yang akan dikirim dalam satu *start bit* dan satu *stop bit* ialah 8 bit. Gambar 2.3. dibawah ini menunjukkan contoh karakter yang ditransmisikan dalam modus asinkron.

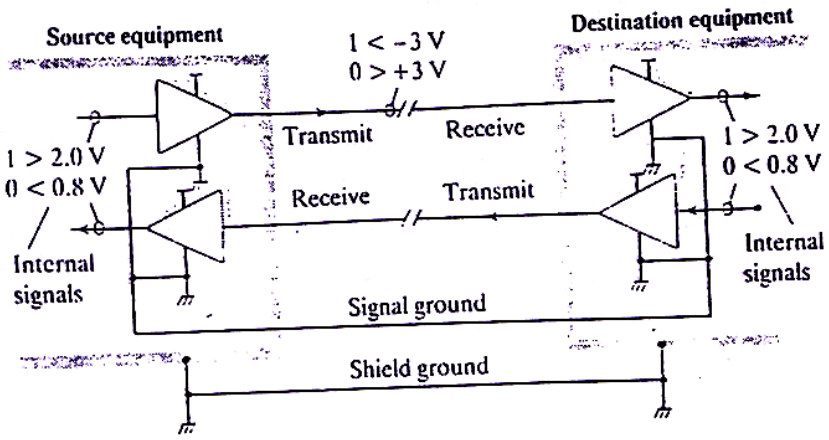


Gambar 2.3. Transmisi Asinkron

2.7. Antarmuka RS232C/V.24

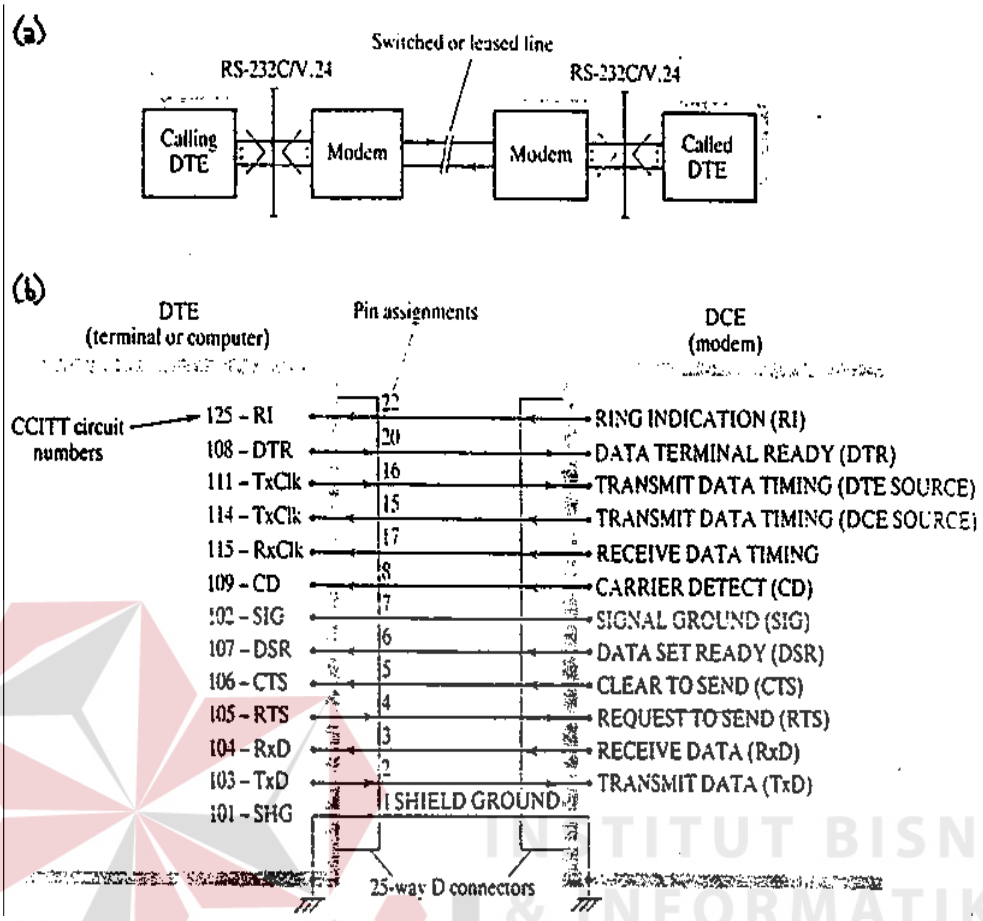
Antarmuka RS232C (distandarkan oleh *Electrical Industries Accosiation,* EIA) dan V.24 (distandarkan oleh *Consultative Commite of the International Telegraph and Telephone*, CCITT) adalah suatu standar antarmuka untuk menghubungkan peralatan terminal data (*Data Terminal Equipment*, DTE) dengan peralatan komunikasi data (*Data Communication Equipment*, DCE) agar memungkinkan beberapa industri penghasil peralatan yang berbeda data menggunakan fasilitas pengiriman data yang tersedia dalam jaringan telepon, salah satu peralatan komunikasi data adalah MODEM. Dalam hal lain peralatan ini juga dapat dipakai untuk menghubungkan peralatan periperal yang berorientasi karakter, seperti unit peraga "Display", printer, dan sebagainya dengan komputer. Jarak maksimum yang diijinkan antara DTE dan DCE sesuai standar RS232C dan V.24 adalah 15 meter dan kecepatan pengiriman data kurang dari 9600 bit per detik (9600 bps).

Tingkat sinyal yang digunakan dalam RS232C (V.24) bersama dengan antarmuka ditunjukkan dalam gambar 2.4. dibawah ini.



Gambar 2.4. Tingkat sinyal RS232C/V.24

Sedangkan gambar 2.5. (a) dibawah ini menunjukkan penempatan standar antarmuka antara dua peralatan terminal data; gambar 2.5. (b) dibawah ini menunjukkan sinyal kontrol yang digunakan oleh RS232C/V.24.



Gambar 2.5. Definisi sinyal RS232C/V.24

Dengan penempatan sinyal seperti dalam gambar 2.5. (b) diatas maka terminal dan komputer mengirim dan menerima data dalam jalur yang sama jika MODEM menyediakan fungsi yang sama untuk kedua peralatan. Jika standar antarmuka RS232C hendak digunakan untuk menghubungkan peralatan periperal yang berorientasi karakter seperti unit peraga, printer, dan sebagainya dengan komputer maka harus diputuskan untuk memilih salah satu peralatan, periperal, atau komputer untuk menggantikan fungsi MODEM.

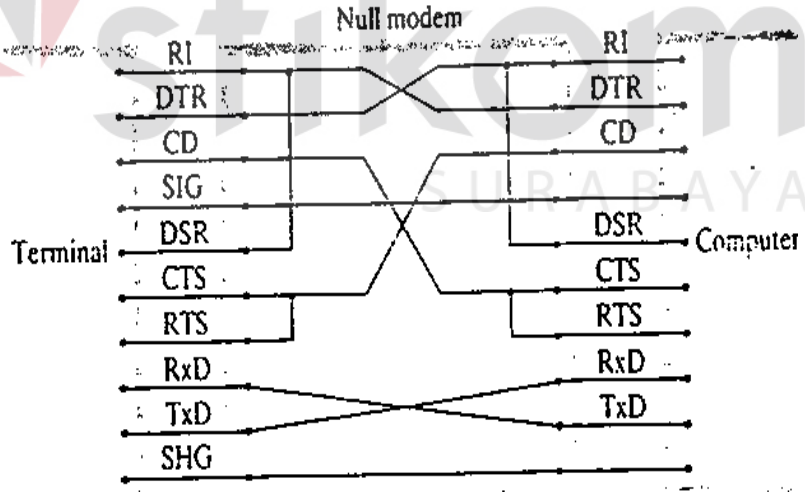
Ada tiga alternatif kemungkinan dalam keadaan ini :

1. Terminal menggantikan fungsi MODEM dan definisi jalur harus sesuai.

2. Komputer menggantikan MODEM, atau

3. Terminal dan komputer tidak berubah tetapi interkoneksi kawat harus diubah.

Dua alternatif pertama lebih sulit dipakai karena terminal dan komputer tidak dapat langsung dipakai sebagai MODEM. Sedangkan alternatif ketiga lebih sering dipakai karena hanya mengubah susunan pengawatan yang menhubungkan terminal dan komputer. Susunan pengawatan untuk menggantikan fungsi MODEM, yang disebut *null modem* dapat dilihat dalam gambar 2.6. dibawah ini.



Gambar 2.6. Pengawatan *Null Modem*

2.8. Komunikasi data biner

Salah satu bagian penting dari komunikasi data biner adalah proses modulasi dan demodulasi, hal ini disebabkan proses ini dapat mempengaruhi kecepatan transmsisi dari pengirim ke penerima.

Modulasi adalah suatu proses yang mengubah satu atau beberapa karakteristik dari gelombang pembawa (*carrier*) sesuai dengan gelombang pemodulasi.

Secara matematis sebuah sinyal listrik dapat diwakili oleh deret dari fungsi sinus atau kosinus yang dinyatakan dalam persamaan berikut :

$$v = V\sin(2\pi Ft + \theta) atau v = V\cos(2\pi Ft + \theta)$$

dimana :

v = tegangan gelombang berubah waktu

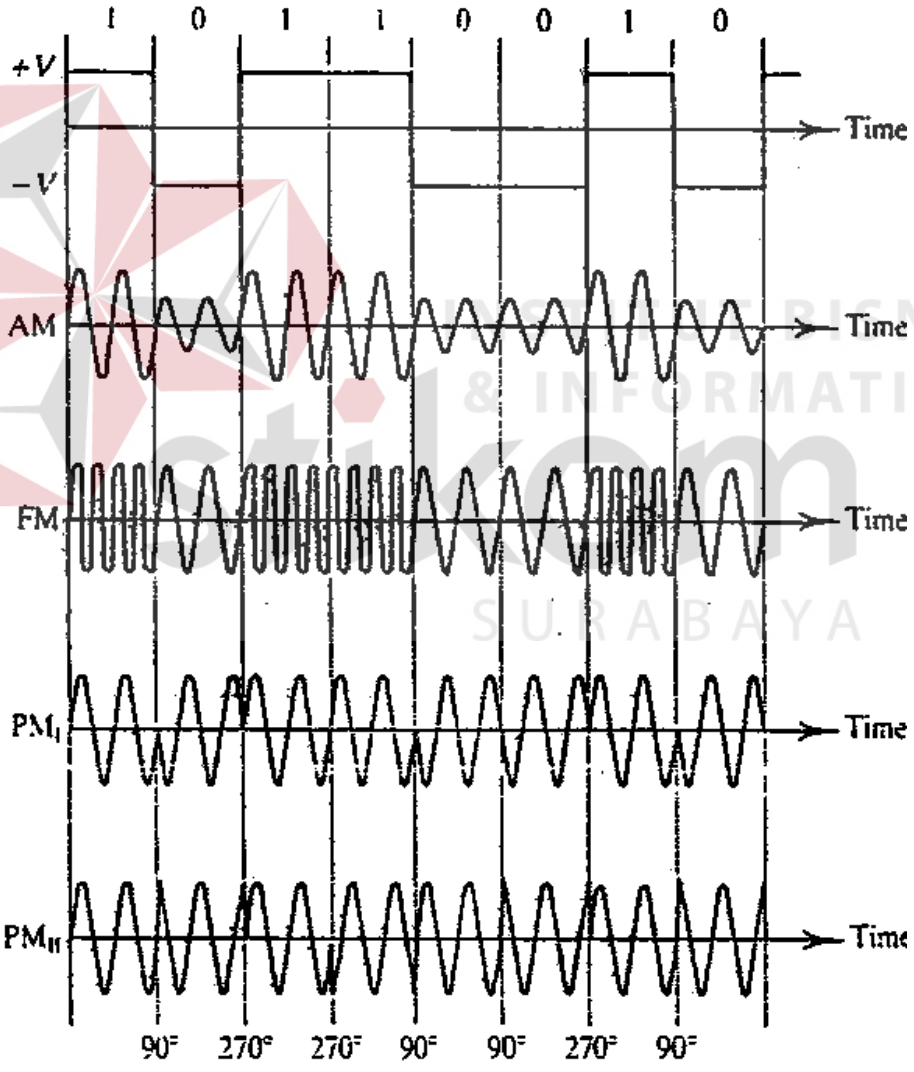
V = tegangan puncak (Volt)

F = frekuensi (Hz)

t = waktu (detik)

θ = fasa (derajat)

Dari persamaan diatas dapat dilihat bahwa untuk memodulasikan sebuah gelombang pembawa sinus dapat dilakukan dengan tiga cara, yaitu : perubahan amplitudo (tegangan puncak), fasa, dan frekuensinya sesuai dengan informasi yang ditransmisikan. Yang paling umum adalah mengubah amplitudonya, berganti antara nol dan satu tingkat amplitudo lain, sistem ini dinamakan dengan *amplitudo shift keying* dan hal ini ditunjukkan dalam gambar 2.7. dibawah ini.



