PERENCANAAN DAN PEMBUATAN PENGATUR KECEPATAN MODEM OTOMATIS

TUGAS AKHIR

Disusun oleh :

DOMINIKUS RIDWAN

NRP : 2292.100.059

JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA

1997

PERENCANAAN DAN PEMBUATAN PENGATUR KECEPATAN MODEM OTOMATIS

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik Elektro
Pada
Bidang Studi Telekomunikasi
Jurusan Teknik Elektro
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
S u r a b a y a

Mengetahui / Menyetujui

Dosen Pembimbing I	Dosen Pembimbing II
Dr. Ir. MOCH. SALEHUDIN, MEng.Sc	Ir. HANNY BUDINUGROHO
NIP. 130 532 026	NIP. 131 651 433

S U R A B A Y A
Maret, 1997

ABSTRAK

Saat ini perkembangan teknologi modem berlangsung cepat. Banyak modem-modem baru muncul dengan menawarkan teknologi yang lebih baik, seperti kelajuan transfer data yang lebih cepat, tingkat kompresi data yang lebih baik, serta kemampuan modem untuk mengirim data pada kelajuan optimum. Hal ini menyebabkan modem-modem terdahulu tergeser kedudukannya oleh modem-modem baru. Padahal teknologinya belum tentu tertinggal, seperti kelajuan transfer data yang cukup tinggi, namun tidak dapat menyesuaikan kelajuan transfer data pada kelajuan optimum. Dan modem-modem ini masih cukup banyak di pasaran.

Dalam tugas akhir ini akan dibuat suatu program pengatur kelajuan transfer data otomatis dengan memanfaatkan modem tersebut. Pembuatan program ini dibatasi pada jenis protokol yang digunakan, saluran transmisi yang digunakan (saluran RS-232 dan saluran telepon), dan jenis bahasa pemrograman (C^{++}). Dalam perencanaan dan pembuatan program ini, parameter yang digunakan untuk mengontrol kelajuan transfer data adalah *checksum error checking*. Apabila terjadi kesalahan penerimaan data secara berurutan maka kelajuan transfer data diturunkan, sedangkan apabila data diterima secara benar dan berurutan maka kelajuan transfer data dinaikkan hingga kelajuan maksimum. Dengan demikian data dapat dikirim dengan kelajuan optimum. Dalam pembuatan program ini, terdapat masalah yang berhubungan dengan pemrograman modem. Masalah tersebut yaitu adanya konflik antara protokol modem dan protokol yang digunakan dalam membuat program ini. Jadi dalam tugas akhir ini program yang dibuat hanya dapat dijalankan dengan menggunakan saluran RS-232. Untuk melakukan uji coba terhadap program, saluran RS-232 diinjeksi dengan *noise* eksternal (*white noise generator*) yang dapat diubah-ubah level *noise*-nya. Dari uji coba yang dilakukan, terdapat level *noise* kritis, yang menyebabkan komunikasi data antar dua DTE tidak dapat berlangsung. Apabila level *noise* yang diinjeksikan lebih kecil dari level *noise* kritis maka komunikasi data dapat berlangsung pada kelajuan tertentu, sedangkan apabila level *noise* yang diinjeksikan lebih besar dari level *noise* kritis maka komunikasi data tidak dapat berlangsung. Dari pengukuran yang dilakukan, diperoleh harga S/N_{kritis} sebesar 13,98 dB.

Dari pengerjaan tugas akhir ini dapat disimpulkan bahwa kelajuan transfer data dapat dibuat optimum dengan melakukan pengontrolan terhadap *error checking* yang diterima, dalam tugas akhir ini digunakan metode *checksum error checking*.

KATA PENGANTAR

Pertama-tama penyusun mengucapkan puji syukur kepada Tuhan Yang Maha Esa karena berkat rahmat dan karunia-Nya penyusun dapat menyelesaikan Laporan Tugas Akhir dengan judul:

" Perencanaan dan Pembuatan Pengatur Kecepatan Modem Otomatis"

Laporan Tugas Akhir ini disusun guna memenuhi persyaratan dalam memperoleh gelar Sarjana Teknik Elektro di Jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember (ITS) Surabaya. Penyusun menyadari bahwa Laporan Tugas Akhir ini masih memiliki banyak kekurangan meskipun penyusun telah berusaha menyelesaikan dengan sebaik-baiknya. Oleh karena itu penyusun bersedia menerima kritik dan saran yang membangun dari pihak-pihak yang berkepentingan.

Dengan adanya Laporan Tugas Akhir ini, penyusun berharap dapat menambah pengetahuan, wawasan, dan berguna bagi pihak yang membutuhkan.

Surabaya, Maret 1997

Penyusun

UCAPAN TERIMA KASIH

Selesainya penyusunan Laporan Tugas Akhir ini berkat bantuan dari berbagai pihak. Untuk itu penyusun mengucapkan banyak terima kasih, antara lain kepada :

1. Bapak Dr. Ir. Moch. Salehudin, MEng.Sc, dosen pembimbing tugas akhir;
2. Bapak Ir. Hanny Budinugroho, dosen pembimbing tugas akhir dan dosen wali;
3. Bapak Ir. M. Aries Purnomo, Koordinator Bidang Studi Teknik Telekomunikasi;
4. Bapak Ir. Teguh Yuwono, Ketua Jurusan Teknik Elektro;
5. Papa, Mama, dan Kakak yang telah memberi semangat dan dorongan sehingga penyusun dapat menyelesaikan Tugas Akhir ini;
6. Teman-teman E-32, Ricky, Bobby, Khamid, Nengah, Toni,dll;
7. Semua pihak yang telah membantu dan memberikan dorongan kepada penyusun selama pembuatan Tugas Akhir ini.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

I.1 LATAR BELAKANG

Pada saat ini sistem komunikasi data semakin penting peranannya. Banyak aspek kehidupan modern yang akan terhenti tanpa adanya komunikasi data. Komunikasi data dapat terjadi antara unit-unit peralatan penyusun di dalam komputer, antara dua buah komputer atau banyak komputer dalam suatu jaringan. Komunikasi data ini sangat berhubungan dalam pengiriman atau penerimaan informasi. Saat ini peranan informasi sangatlah penting, sehingga semakin diperlukan teknologi komunikasi yang mampu mengirimkan atau menerima data secara cepat dan tepat.

Saat ini perkembangan teknologi komunikasi berlangsung sangat cepat, di antaranya perkembangan teknologi modem. Banyak modem baru muncul dengan menawarkan teknologi yang lebih baik, seperti kelajuan transfer data yang lebih tinggi, tingkat kompresi data yang lebih tinggi, serta kemampuan untuk mengirim data dengan kelajuan optimum. Hal ini menyebabkan modem-modem yang terdahulu tergeser kedudukannya oleh modem yang lebih baru. Padahal modem-modem lama belum tentu tertinggal teknologinya seperti kelajuan transfer datanya yang cukup tinggi, namun tidak dapat menyesuaikan kelajuan transfer data pada kelajuan optimum, dan modem jenis ini masih cukup banyak di pasaran.

I.2 PERMASALAHAN DAN BATASAN MASALAH

Komunikasi data antar komputer saat ini pada umumnya menggunakan jaringan telepon. Untuk memperoleh kelajuan transfer data yang tinggi diperlukan modem yang memiliki kelajuan transfer data yang tinggi pula. Namun kelajuan transfer data modem yang tinggi belum menjamin bahwa kelajuan transfer data antar komputer berlangsung dengan kelajuan yang tinggi. Hal ini disebabkan karena keadaan jaringan telepon yang dipakai dan keadaan trafik saluran saat terjadi komunikasi data. Keadaan trafik saluran ini sangat berpengaruh pada kelajuan transfer data, bila kelajuan transfer data terlalu tinggi akan mengakibatkan terjadinya kesalahan (*error*) pada data-data yang dikirimkan. Oleh karena itu kelajuan transfer data harus diubah-ubah untuk disesuaikan dengan keadaan trafik saluran yang digunakan.