Gambar 2.7. Metode Modulasi

Begitu juga halnya dengan *Phase Shift Keying*, fasa dari gelombang pembawa bergantian sebesar radian atau 180^{o}. Pada *Frequency Shift Keying*, gelombang pembawa bergantian antara dua frekuensi yang sudah ditentukan sebelumnya. *Frequency Shift Keying* sering disebut sebagai *Digital FM*.

Modulasi *Amplitudo Shift Keying* merupakan tipe modulasi yang paling sederhana, tetapi mudah dipengaruhi oleh redaman yang ditimbulkan oleh perubahan kondisi perambatan. Modulasi *Frequency Shift Keying* adalah metode yang sering dipakai karena rangkaian demodulator relatif sederhana dan lebar pita yang digunakan lebih sempit, sedangkan *Phase Shift Keying* membutuhkan rangkaian demodulator yang cukup kompleks dan juga tipe modulasi ini sangat rentan terhadap perubahan fasa dari gelombang yang ditransmisikan. Hal tersebut diatas merupakan alasan penulis untuk memilih *Frequency Shift Keying* sebagai metode modulasi dalam skripsi ini.

2.8.1. Pembangkit *Frequency Shift Keying*

Pada modulasi FSK, frekuensi sinyal pembawa diubah-ubah antara dua nilai yang berbeda, sesuai dengan persamaan berikut ini :

$$S1(t) = A.Cos(\varpi_c t - \varpi_d t)$$

$$S1(t) = A.COs(\varpi_c t + \varpi_d t)$$

dimana :

A = amplitudo sinyal.

ω_1,ω_2 = frekuensi untuk menyampaikan digit 0 dan 1.

ω_c = frekuensi pembawa (*carrier*)

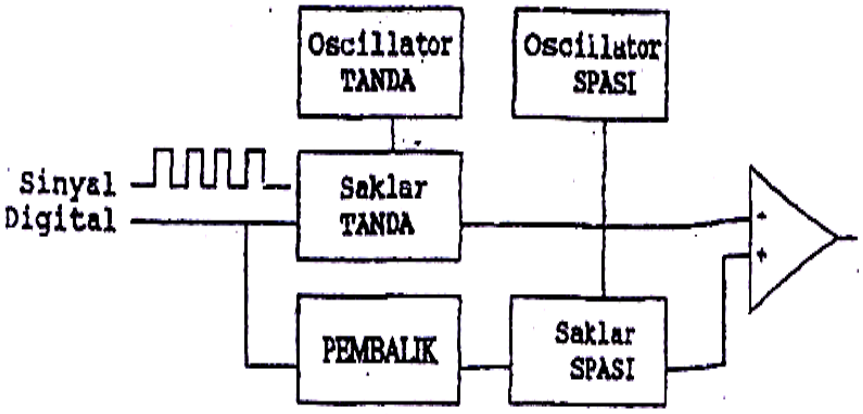
ω_d = pergeseran frekuensi

Persamaan diatas untuk menyampaikan digit 0 dan 1 secara berurutan. Dari persamaan itu tampak bahwa dua nilai frekuensi yang berbeda yaitu :

$$\varpi_1 = \varpi_c t - \varpi_d t$$

$$\varpi_2 = \varpi_c t + \varpi_d t$$

Diagram blok pembangkit (modulator) FSK ditunjukkan dalam gambar 2.8. dibawah ini.



Gambar 2.8. Diagram blok modulator FSK

Modulator FSK terdiri dari 2 osilator lokal yang mempunyai frekuensi berbeda yaitu f_1 dan f_2. Apabila masukan diberi logika 1, maka osilator dengan frekuensi f_1 *on* dan osilator dengan frekuensi f_2 *off*. Dengan demikian modulator menghasikan frekuensi f_1. Sebaliknya apabila masukan diberi logika 0, dengan adanya rangkaian pembalik, odilator dengan frekuensi f_2 akan *on*, dan osilator dengan frekuensi f_1 *off*. Jadi pada keadaan ini modulator menghasilkan frekuensi f_2.

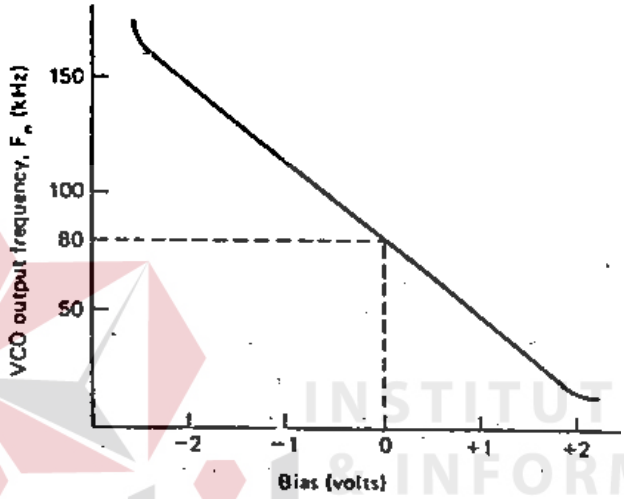
2.8.2. *Phase Lock Loop* (PLL, loop fasa terkunci)

Phase Lock Loop (PLL) adalah sistem kendali umpan balik loop tertutup dengan sinyal umpan balik suatu tegangan yang dihasulkan dari perbedaan fasa dua frekuensi. PLL terdiri atas pembanding fasa (*Phase Comparator*), *Low Pass Filter* (LPF) dan *Voltage Control Oscilator* (VCO).

Pembanding fasa adalah pecamput tak linier dengan dua masukan yaitu frekuensi yang dibangkitkan dari luar (F_i) dan sinyal keluaran VCO (F_o), Keluaran pembanding fasa adalah hasil kali (*Product*) F_o dan F_i, oleh sebab itu menghasilkan frekuensi-frekuensi sebagai berikut F_i, F_o, $F_i - F_o$, dan $F_i + F_o$.

VCO adalah sebuah osilator dengan frekuensi osilasi yang stabil yang tergantung pada tegangan bias. Keluaran VCO adalah frekuensi dan masukkannya

adlah tegangan bias dc atau tegangan kontrol. Jika sebuah tegangan dc atau tegangan ac dengan perubahan yang lambat digunakan sebagai tegangan bias PLL, maka perubahan atau simpangan frekuensi keluarannya akan sesuai dengan perubahan tegangan. Gambar 2.9. dibawah ini menunjukkan kurva perpindahan dari sebuah VCO.



Gambar 2.9. Karakteristik masukan VCO terhadap keluaran

Frekuensi keluaran VCO pada tegangan bias 0 volt disebut sebagai frekuensi alamiah (*Natural Frequency*) dan perubahan frekuensi akbat perubahan tegangan bias disebut frekuensi simpangan. Jadi frekuensi $F_o = F_n + \Delta F$. Secara matematis keluaran dari pembanding fasa adalah :

$$V_d = (Sin2\pi F_o t + \theta_o) x (VSin2\pi F_i t + \theta_i)$$

$$= \frac{v}{2} Sin(2\pi F_o t + \theta_o - 2\pi F_i t - \theta_i)$$

$$= \frac{v}{2} Sin(2\pi F_o t + \theta_o + 2\pi F_i t + \theta_i)$$

Jika PLL terkunci ($F_o = F_i$), maka persamaan adalah sebagai berikut :

$$V_d = \frac{v}{2} Sin(\theta_o - \theta_i) - \frac{v}{2} Sin[2(2\pi F_o t) + \theta_o + \theta_i]$$

dan setelah melalui tapis lolos bawah maka komponen :

$$\frac{v}{2} Sin[2(2\pi F_o t) + \theta_o + \theta_i]$$

akan hilang dan keluaran pembanding fasa menjadi :

$$V_d = \frac{v}{2} Sin(\theta_o - \theta_i)$$

dimana $\theta_o - \theta_i = \theta_e$ disebut kesalahan fasa sehingga V_d berupa tegangan dc yang dibutuhkan untuk mengubah frekuensi keluaran VCO dari F_n ke F_o.

2.8.3. Pendeteksian *Frequency Shift Keying* (FSK)

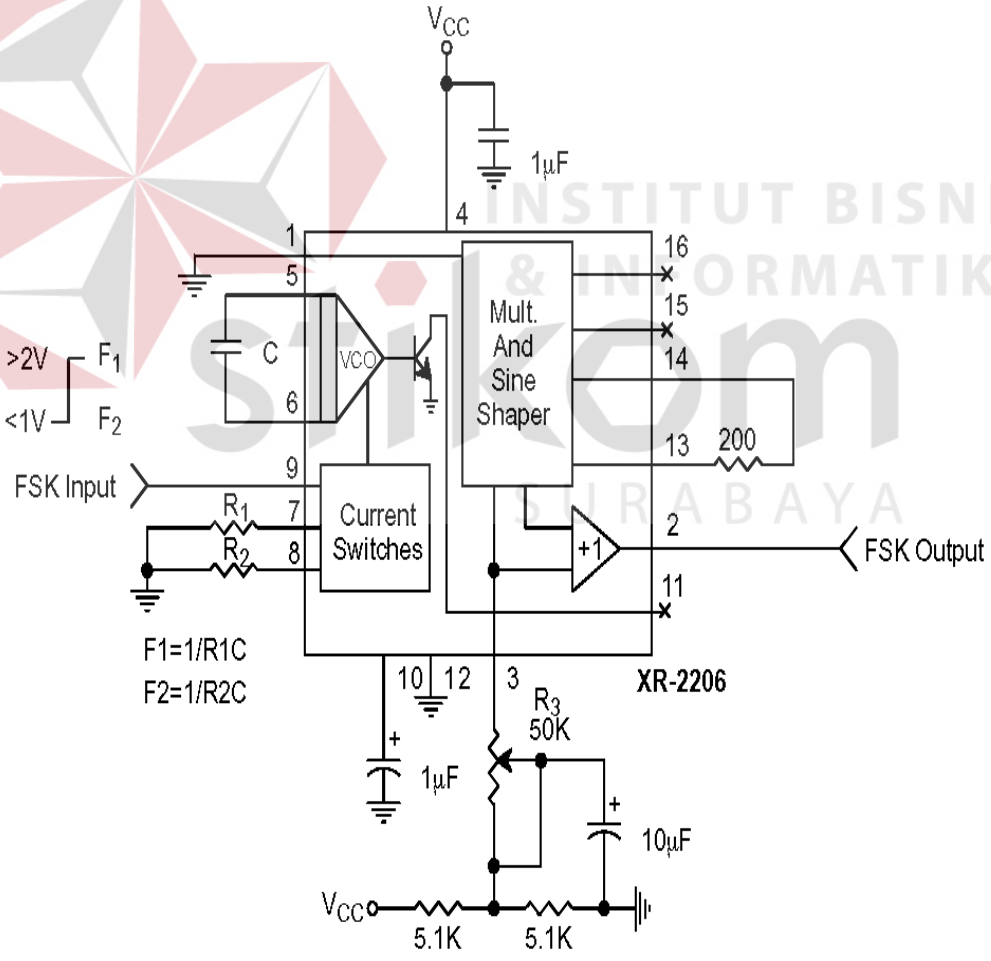
Pendeteksian FSK disebut juga dengan pendemodulasian sinyal yang dimodulasi FSK, dan hal ini dilakukan dengan menggunakan PLL.

2.8.3.1. Modulasi FSK

Pada proses pemodulasian sinyal dapat dibuat persamaan secara umum sebagai berikut :

$$F = \frac{1}{R.C}$$

Gambar 2.10.1 dibawah ini adalah bentuk rangkaian *Modulator* FSK dengan menggunakan IC XR2206 :



Gambar 2.10.1 Rangkaian *Modulator* FSK menggunakan IC XR2206

Dari bentuk rangkaian *Modulator* FSK tersebut dapat dibuat persamaan untuk menentukan besarnya nilai dari masing – masing komponen adalah sebagai berikut :

$$F_1 = \frac{1}{R_1 C} \qquad\qquad F_2 = \frac{1}{R_2 C}$$

Dimana :

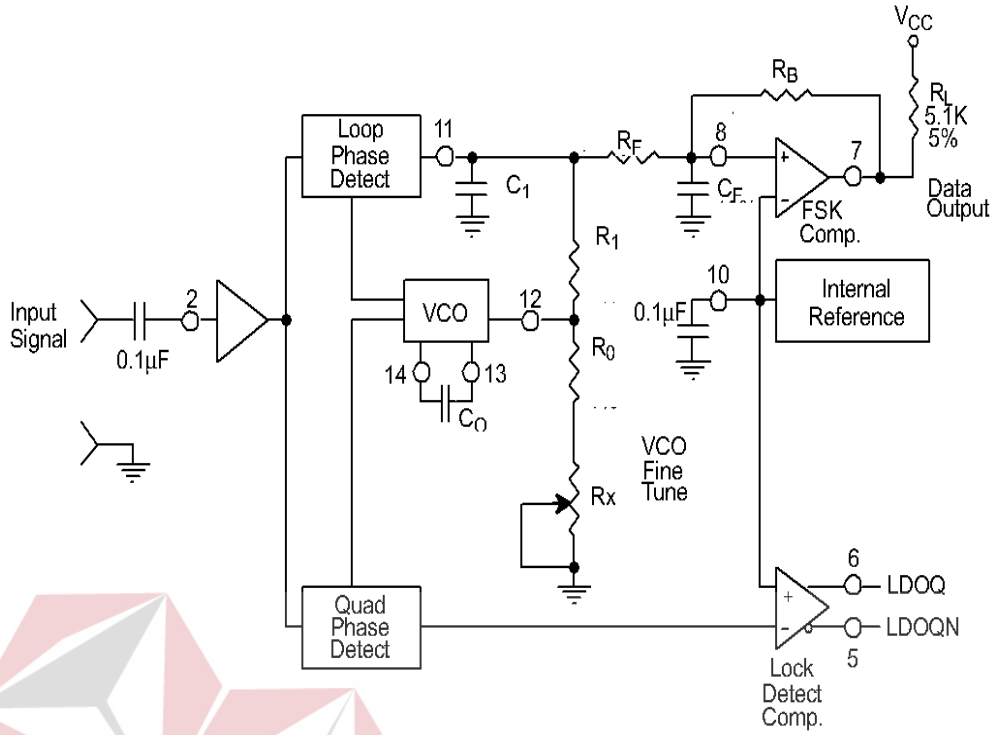
F_1 = Frekuensi saat bit bernilai '0'

F_2 = Frekuensi saat bit bernilai '1'

IC XR 2206 adalah jenis IC *monolithic function general* yang mampu menghasilkan gelombang *sinus*, persegi, ramp dan segitiga dengan kestabilan frekuensi cukup akurat.

2.8.3.2. Demodulasi FSK

Proses *demodulasi* FSK ini, mengubah kembali sinyal dari hasil modulasi yang dilakukan oleh *modulator* FSK kembali menjadi bentuk sinyal *biner.* Gambar 2.10.2. dibawah ini memperlihatkan rangkaian *demodulator* FSK menggunakan IC XR2211.



Gambar 2.10.2. Gambar rangkaian *Demodulator* FSK menggunakan IC XR2211

Rangkaian *demodulator* FSK ini bekerja dengan sistem PLL yang mempunyai kemampuan dalam ketepatan dan mempertahankan sebuah frekuensi berada pada posisinya. PLL ini dapat dibangkitkan dengan menggunakan sebuah persamaan sebagai berikut :

$$f_o = \sqrt{F_1.F_2}$$

Dimana :

F_1 = Frekuensi *Osilator* saat kondisi '0'

F_2 = Frekuensi *Osilator* saat kondisi '1'

Besarnya frekuensi lokal (f_o) yang dibangkitkan adalah sama dengan frekuensi yang digunakan untuk menunjukkan biner 0 dan biner 1, sehingga sinyal FSK dapat didemodulasi dan kembali didapatkan data biner 0 dan 1.

Selain itu pada PLL menggunakan prinsip perhitungan berdasarkan waktu dan pada rangkaian tersebut diatas ada beberapa komponen yang digunakan untuk mengaktifkan, yaitu *timing resistor* (R_o) dengan persamaan sebagai berikut :

$$R_0 = R_0 + \frac{R_x}{2}$$

Dimana R_x dapat diberikan sebuah *variable resistor* untuk membantu memperoleh ketepatan dalam nilai yang diinginkan.

Selain *timing resistor* juga dibutuhkan sebuah *timing* capasitor (C_0) dengan persamaan :

$$C_0 = \frac{1}{R_0 . f_0}$$

Selain menentukan besarnya frekuensi, juga harus diperhatikan masalah *BandWidth*. Pada rangkaian diatas komponen – komponen yang menentukan besarnya *BandWidth* yang dibutuhkan dapat digunakan persamaan :

$$R_1 = \frac{R_0 . f_0}{(F_1 . F_2)} .2$$

Dimana : R_1 = nilai *resistor* untuk menentukan besarnya *Band Width*.

Pada PLL terdapat *loop filter* yang dapat membuang terjadinya *phase error* sehingga hanya sinyal yang asli yang dapat masuk. PLL *loop filter* ini terdapat pada C_1 dengan persamaan :

$$C_1 = \frac{1250.C_0}{R_1.\varsigma^2}$$

Dimana : (ζ = 0,5) = *Loop Damping*

Untuk nilai R_F diambil dari 5 kali besarnya R_1 yaitu :

$$R_F = R_1.5$$

Untuk nilai R_B diambil dari 5 kali besarnya R_F yaitu :

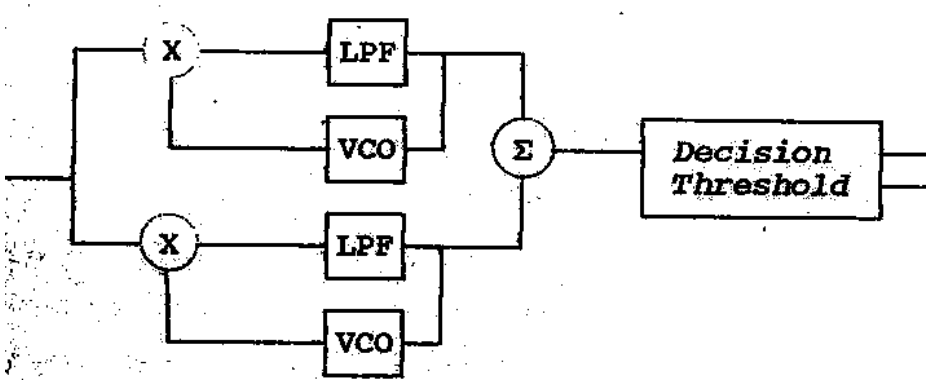
$$R_B = R_F.5$$

Untuk besarnya *Data Filter Capacitance* (C_F) didapat dari persamaan :

$$R_{sum} = \frac{(R_F + R_1).R_B}{(R_1 + R_F + R_B)}$$

$$C_F = \frac{0{,}25}{(R_{sum}.Baudrate)} \; Baudrate \text{ dalam } \frac{1}{\text{sec } onds}$$

Dibawah ini digambarkan (gambar 2.10.3) diagram blok pendeteksi FSK.



Gambar 2.10.3. Diagram blok pendeteksian FSK

2.9. Pandangan umum tentang Mikrokontroler INTEL 8031

Mikrokontroler 8031 merupakan salah satu anggota keluarga dari MCS-51, yaitu suatu komponen produksi INTEL yang berorientasi pada kontrol (*Microcontroller*) serta oleh INTEL sendiri diklasifikasikan dalam kelompok *embedded microcontroller*, yaitu berarti mikrokontroler yang dapat diprogram ulang (*reprogrammable*).

Spesifikasi perangkat keras dari mikrokontroler 8031 adalah :

⇨ Ukuran pengolah data : 8 bit

⇨ Ruang memori program : 64 k*Byte*

⇨ Ruang memori data : 64 k*Byte*

⇨ Kapasitas RAM : 128 *Byte*

⇨ I/O : 32 jalur *bi-directional* dan setiap bit dapat dialamati

⇨ *Counter/timer* 16 bit : 2 buah

⇨ UART : *full duplex*

⇨ Interupsi : 5 jalur dengan 2 tingkat prioritas dapat diprogram

⇨ Osilator : Internal

⇨ Proses *Boolean* (logika untuk 1 bit)

⇨ Pelaksanaan instruksi per siklus 1 mikrodetik pada frekuensi *clock* 12 MHz

Keluarga MCS-51 yang diproduksi oleh INTEL mempunyai konfigurasi yang berbeda-beda sesuai dengan jenisnya. Dalam tabel 2.1. dibawah ini diperlihatkan sebagian keluarga MCS-51.

Tipe	Tipe tanpa EPROM	Tipe ber-EPROM	ROM	RAM (byte)	I/O
8051	8031	-	4K	128	4
8051AH	8031AH	8751H	4K	128	4
		8751BH			
8052AH	8032AH	8752BH	8K	256	4
80C51BH	80C31BH	87C51	4K	128	4
83C51FA	80C51FA	87C51FA	8K	256	4
83C51FB	80C51FA	87C51FB	16K	256	5

Tabel 2.1. Keluarga MCS-51

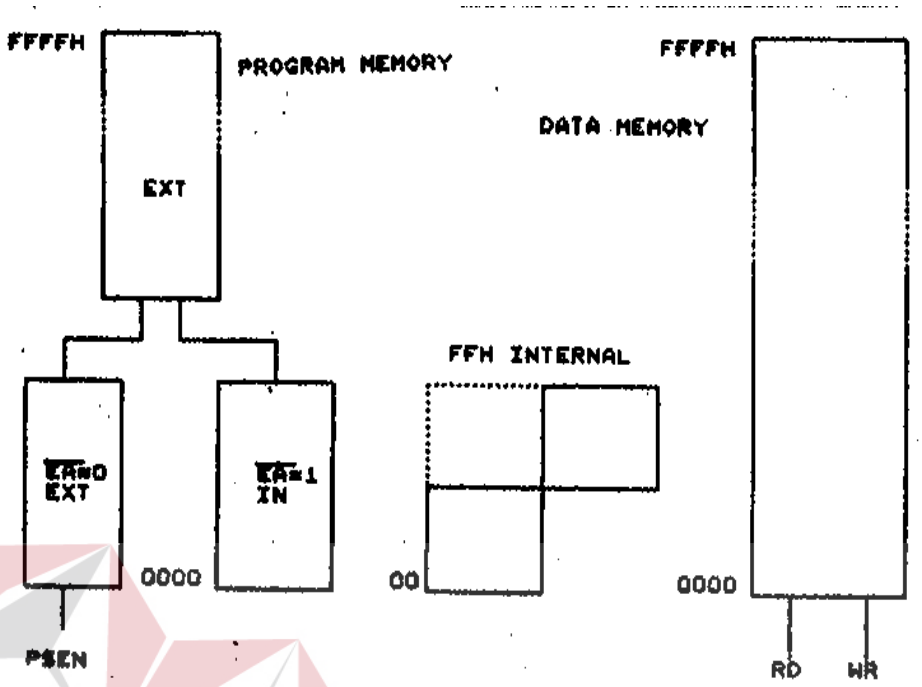
2.10. Arsitektur Mikrokontroler 8031

2.10.1. Memori

Organisasi memori mikrokontroler 8031 dapat dibagi atas bagian yang berbeda, yaitu memori program dan memori data. Pembagian tersebut didasarkan atas fungsinya dalam penyimpanan data atau program. Memori program digunakan untuk menyimpan instruksi-instruksi yang akan dijalankan mikrokontroler, sedangkan memori data digunakan sebagai tempat menyimpan data-data yang sedang diolah mikrokontroler.