Saat ini telah ada modem yang mampu mengatur kelajuan transfer datanya secara otomatis, namun harganya cukup tinggi. Sementara itu masih terdapat modem dengan harga yang relatif murah dengan kelajuan transfer data yang cukup tinggi, tetapi modem ini tidak dapat menyesuaikan kelajuan transfer datanya dengan keadaan trafik saluran telepon yang digunakan. Dalam tugas akhir ini akan dibuat suatu program pengatur kelajuan transfer data secara otomatis untuk modem-modem tersebut.

Pembuatan program ini dibatasi pada saluran transmisi yang digunakan, yaitu saluran telepon digital umum. Selain itu juga dibatasi pada jenis

protokol transfer data dan bahasa pemrograman yang digunakan, dalam hal ini bahasa pemrograman C++.

I.3. TUJUAN

Menghasilkan perangkat lunak atau program yang mampu menyesuaikan kelajuan transfer data atau kecepatan modem agar komunikasi data dapat dilakukan dengan kelajuan optimum.

I.4 METODOLOGI

Metodologi yang digunakan dalam menyelesaikan tugas akhir ini adalah:

- studi literatur tentang protokol transfer data seperti protokol XModem. Selain itu studi literatur tentang komunikasi serial dan modem serta cara-cara pemrogramannya.
- perencanaan pembuatan program transfer data antara dua buah komputer melalui saluran RS-232, dilanjutkan dengan perencanaan pengembangan program tersebut dengan menggunakan modem dan saluran telepon.
- pembuatan program transfer data antara dua buah komputer baik melalui saluran RS-232, maupun melalui saluran telepon, dengan menggunakan bahasa pemrograman C++.
- pengujian program yang telah dibuat sebelum mengambil kesimpulan.

I.5 SISTEMATIKA

Pembahasan pada laporan tugas akhir ini dibagi menjadi lima bab yang merupakan suatu rangkaian yang saling berhubungan. Bab I merupakan bab pendahuluan yang menjelaskan garis besar tentang tugas akhir yang telah dilakukan. Bab berikutnya yaitu bab II, III, dan IV merupakan isi dari laporan tugas akhir ini. Teori dasar sistem komunikasi serial dan dasar pemrogramannya dituliskan pada bab II. Bab III berisi tentang perencanaan dan pembuatan program sesuai dengan tujuan tugas akhir, sedangkan pada bab IV diberikan ulasan tentang hasil uji coba program. Bab terakhir yaitu bab V merupakan bab penutup yang berisi kesimpulan dan saran dari tugas akhir yang telah dilakukan.

I.6 RELEVANSI

Dengan adanya pengatur kecepatan modem secara otomatis diharapkan transfer data dapat berlansung dengan kelajuan optimum.

BAB II
SISTEM KOMUNIKASI DATA

II.1 UMUM

Pada saat ini sistem komunikasi data semakin penting peranannya. Banyak aspek kehidupan modern yang akan terhenti tanpa adanya komunikasi data. Komunikasi data dapat terjadi antar 2 buah komputer atau antar banyak komputer dalam suatu jaringan komunikasi. Contoh komunikasi data tersebut antara lain: komunikasi data antar mesin fax, komunikasi data pada jaringan ATM, atau komunikasi data pada jaringan lokal (*Local Area Network*).

Ditinjau dari cara pengiriman data (pengiriman karakter), komunikasi data dapat dibedakan menjadi dua yaitu komunikasi data paralel dan komunikasi data serial. Pada komunikasi data paralel sinyal bit-bit data dikirimkan melalui kanal-kanal yang berbeda secara sekaligus, sehingga kanal yang diperlukan harus sebanyak jumlah bit data yang dikirimkan. Sedangkan pada komunikasi data serial sinyal bit-bit data dikirimkan secara bergantian pada suatu kanal.