Memori program disimpan dalam EPROM dan memori data disimpan dalam RAM. Lebar alamat memori program selalu 16 bit meskipun alamat yang digunakan lebih kecil dari 64 k*Byte*. Lebar alamat memori data internal adalah 8 bit dan eksternal adalah 8 bit atau 16 bit.

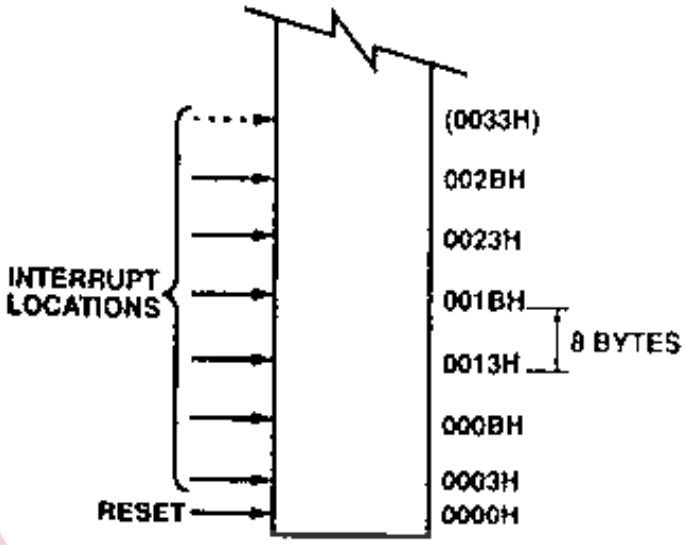
Gambar 2.11. dibawah ini memperlihatkan struktur perbedaan pada memori data dan memori program. Pada memori program, ruang memori dapat diperluas sampai 64 k*Byte*. Untuk memilih penggunaan memori data internal atau memori data eksternal digunakan penyemat EA (*external access enable*). Setiap eksekusi memori program eksternal dipakai sinyal baca PSEN (*program store enable*).



Gambar 2.11. Struktur memori mikrokontroler

a. Memori Program

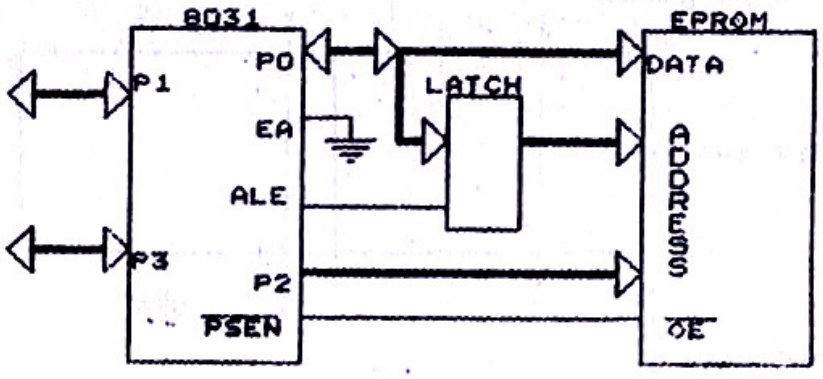
Pada gambar 2.12. dibawah ini diperlihatkan bagian bawah dari memori program. Setelah *reset* CPU memulai eksekusi dari lokasi 0000H.



Gambar 2.12. Memori program bagian bawah

Setiap interupsi mempunyai lokasi tetap dalam memori program. Interupsi menyebabkan CPU melompat ke lokasi tersebut dimana pada lokasi tersebut terdapat sub-rutin yang harus dilaksanakan.

Susunan perangkat keras yang menggunakan EPROM eksternal diperlihatkan pada gambar 2.13 dibawah ini.



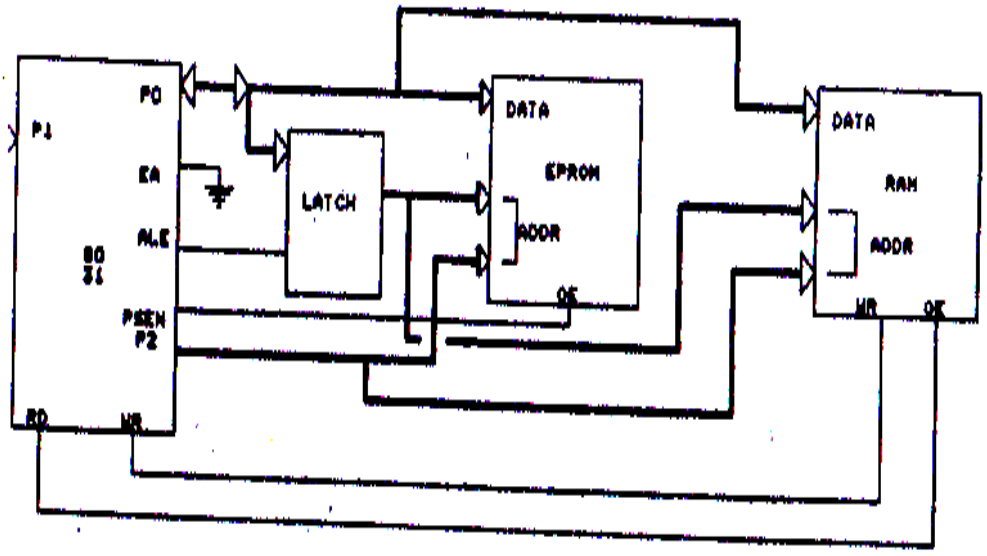
Gambar 2.13. Mengakses memori program

Port 0 dan *Port* 2 digunakan untuk menghubungkan EPROM, digunakan sebagai *bus* data dan *bus* alamat. *Port* 0 memultipleks alamat dan data. *Port* ini mengirimkan *byte* bawah dari program *counter* sebagai suatu alamat dan kemudian *port* ini akan berada pada keadaan mengambang menunggu datangnya kode *byte* dari memori program. Selama waktu *byte* bawah dari program *counter* sah (*valid*) pada *port* 0, sinyal ALE dikirimkan sehingga *byte* bawah program *counter* akan di *latch*.

Sementara itu *port* 2 mengirimkan *byte* atas program *counter*. Baru kemudian $\overline{PSEN}$ mengirimkan sinyal ke EPROM untuk dapat dibaca kode *byte*-nya oleh mikrokontroler. Alamat memori program selalu 16 bit lebarnya walaupun jumlah memori program yang digunakan kurang dari 64 k*Byte*

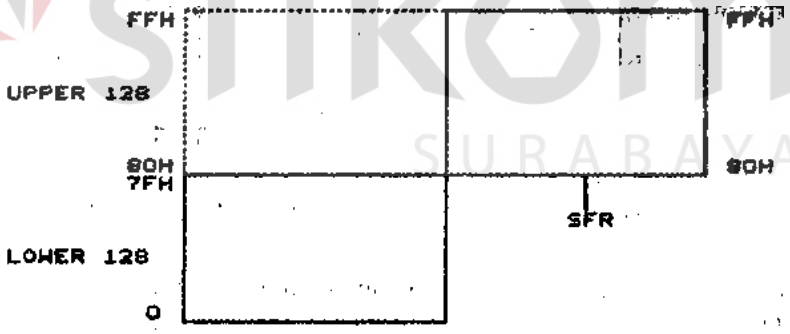
b. Memori Data

Gambar 2.14. dibawah ini memperlihatkan hubungan mikrokontroler 8031 untuk mengakses RAM eksternal. Untuk melakukan pembacaan atau penulisan, mikrkontroler akan mengirimkan sinyal $\overline{RD}$ atau $\overline{WR}$. RAM yang digunakan mempunyai kapasitas 8 k*Byte*.



Gambar 2.14. Menghubungkan mikrokontroler 8031 dengan RAM eksternal

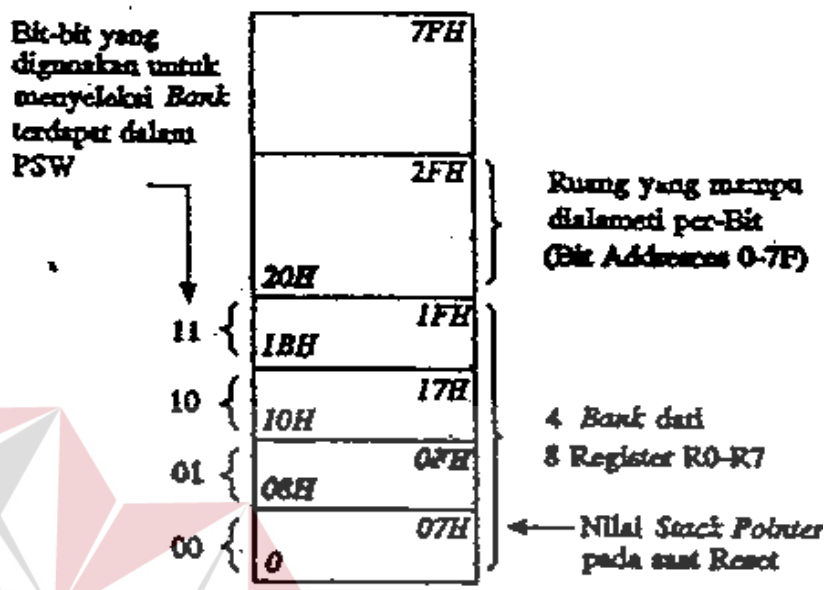
Memori data internal dipetakan seperti pada gambar 2.15. dibawah ini. Ruang memorinya dibagi menjadi tiga blok, yaitu sebagai *lower* 128, *upper* 128, dan ruang SFR (*special function register*).



Gambar 2.15. Memori data internal

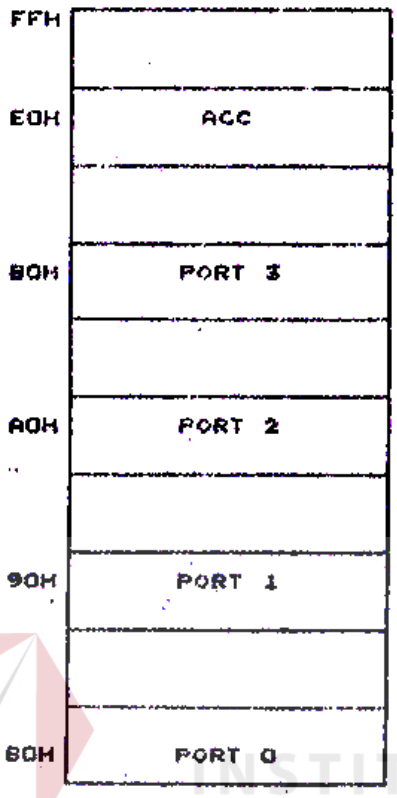
Bagian bawah dari 128 *byte* RAM dipetakan seperti terlihat pada gambar 2.16 dibawah ini. Tiga puluh dua *byte* paling bawah dikelompokkan dalam 4 *bank*

(8 *register*), yaitu R0 sampai R7. Dua bit dalam PSW (*program status word*) memilih *register bank* yang digunakan.



Gambar 2.16. Bagian bawah 128 *byte* RAM *internal*

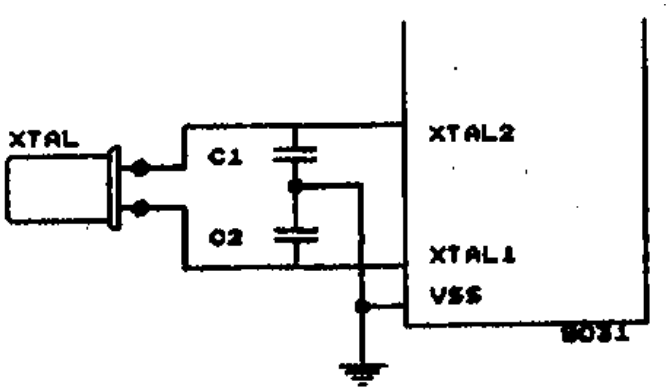
Gambar 2.17 dibawah ini menunjukkan tuang SFR. SFR ini berisi penahan *port* (*port latch*), pewaktu (*timer*), pengontrol periperal, dan lain-lain. *Register* ini hanya dapat diakses oleh pengalamatan langsung.



Gambar 2.17. Ruang *special function register*

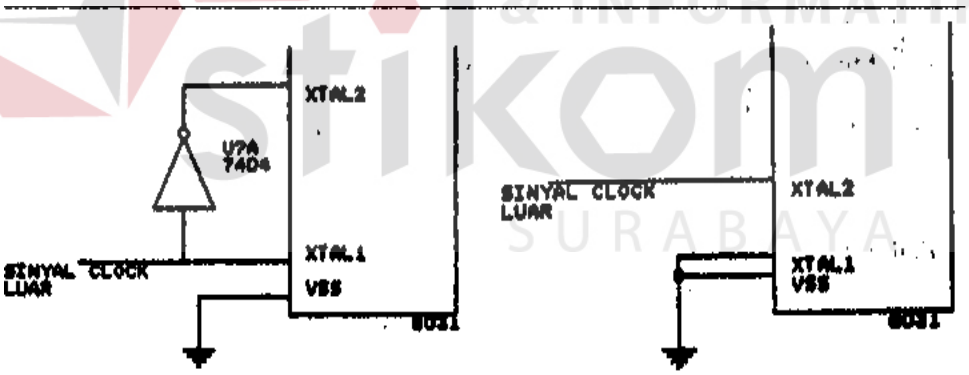
2.10.2. Pewaktuan CPU

Mikrokontroler 8031 memiliki osilator internal (*on chip oscillator*) yang dapat digunakan sebagai sumber *clock* bagi CPU. Untuk menggunakan osilator internal diperlukan sebuah kristal atau *resonator* keramik antara pena XTAL 1 dan pena XTAL 2 dan sebuah kapasitor ke *ground* seperti pada gambar 2.18 dibawah ini.



Gambar 2.18. Menggunakan osilator internal

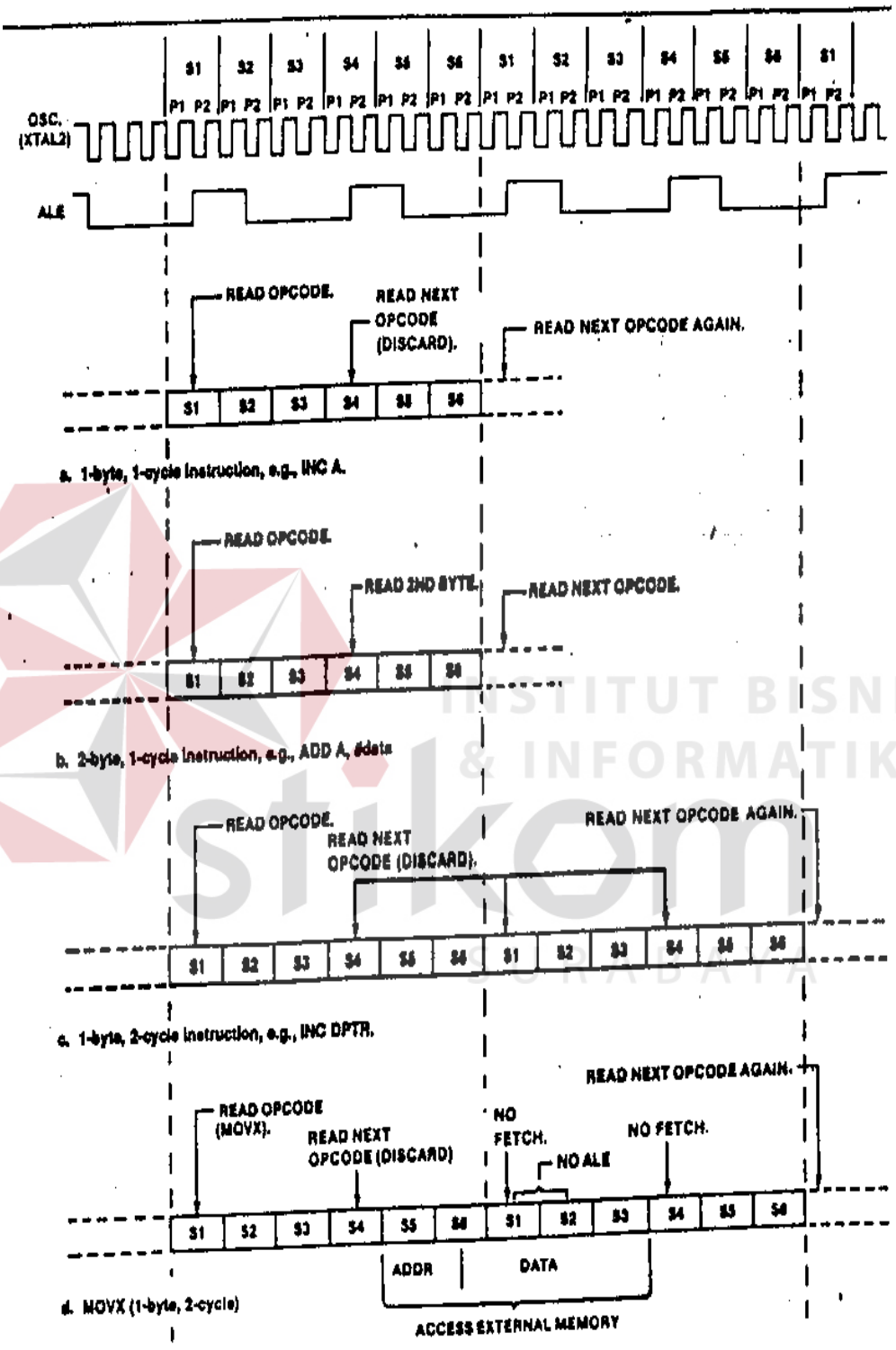
Untuk kristalnya dapat digunakan frekuensi dari 6 sampai 12 MHz, sedangkan untuk kapasitor dapat bernilai antara 27 pF sampai 33 pF. Bila menggunakan *clock* eksternal, rangkaiannya dihubungkan seperti dalam gambar 2.19 dibawah ini.



Gambar 2.19. Menggunakan sumber *clock* eksternal

2.10.3. Siklus Mesin

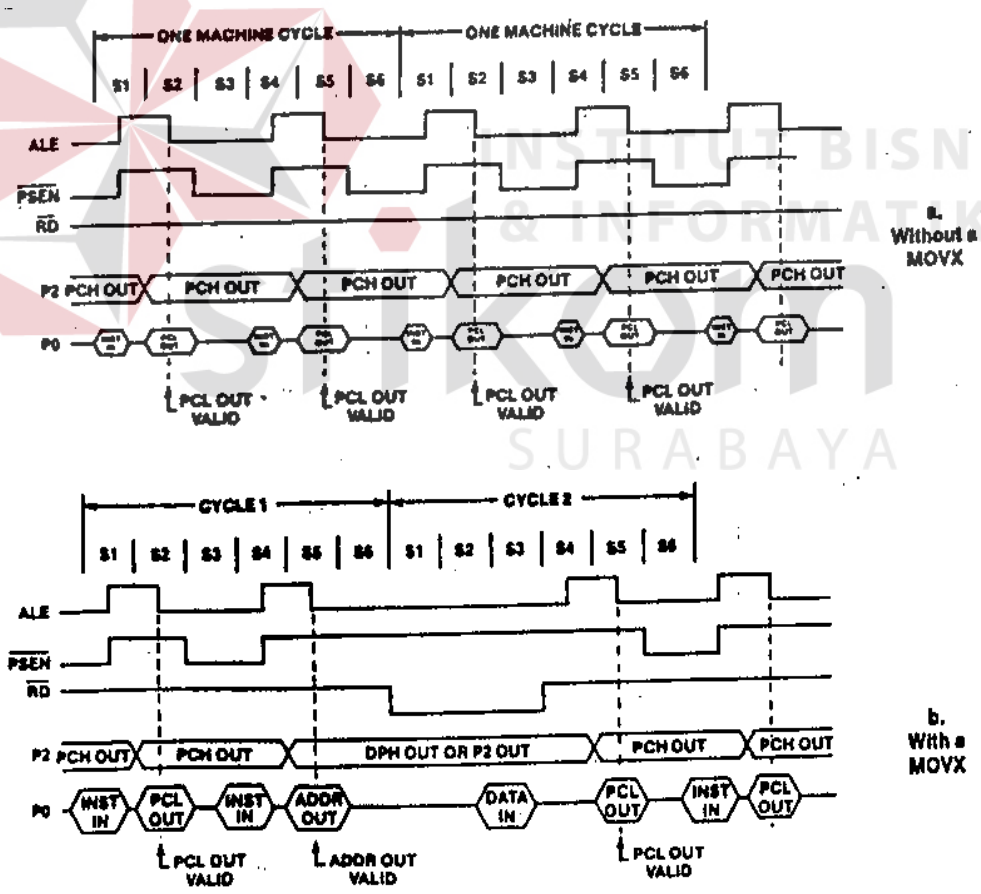
Satu siklus mesin (*machine cycle*) berisi urutan 6 keadaan, diberi nomor S1 sampai S6. Setiap keadaan waktu adalah sepanjang dua periode osilator, sehingga siklus mesin membutuhkan 12 periode osilator atau 1 mikrodetik jika frekuensi osilator 12 MHz. Gambar 2.20 dibawah ini memperilhatkan urutan pengambilan/eksekusi program untuk beberapa instruksi.



Gambar 2.20. Urutan pengambilan/eksekusi program

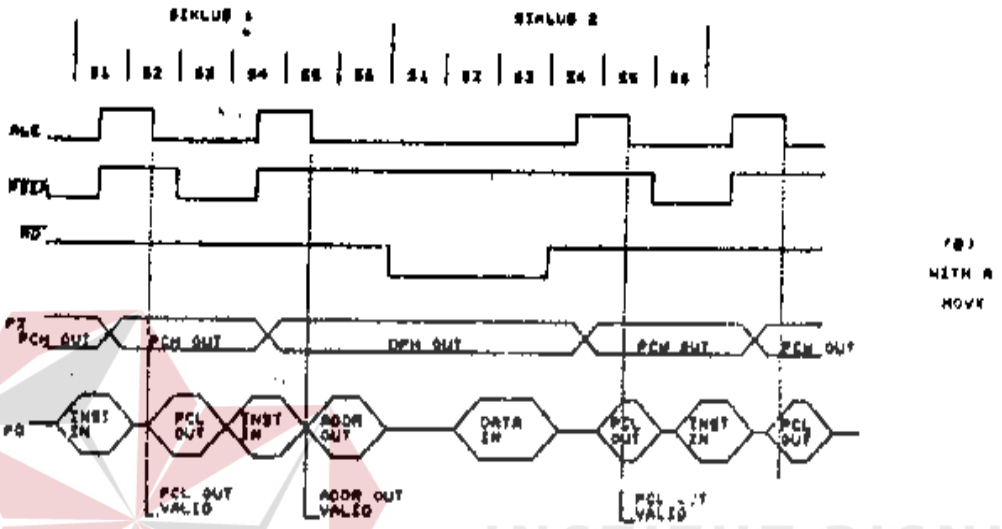
Eksekusi satu siklus instruksi dimulai keadaan 1 (*state* 1) dari siklus mesin, saat kode operasi (*opcode*) ditahan dalam *register* instruksi (IP, *instruction register*). Pengambilan kedua terjadi selama S4 pada siklus mesin yang sama. Eksekusi secara lengkap selesai pada akhir keadaan 6 dari siklus mesin.