Masing-masing metode pengiriman ini memiliki kelebihan dan kekurangan. Komunikasi paralel mempunyai kelebihan dalam hal kecepatan pengiriman, tetapi tidak dapat digunakan untuk jarak jauh, bahkan jarak pengirimannya relatif pendek, biasanya sepuluh kaki atau kurang. Komunikasi paralel pada umumnya dipakai untuk komunikasi data antar unit peralatan dalam

komputer itu sendiri, seperti unit pengolahan pusat (CPU) dengan monitor. Sedangkan komunikasi serial umum digunakan untuk komunikasi data yang jaraknya lebih jauh, seperti komunikasi data antara dua komputer dengan menggunakan modem.

Sehubungan dengan tugas akhir ini, akan dibahas lebih lanjut komponen-komponen dasar yang berkaitan dengan sistem komunikasi serial, yaitu saluran RS-232, UART (*Universal Asynchronous Receiver Transmitter*), modem (*Modulator Demodulator*), dan protokol transfer data (*File Transfer Protocol*).

II.2 STANDAR INTERFACE RS-232

Interface kalau diterjemahkan dapat berarti penghubung, yaitu sebagai penghubung antar dua lingkungan yang berbeda. Dalam hal ini dua lingkungan yang dimaksud adalah peralatan komputer dan peralatan di luar komputer seperti modem.

RS-232 merupakan seperangkat alat yang berfungsi sebagai *interface* dalam proses transfer data antar komputer dalam bentuk komunikasi serial. RS-232 merupakan kependekan dari *Recommended Standard Number 232*. Alat ini dibuat oleh *Engineering Department of the Electronic Industries Association* (EIA). Nama resmi dari RS-232 adalah *Interface between Data Terminal Equipment and Data Communication Equipment Employing Serial Binary Data Interchange*. Yang dimaksud dengan *Data Terminal Equipment* (DTE) dan *Data Communication Equipment* (DCE) di sini adalah peralatan komputer dan modem.

Standar *interface* ini direvisi secara periodik. Sampai saat ini sudah terdapat lima revisi. Revisi A-C berisi tentang *editorial in nature*, revisi D tentang *physical connector*, dan revisi E membahas tentang *error-correcting modem*.

Pada dokumen yang dibuat oleh *Electronic Industries Association* (EIA) dijelaskan tentang tiga macam aspek yang berbeda dalam hubungan *Data Terminal Equipment* (DTE) dan *Data Communication Equipment* (DCE), yaitu:

- Deskripsi fungsi formal dari rangkaian pertukaran (*Formal functional descriptions of interchange circuits*)
- Karakteristik sinyal listrik (*Electrical signal characteristic*)
- Deskripsi mekanis dari rangkaian antar muka termasuk penghubung (*Mechanical description of interface circuit including connectors*)

Tabel 2.1 berikut ini akan memberikan rangkuman tentang standar *interface* RS-232. Dari tabel ini dapat diperoleh informasi sebagai berikut:

- Fungsi standar dari pin untuk EIA/TIA 232-E, 561 (8 pin), dan 574 (9 pin).
- Nama EIA/TIA untuk rangkaian, yaitu AA, BB, dan lain-lain.
- Nomer rangkaian CCITT, yaitu 101, 102, dan lain-lain.
- Arah dari sinyal kontrol, yaitu ke arah DTE atau DCE.
- Kependekan umum untuk rangkaian, yaitu TD, RD, DSR, dan lain-lain.
- Deskripsi resmi EIA/TIA dari rangkaian, yaitu *Transmit Data*, *Clear to Send*, dan lain-lain.