Urutan eksekusi adalah sama untuk memori program eksternal maupun internal. Waktu eksekusi tidak tergantung pada memori program internal atau eksternal. Gambar 2.21 dibawah ini memperlihatkan sinyal dan pewaktuan dalam mengambil program jika memakai memori program eksternal.



Gambar 2.21. Siklus *bus* pada pengaksesan memori program eksternal

Jika memakai memori program eksternal, sinyal $\overline{PSEN}$ diaktifkan dua kali setiap siklus mesin. Apabila terjadi akses pada memori data eksternal, sinyal $\overline{PSEN}$ terlihat seperti pada gambar 2.22 dibawah ini.



Gambar 2.22. Sinyal $\overline{PSEN}$

2.10.4. Interupsi

Apabila CPU pada mikrokontroler 8031 sedang melaksanakan suatu program, kita dapat menghentikan pelaksanaan program tersebut secara sementara dengan meminta interupsi. Apabila CPU mendapat permintaan interupsi, *program counter* (PC) akan diisi alamat dari vektor interupsi. CPU kemudian melaksanakan rutin pelayanan interupsi mulai dari alamat tersebut. Bila rutin pelayanan interupsi selesai dilaksanakan. CPU 8031 kembali ke pelaksanaan program utama

yang ditinggalkan. Pada mikrokontroler INTEL 8031 terdapat beberapa saluran interupsi. Interupsi pada 8031 dibedakan dalam 2 jenis, yaitu :

1. Interupsi yang tak dapat dihalangi oleh perangkat lunak (*non maskable interrupt*), misalnya *reset*.

2. Interupsi yang dapat dihalangi perangkat lunak (*maskable interrupt*). Controh interupsi jenis adalah $\overline{INT}0$ dan $\overline{INT}1$ (eksternal) serta *timer counter* 0, *timer counter* 1, dan interupsi dari *port serial* (internal).

Instruksi RETI (*return from interrupt*) harus digunakan untuk kembali dari layanan rutin interupsi. Instruksi ini dipakai agar saluran interupsi kembali dapat dipakai. Alamat awal layanan rutin interupsi dari setiap sumber interupsi diperlihatkan dalam tabel 2.2 dibawah ini.

NAMA	Lokasi	Alat Interupsi
Reset	00H	*Power on reset*
$\overline{INT}0$	03H	INT 0
Timer 0	0BH	Timer 0
$\overline{INT}1$	13H	INT 1
Timer 1	1BH	Timer 1
Sint	23H	*Port I/O serial*

Tabel 2.2. Alamat layanan rutin interupsi

Mikrokontroler INTEL 8031 menyediakan 5 sumber interupsi dengan 2 interupsi eksternal. 2 interupsi *timer* dan satu interupsi *port serial*. Interupsi

eksternal $\overline{INT0}$ dan $\overline{INT1}$ masing-masing dapat diaktifkan berdasarkan level atau transisi, tergantung pada bit IT0 dan IT1 dalam TCON. *Flag* yang menghasilkan interupsi ini adalah bit dalam IE0 dan IE1 dari TCON.

Interupsi *Timer 0* dan *Timer 1* dihasilkan oleh TF0 dan TF1. Interupsi *port serial* dihasilkan oleh logika OR dari R1 dan T1.

Ada dua buah *register* yang mengontrol interupsi, yaitu IE (*interrupt enable*) dan IP (interrupt priority). Prosesor 8031 tidak menanggapi permintaan interupsi jika suatu instruksi belum dilaksanakan secara lengkap (dapat selama empat siklus).

a. ***Interrupt Enable***

Setiap sumber interupsi dapat diaktifkan maupun dilumpuhkan secara individual dengan mengatur satu bit di SFR yang bernama IE (*interrupt enable*). Bit-bit IE didefinisikan sebagai berikut :

MSB **LSB**

EA	-	-	ES	ET1	EX1	ET0	EX0

Simbol	Posisi	Fungsi
EA	IE.7	Melumpuhkan semua interupsi. Jika EA=0 tidak ada interupsi yang akan dilayani. Jika EA=1 setiap sumber interupsi dapat dijalankan atau dilumpuhkan secara individual.

-	IE.6	Kosong
-	IE5.	Kosong
ES	IE.4	Bit pembuat *enable port serial*
ET1	IE.3	Bit pembuat *enable timer 1*
EX1	IE.2	Bit pembuat *enable* $\overline{INT1}$
ET0	IE.1	Bit pembuat *enable timer 0*
EX0	IE.0	Bit pembuat *enable* $\overline{INT0}$

Jika akan mengaktifkan interupsi 0 ($\overline{INT0}$), misalnya nilai yang harus diberi ke IE adalah 81H (yaitu memberikan logika 1 ke EA dan EX0).

b. ***Interrupt Priority***

Setiap sumber interupsi dapat diprogram secara individual (sendiri-sendiri) menjadi satu atau dua tingkat prioritas dengan mengatur bit pada SFR yang bernama IP (*Interrupt Priority*). Interupsi dengan prioritas rendah (*Low Priority*) dapat diinterupsi oleh interupsi yang memiliki prioritas yang lebih tinggi (*High Priority*), tetapi tidak dapat diinterupsi oleh interupsi yang memiliki prioritas lebih rendah. Interupsi yang memiliki prioritas tertinggi tidak dapat diinterupsi oleh interupsi lainnya.

Jika dua permintaan interupsi dengan tingkat prioritas yang berbeda diterima secara bersamaan, permintaan interupsi dengan prioritas tertinggi yang

akan dilayani. Jika permintaan interupsi dengan prioritas yang sama diterima bersamaan, maka dilakukan *polling* untuk menentukan mana yang akan dilayani.

Bit-bit pada IP adalah sebagai berikut :

MSB							LSB
-	-	-	PS	PT1	PXI	PT0	PX0

Priority bit = 1 menandakan prioritas tinggi

Priority Bit= 0 menandakan prioritas rendah

Simbol	Posisi	Fungsi
-	IP.7	Kosong
-	IP.6	Kosong
-	IP.5	Kosong
PS	IP.4	Bit prioritas interupsi *port serial*
PT1	IP.3	Bit prioritas interupsi *timer* 1
PX1	IP.2	Bit prioritas interupsi $\overline{INT}1$
PT0	IP.1	Bit prioritas interupsi *timer* 0
PX0	IP.0	Bit prioritas interupsi $\overline{INT}0$

2.10.5. *Special Function Register*

SFR berisi *register-register* dengan fungsi tertentu. Masing-masing *register* ditunjukkan dalam tabel 2.3 dibawah ini yang meliputi simbol nama dan alamatnya

Simbol	**Nama**	**Alamat**
ACC	*Accumulator*	E0H
B	B *Register*	F0H
PSW	*Program Status Word*	D0H
SP	*Stack Pointer*	81H
DPTR	*Data Pointer* 16 bit DPL *Byte* rendah DPH *Byte* tinggi	 82H 83H
P0	*Port* 0	80H
P1	*Port* 1	90H
P2	*Port*2	A0H
P3	*Port* 3	B0H
IP	*Interrupt Priority Control*	B8H
IE	*Interrupt Enable Control*	A8H
TMODE	*Timer Counter mode control*	89H

TCON	*Timer Counter control*	88H
TH0	*Timer Counter 0 high byte*	8CH
TL0	*Timer Counter 0 low byte*	8AH
TH1	*Timer Counter 1 high byte*	8DH
TL1	*Timer Counter 1 low byte*	8BH
SCON	*Serial Control*	98H
SBUF	*Serial Data Buffer*	99H
PCON	*Power Control*	87H

Tabel 2.3. *Special Function Register*

Pada saat terjadi *power on reset*, isi dari SFR dapat dilihat pada tabel 2.4 berikut ini :

Register	**Nilai dalam biner**
ACC	0000 0000
B	0000 0000
PSW	0000 0000
SP	0000 0111
DPTR	
DPH	0000 0000
DPL	0000 0000
P0	1111 1111
P1	1111 1111
P2	1111 1111
P3	1111 1111
IP	xxx0 0000

IE	0xx0 0000
TMODE	0000 0000
TCON	0000 0000
TH0	0000 0000
TL0	0000 0000
TH1	0000 0000
TL1	0000 0000
SCON	0000 0000
PCON	0xxx xxxx

Tabel 2.4. Keadaan SFR setelah *power on reset*

2.10.6. Timer/Counter

Pada mikrokontroler 8031 terdapat dua buah *timer/counter* 16 bit yang dapat diatur melalui perangkat lunak, yaitu *timer/counter* 0 dan *timer/counter* 1.

Apabila *timer/counter* diaktifkan pada frekuensi kerja mikorkontroler 12 MHz, *timer/counter* akan melakukan perhitungan waktu sekali setiap satu mikrodetik secara independen, tidak tergantung pada pelaksanaan satu instruksi. Satu siklus pencacahan waktu berpadanan dengan satu siklus pelaksanaan instruksi, sedangkan satu siklus diselenggarakan dalam waktu satu mikrodetik. Bila dimisalkan suatu urutan instruksi telah selesai dilasanakan dalam waktu lima mikodetik, pada saat itu pula *timer/counter* telah menunjukkan periode waktu lima mikrodetik.

Apabila periode waktu tertentu telah dilampaui, *timer/counter* segera menginterupsi mikrokontroler untuk memberitahukan bahwa perhitungan periode waktu telah selesai dilaksanakan. Periode waktu *timer/counter* secara umum ditentukan oleh persamaan berikut ini :

a. sebagai *timer/counter* 8 bit

$$T = (255 - TLx) * 1\mu s$$

dimana TLx adalah isi *register* TL0 atau TL1

b. sebagai *time/counter* 16 bit

$$T = (65535 - THx * TLx) * 1\mu s$$

THx = isi *register* TH0 atau TH1

TLx = isi *register* TL0 atau TL1

Pengontrol kerja *timer/counter* adalah *register timer* kontrol (TCON). Adapun definisi dari bit-bit pada *timer* kontrol adalah sebagai berikut :

MSB							**LSB**
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Simbol	**Posisi**	**Fungsi**
TF1	TCON.7	*Timer* 1 *overflow flag*. Di-*set* oleh perangkat keras saat *timer counter* menghasilkan limpahan (*overflow*).
TR1	TCON.6	Bit untuk menjalakan *timer* 1. Di-*set* atau *clear* oleh *software* untuk membuat *timer on* atau *off*.

TF0	TCON.5	*Timer* 0 *overflow flag*. Di-*set* oleh perangkat keras.
TR0	TCON.4	Bit untuk menjalankan *Timer* 0. Di-*set* oleh *software* untuk membuat *timer on* atau *off*.
IE1	TCON.3	Eksternal *interrupt* 1 *edge flag*.
IT.1	TCON.2	*Interrupt* 1 *type* control byte. *Di*-set *clear* oleh *siftware* untuk menspesifikasikan sisi turun/*level* rendah *trigger* dari interupsi eksternal.
IE0	TCON.1	Eksternal *interrupt* 0 *edge flag*.
IT0	TCON.0	*Interrupt* 0 *type* control bit.

Pengontrol pemilihan *mode* operasi *timer/counter* adalah *register timer mode* (TMODE) yang mana definisi bit-bitnya adalah sebagai berikut :

MSB **LSB**

GATE	**C/T**	**M1**	**M0**	**GATE**	**C/T**	**M1**	**M0**

Keterangan :

GATE : saat TRx dalam TCON di-*set* 1 dan GATE = 1, *timer/counter* x akan berjalan ketika TRx = 1 (*timer* dikontrol *software*).

C/T : pemilih fungsi *timer* atau *counter*. *Clear* (0) untuk operasi *timer* dengan masukkan dari sistem *clock* internal. *Set* (1) untuk operasi *counter* dengan masukkan dari pena T0 atau T1.

M1 : bit pemilih *mode*

M0 : bit pemilih *mode*

Kombinasi M0 dan M1 adalah sebagai berikut :

M1	M0	MODEOPERASI	
0	0	0	*Timer* 13 bit
0	1	1	*Timer/counter* 16 bit
1	0	2	*Timer autoreload* 8 bit (pengisian otomatis)
1	1	3	TL0 adalah *timer/counter* 8 bit yang dikontrol oleh kontrol bit standar *timer* 0. TH0 adalah *timer* 8 bit dan dikontrol oleh kontrol *timer* 1.

a. *Mode* 0

Dalam *mode* ini *register timer* disusun sebagai *register* 13 bit. Setelah semua perhitungan selesai, mikrokontroler akan men-*set timer interrrupt flag* (TF1). Dengan membuat GATE = 1, *timer* dapat dikontrol oleh masukkan luar $\overline{INT1}$, untuk fasilitas pengukuran lebar pulsa.

b. *Mode* 1

Mode 1 sama dengan *mode* 0 kecuali *register timer* akan bekerja dalam 16 bit.

c. *Mode* 2

Mode 2 menyusun *register timer* 8 bit *counter*. Limpahan (*overflow*) dari TL1 tidak hanya men-*set* TF1 tetapi juga mengisi TL1 dengan isi TH1 yang diatur secara *software*. Pengisian ini tidak mengubah TH1.

d. *Mode* 3

Timer 1 dalam *Mode* 3 semata-mata memegang hitungan. Efeknya sama sperti men-*set* TR1 = 0. *Timer* 0 dalam *mode* 3 menetapkan TL0 dan TH0 sebagai dua *counter* terpisah. TL0 menggunakan kontrol bit *timer* 0 yaitu C/T, GATE, TR0, $\overline{INT}0$, dan TF0. TH0 ditetapkan sebagai fungsi *timer*.

Mode 3 diperlukan untuk aplikasi yang membutuhkan *timer/counter* ekstra 8 bit. Dengan *timer* 0 dalam *mode* 3, mikrokontroler 8031 seperti memiliki 3 *timer/counter*. Saat *timer* 0 dalam *mode* 3, *timer* 1 dapat dihidupkan atau dimatikan, atau dapat digunakan oleh *port serial* sebagai pembangkit *baud rate*.

2.10.7. Melakukan *Setting Timer*

Tabel 2.5 sampai 2.8 ini memberikan beberapa nilai bagi TMOD yang dapat digunakan untuk melakukan *setting timer* 0 dalam *mode* yang berbeda.

Diasumsikan hanya satu *timer* yang digunakan. Jika diinginkan untuk menjalankan *timer* 0 dan *timer* 1 secara bersamaan, dalam beberapa *mode* nilai

TMOD harus di-OR-kan dengan nilai seperti terlihat untuk *timer* 1 (tabel 2.7 dan tabel 2.8). Sebagai contoh, jika diinginkan untuk menjalankan *timer* 0 dalam *mode* 1 GATE (kontrol eksternal) dan *timer* 1 dalam *mode* 2 *counter*, maka nilai yang harus diisikan pada TMOD adalah 69H (nilai 09H dari tabel 1 di-OR-kan dengan 60H dari tabel 2.7).

09H ——————————— 0000 1001

60H ——————————— 0110 0001

——————————— OR

69H ——————————— 0110 1001

2.10.8. *Timer/Counter* 0

Pada kontrol internal, *timer* dihidup-matikan dengan men-*set* bit TR0 (kontrol *software*). Pada kontrol eksternal, *timer* dihidup-matikan dengan memberikan logika 0 pada pena $\overline{INT0}$ (kontrol *hardware*).

Mode	**Fungsi *Timer* 0**	**TMOD**	
		Kontrol Internal	**Kontrol Eksternal**
0	13 bit *timer*	00H	08H
1	16 bit *timer*	01H	09H
2	8 bit *autoreload*	02H	0AH
3	Dua 8 bit *timer*	03H	0BH

Tabel. 2.5. sebagai *Timer*

Mode	Fungsi *Counter* 0	TMOD	
		Kontrol Internal	Kontrol Eksternal
0	13 bit *timer*	04H	0CH
1	16 bit *timer*	05H	0DH
2	8 bit *autoreload*	06H	0EH
3	Dua 8 bit *timer*	07H	0FH

Tabel. 2.6. sebagai *Counter*

2.10.9. *Timer/Counter* 1

Mode	Fungsi *Timer* 1	TMOD	
		Kontrol Internal	Kontrol Eksternal
0	13 bit *timer*	00H	80H
1	16 bit *timer*	10H	90H
2	8 bit *autoreload*	20H	A0H
3	Dua 8 bit *timer*	30H	B0H

Tabel. 2.7. sebagai *Timer*

Mode	Fungsi *Counter* 1	TMOD	
		Kontrol	Kontrol

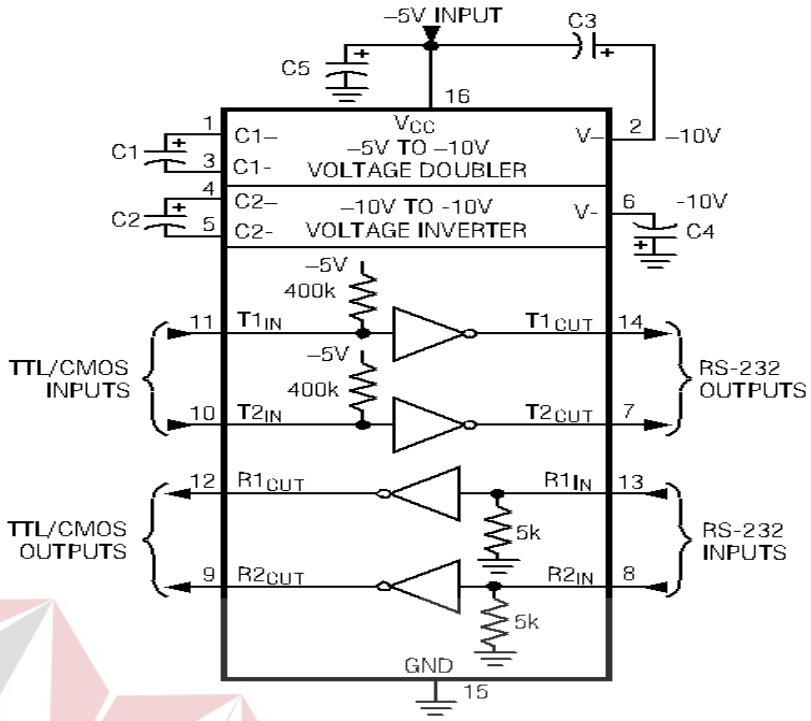
		Internal	Eksternal
0	13 bit *timer*	40H	C0H
1	16 bit *timer*	50H	D0H
2	8 bit *autoreload*	60H	0EH

Tabel 2.8. sebagai *Counter*

2.10.10. *Port Serial*

Mikrokontroler 8031 juga dilengkapi dengan *port serial*. *Port serial* memungkinkan kita mengirim data dalam format *serial*.

Apabila hendak menghubungkan mikrokontroler 8031 dengan PC (komputer pribadi) melalui *port serial*, level TTL harus diubah menjadi *level* RS232. Untuk keperluan ini dapar digunakan IC MAX 232. Hubungannya diperlihatkan pada gambar 2.23 dibawah ini.



Gambar 2.23. Pengubah *level* TTL ke RS232 dengan IC MAX 232

Port serial dalam mikrokontroler 8031 memiliki sifat *full duplex*, yang berarti dapat mengirim dan menerima data secara bersamaan. *Register* penerima dan pengirim pada *port serial* diakses pada SBUF (*serial buffer*). *Register* pengontrol kerja *port serial* ini adalah SCON (*serial control*). Bit-bit SCON ini didefinisikan sebagai berikut :

MSB **LSB**

SM0	SM1	SM2	REN	TB8	RB8	T1	R1

Simbol	**Posisi**	**Fungsi**
SM0	SCON.7	Pemilih *mode port serial*
SM1	SCON.6	Pemilih *mode port serial*
SM2	SCON.5	Membuat *enable* komunikasi *multiprocessor* dalam *mode* 2 dan *mode* 3

REN	SCON.4	*Set/clear* oleh perangkat lunak untuk menjalankan/melumpuhkan penerimaan
TB8	SCON.3	Bit ke-9 yang akan dikirimkan dalam *mode* 2 dan *mode* 3. *Set/clear* secara *software*
RB8	SCON.2	Dalam *mode* 2 dan *mode* 3 adalah bit data ke-9 yang diterima. Dalam *mode* 0 RB8 tidak digunakan
T1	SCON.1	*Transmit interrupt flag*. Di-*set* oleh perangkat keras pada akhir waktu ke-8 dalam *mode* 0, atau pada permulaan dari bit *stop* dalam *mode* lainnya. Di *clear* secara *software*
R1	SCON.0	*Receive Interrupt flag*. Di-*set* oleh perangkat keras pada akhir waktu bit ke-8 dalam *mode* 0

SM0	SM1	Mode	Keterangan	Baud Rate
0	0	0	*Shift Register*	Frek. Osc/12
0	1	1	8 bit UART	Variabel
1	0	2	9 bit UART	Frek. Osc/64 atau Frek Osc/32

a. *Mode* 0

Data *serial* masuk dan keluar melalui RxD. TxD mengeluarkan *clock* pergeseran (*shift clock*). Data 8 bit dikirimkan/diterima dengan bagian yang pertama masuk sebagai LSB. *Baud rate* tetap pada 1/12 frekuensi osilator.

b. *Mode* 1

Sepuluh bit dikirim melalui TxD atau diterima melalui RxD. Format ke-10 bit tersebut adalah sebagai berikut :

Start	*	*	*	*	*	*	*	*	**Stop**

(0) LSB MSB (1)

Pada saat diterima *stop* bit, bit ini akan masuk ke RB8 pada kontrol *serial* (SCON). *Baud rate* dapat diubah-ubah.

c. *Mode* 2

Sebelas bit dikirim melalui TxD atau diterima melalui RxD. Format datanya adalah sebagai berikut :

Start	*	*	*	*	*	*	*	*	**Prog**	**Bit Stop**

(0) LSB MSB (1)

Bit data ke-9 (TB8 dalam SCON) dapat menunjukkan nilai 0 atau 1. Sebagai contoh, bit paritas (P dalam PSW) dapat dimasukkan dalam TB8. *Baud rate* dapat diprogram menjadi 1/32 atau 1/64 dari frekuensi osilator.

d. *Mode* 3

Sebelas bit dikirim melalui TxD dan diterima melalui RxD. Format datanya adalah sebagai berikut :

Start	*	*	*	*	*	*	*	*	**Prog. Bit**	**Stop**

(0) LSB MSB (1)

Pada kenyataannya *mode* 3 sama dengan *mode* 2 kecuali dalam masalah *baud rate*. Dalam *mode* 3, *baud rate* adalah variabel.

Keempat *mode* pengiriman ditandai oleh instruksi yang menggunakan SBUF sebagai *register* tujuan. Penerimaan dalam *mode* 0 dibuat dengan R1 = 0 dan REN = 1. Tabel 2.9 dibawah ini memperlihatkan nilai yang harus diisikan pada SCON untuk masing-masing *mode*.

Mode	**SCON**	**Variabel SM2**
0	10H	
1	50H	*Single Processor*
2	90H	(SM2 = 0)
3	D0H	
0	-	
1	70H	*Multi Processor*
2	B0H	*(SM2 = 1)*
3	F0H	

Tabel 2.9. Nilai untuk SCON untuk setiap *mode*

2.10.11. *Baud Rate*

Port serial pada *mode* 0 mempunyai *baud rate* yang tetap, yaitu 1/12 dari frekuensi osilator. Untuk menjalankan dalam *mode* ini tidak ada *timer/counter* yang dibutuhkan untuk setting. Hanya *register* SCON yang diperlukan.

Port serial pada *mode* 1 memiliki *baud rate* yang dapat diubah. *Baud rate* dapat dihasilkan oleh *timer* 1. Untuk melakukan hal ini *timer* 1 digunakan dalam *mode* 2 (*auto reload*).

$$BaudRate = \frac{(K * Frekuensiosilator)}{(32 * 12 * [256 - TH1])}$$

Nilai K ditentukan oleh bit SMOD dalam *power control register* (PCON). Bila SMOD = 0 maka K = 1, bila SMOD = 1 maka K = 2.

Bila diketahui *baud rate*, nilai TH1 dapat dicari melalui persamaan berikut ini :

$$TH1 = 256 - \frac{(K * frekuensiosilator)}{(384 * Baudrate)}$$

Nilai TH1 harus dalam bentuk *integer*. Pembulatan nilai TH1 pada nilai *integer* terdekat tidak akan menghasikan *baudrate* yang dikehendaki. Dalam hal ini pemakai dapat mengganti frkuensi kristal dengan nilai 11,0592 MHz. Karena

PCON tidak dapat dialamati per bit, untuk men-*set* PCON dilakukan dengan mengirimkan perintah :

```
ORL PCON,#80H
```

Pada *port serial mode* 2, *baudrate* memiliki nilai tetap yaitu 1/12 atau 1/64 dari frekuensi osilator, tergantung pada nilai SMOD dalam *register* PCON. Pada *mode* ini tidak ada *timer* yang digunakan. Bila SMOD = 1 *baudrate*-nya 1/32 frekuensi osilator. Bila SMOD = 0, *baudrate* 1/64 frekuensi osilator. Pada *port serial mode* 3, *baudrate* dapat diatur seperti dalam *mode* 1.

2.10.12. *Program Status Word*

Program Status Word berisi beberapa bit status yang menggunakan keadaan mikrokontroler. Definisi dari bit-bit dalam PSW dijelaskan dibawah ini :

CY	AC	F0	RS1	RS0	OV	-	P

Simbol	Posisi	Fungsi/Arti
CY	PSW.7	*Carry flag*
AC	PSW.6	*Auxiliary Carry Flag*
F0	PSW.5	*Flag* 0 untuk kegunaan umum
RS1	PSW.4	Bit pemilih *Bank register*
RS0	PSW.3	Bit pemilih *Bank register*
OV	PSW.2	*Overflow flag*
-	PSW.1	*Flag* didefinisikan oleh pemakai

P	PSW.0	*Parity flag*. *Set/clear*oleh perangkat keras setiap siklus instruksi untuk menunjukkan jumlah bit 1 dalam akumulator, ganjil atau genap.

RS1 dan RS1 digunakan untuk memilih *Bank register*. Delapan buah *register* ini merupakan *register* serbaguna. Lokasinya pada awal 32 *byte* RAM internal yang memiliki alamat dari 00H sampai 1FH *Register* ini dapat diakses melalui simbol *assembler* (R0 sampai dengan R7). Pemilihan b*ank register* diperlihatkan dalam tabel 2.10 dibawah ini.

RS1	**RS0**	**Bank**	**Lokasi *Memori***
0	0	0	00H – 07H
0	1	1	08H – 0FH
1	0	2	10H – 17H
1	3	1	18H – 1FH

Tabel 2.10. Pemilihan *bank register*

Register R0 dan R1 dapat digunakan untuk pengalamatan tak langsung pada RAM internal. Sisa *register* lainnya tidak dapat digunakan untuk pengalamatan tak langsung.

2.10.13. *Power Control Register*

Definisi bit dalam *power control register* (PCON) dijelaskan berikut ini. Perlu diingat bahwa *register* PCON ini tidak dapat dialamati per bit. Adapun bit-bit dalam *register* PCON adalah :

SCON	-	-	-	**GF1**	**GF0**	**PD**	**IDL**

SMOD Bila *timer* 1 digunakan untuk menghasilkan *baudrate* dan SMOD = 1, *baudrate* akan dikalikan dua ketika *port serial* digunakan dalam *mode* 1,2, dan 3.

\- Tidak dipakai, untuk pengembangan lebih lanjut

GF1 Bit *flag* serbaguna

GF0 Bit *flag* serbaguna

PD Bit *power down*. Bila bit-bit di-*set* (1), *mode power down* akan aktif. Hanya berlaku untuk tipe CMOS.

IDL *Idle mode* bit. Bila bit ini di-*set*, akan diperoleh *mode idle*.

2.11. Perangkat Lunak Mikrokontroler 8031

Sebuah mikrokontroler 8031 tidak akan bekerja bila tidak diberikan program kepadanya. Program tersebut memberitahukan mikrokontroler apa yang harus dia lakukan. Sebuah mikrokontroler yang telah bekerja baik dengan suatu program, tidak akan bekerja lagi jika programnya diganti.

Instruksi-instruksi perangkat lunak berbeda untuk masing-masing jenis mikrokontroler. Instruksi-instruksi ini hanya bisa dipahami oleh jenis mikrokontroler yang bersangkutan. Sebuah mikrokontroler tidak dapat memahami instruksi-instruksi yang berlaku pada mikrokontroler lain. Sebagai contoh, mikrkontroler buatan Intel dengan mikrkontroler buatan Motorola

memiliki perangkat instruksi yang berbeda. Instruksi-instruksi inilah yang dikenal sebagai bahasa pemrograman sistem mikrokontroler.

2.11.1. *Operand* dan Ekspresi

Bentuk umum semua instruksi dalam *assembler* Intel 8031 dapat dituliskan sebagai berikut :

[label:] Mnemonic [operand] [,operand] [,operand] [;komentar]

Jumlah operand tergantung pada tipe *mnemonic*. Semua *operand* dapat dibagi dalam 6 kelompok, yaitu :

1. Simbol khusus *assembler*
2. Pengalamatan tak langsung
3. Data langsung
4. Pengalamatan data
5. Pengalamatan bit
6. Pengalamatan kode

1. Simbol Assembler Khusus

Assembler telah menyediakan beberapa simbol untuk menunjukkan *register* tertentu sebagai *operand*. Tabel 2.11 dibawah ini menunjukkan simbol *assembler* khusus.

Simbol Khusus	Arti
A	Akumulator
R0..R7	*Register* serbaguna
DPTR	*Data pointer. Register*16 bit
PC	*Program Counter*. *Register* 16 bit yang berisi alamat instruksi berikutnya yang akan dijalankan
C	*Carry Flag*
AB	Akumulator/*register* B. Pasangan *register* untuk perkalian dan pembagian

Tabel 2.11. Simbol *Assembler* Khusus

2. Pengalamatan Tak Langsung

Operand pengalamatan tak langsung menunjuk ke sebuah *register* yang berisi lokasi alamat memori yang akan digunakan dalam operasi. Lokasi yang nyata tergantung pada isi *register* saat instruksi dijalankan. Untuk melaksanakan pengalamatan tak langsung digunakan simbol @.

Contoh :

ADD A,@R0 ; Tambahkan isi RAM yang lokasinya ditunjukkan oleh *register*R0 ke akumulator

3. Pengalamatan Langsung

Pengalamatan langsung dilakukan dengan memberikan nilai ke suatu *register* secara langsung. Untuk melaksanakan hal tersebut digunakan tanda #.

Contoh :

MOV A,#01H ; Isi akumulator dengan bilangan 01H

Pengalamatan data langsung dari 0 sampai 127 akan mengakses RAM internal, sedang pengalamatan dari 128 sampai 255 akan mengakses *register* perangkat keras.

Contoh :

MOV P2,A ; Pindahkan isi akumulator ke alamat data B0H (B0H adalah alamat *port* 3)

4. Pengalamatan Bit

Pengalamatan bit adalah penunjukkan alamat lokasi bit baik dalam RAM internal (*byte* 32 sampai 47) atau bit perangkat keras. Untuk melakukan pengalamatan bit digunakan simbol titik (.).

Contoh :

- FLAG.3,40.5,21H.5

- ACC.7

5. Pengalamatan Kode

Ada tiga macam instruksi yang dibutuhkan dalam pengalamatan kode, yaitu *Relative Jump, In-block Jump* atau *Call,* dan *Long Jump* atau *Call.*

5.1. Operator *Assembler*

Ada empat belas operator dalam *assembler* yang meliputi aritmatika, logika, operator khusus, dan operator hubungan (relasional).

5.1.1. Operator Aritmatika

+ plus/tambah

- minus/kurang

*	perkalian
/	pembagian *integer*
MOD	pembagian modular

5.1.2. Operator Logika

OR	16 bit OR
AND	16 bit AND
XOR	16 bit *exclusive* OR
NOT	16 bit komplemen

5.1.3. Operator Khusus

SHR	16 bit geser kanan
SHL	16 bit geser kiri
HIGH	pilih bagian atas bit
LOW	pilih bagian bawah bit
()	operator dalam kurung didahulukan

5.1.4. Operator Hubungan

EQ	=	sama dengan
NE	<>	tidak sama dengan

LT	<	lebih kecil
LE	<=	lebih kecil atau sama dengan
GT	>	lebih besar
GE	>=	lebih besar atau sama dengan

Urutan operator yang harus didahulukan berturut-turut adalah :

Tanda kurung ( )

HIGH,LOW

*,/,MOD,SHL,SHR

+,-

EQ,NE,LT,LE,GT,GE,=,<>,<,<=,>,>=

NOT

AND

OR,XOR

2.11.2. Perangkat Instruksi

Mikrokontroler Intel 8031 memiliki 256 perangkat instruksi. Seluruh instruksi dapat dikelompokkan dalam 4 bagian yang meliputi instruksi 1 *byte* sampai 4 *byte*.

Apabila frekuensi *clock* mikrokontroler yang digunakan adalah 12 MHz, kecepatan pelaksanaan instruksi akan bervariasi dari 1 sampai 4 mikrodetik. Perangkat instruksi mikrokontroler Intel 8031 dapat dibagi menjadi lima kelompok sebagai berikut :

a. Instruksi *Transfer* Data

Instruksi ini memindahkan data antara *register-register*, memori-memori, *register*-memori, antar muka-*register*, dan antar muka-memori.

b. Instruksi Aritmatika

Instruksi ini melaksanakan operasi aritmatika yang meliputi penjumlahan, pengurangan, penambahan satu (inkremen), pengurangan satu (dekremen), perkalian, dan pembagian.

c. Instruksi Logika dan Manipulasi Bit

Melaksanakan operasi logika AND,OR,XOR, perbandingan, pergeseran, dan komplemen data.

d. Instruksi Percabangan

Instruksi ini mengubah urutan normal pelaksanaan suatu program. Dengan instruksi ini program yang sedang dilaksanakan akan mencabang ke suatu alamat tertentu. Instruksi percabangan dibedakan atas percabangan bersyarat dan percabangan tanpa syarat.

e. Instruksi *Stack,I/O*, dan Kontrol

Instruksi ini mengatur penggunaan *stack*, membaca.menulis *port* I/O, serta pengontrolan-pengontrolan.