TABEL 2.1[1)]
RANGKAIAN RS-232

25 Pin	9 Pin	8 Pin	Nama EIA/TIA	CCITT V.26	Arah Sinyal	Mnemonic	Deskripsi
1	5	-	n/a	n/a	n/a	n/a	Protective Ground
2	3	6	BA	103	ke DCE	TD	Transmitted Data
3	2	5	BB	104	ke DTE	RD	Received Data
4	7	8	CA/CI	105/133	ke DCE	RTS/RTR	Request to Send / Ready to Receive
5	8	7	CB	106	ke DTE	CTS	Clear to Send
6	6	-	CC	107	ke DTE	DSR	Data Set Ready (DCE Ready)
7	-	4	AB	102	-	-	Signal Ground or Common
8	1	2	CF	109	ke DTE	DCD	Data Carrier Detect
9	-	-	-	-	-	-	Reserved for Testing
10	-	-	-	-	-	-	Reserved for Testing
11	-	-	-	126	-	-	Reserved for Testing
12	-	-	SCF	122	ke DTE	-	Secondary Received Line Signal Detect
13	-	-	SCB	121	ke DTE	-	Secondary Clear to Send
14	-	-	SBA	118	ke DCE	-	Secondary Transmitted Data
15	-	-	DB	114	ke DTE	-	Transmission Signal Element Timing
16	-	-	SBB	119	ke DTE	-	Secondary Received Data
17	-	-	DD	115	ke DTE	-	Receiver Signal Element Timing
18	-	-	LL	141	ke DCE	-	Local Loopback
19	-	-	SCA	120	ke DCE	-	Secondary Request to Send
20	4	3	CD	108/1,/2	ke DCE	DTR	Data Terminal Ready (DTE ready)
21	-	-	RL/CG	140/110	ke CDE	-	Remote Loopback / Signal Quality Detect
22	-	1	CE	125	ke DTE	RI	Ring Indicator
23	-	-	CH/CI	111/112	keduanya	DRI	Data Signal Rate Detector
24	-	-	DA	113	ke DCE	-	Transmit Signal Element Timing
25	-	-	TM	142	ke DTE	-	Test mode

Bagian dari tabel yang diarsir akan digunakan dalam pembahasan karena berhubungan dengan komunikasi asinkron, sedangkan bagian lainnya digunakan pada komunikasi sinkron.

Fungsi RS-232 dapat dibagi menjadi dua bagian utama, yaitu fungsi data dan fungsi kontrol. Yang termasuk fungsi data yaitu pin 2 sebagai *transmitter* dan pin 3 sebagai *receiver*. Sedangkan pin-pin yang lain termasuk fungsi kontrol,

---

1) Joe Campbell, *C Programmer's Guide to Serial Communication*, Second Edition, Prentice Hall, Indianapolis, 1994, p. 160

dinamakan demikian karena pin-pin tersebut memberikan status atau perintah dalam pengontrolan modem.

II.2.1 Deskripsi Fungsi Formal dari Rangkaian Pertukaran

Berikut ini akan diuraikan fungsi dari masing-masing pin pada RS-232 yang digunakan pada komunikasi asinkron.

PROTECTIVE GROUND (PIN 1)

Meskipun bukan merupakan rangkaian yang didefinisikan secara resmi, RS-232 menyarankan pin ini digunakan pada rangkaian dan dihubungkan dengan kerangka dari peralatan, dan oleh sebab itu *earth ground* disediakan dari tegangan AC *power supply*.

SIGNAL COMMON (PIN 7)

Pin ini adalah *common* dari semua sinyal dari rangkaian dan harus ada di setiap *interface*. Hubungan dari pin ini dan *protective ground* (pin 1) dapat mencegah kerusakan fatal dari peralatan.

TRANSMITTED DATA (TD, PIN 2, KE DCE)

Pin ini berfungsi sebagai saluran pengiriman data dari DTE. Saluran TD ini membawa sinyal data serial dari DTE (UART) ke DCE (Modem). Sesuai dengan perkembangan teknologi, pemancar ditandai dengan MARK selama

periode saluran *idle* (tidak sibuk). Kemampuan DTE ditentukan oleh *Request to Send, Clear to Send, Data Set Ready*, dan *Data Terminal Ready*.

RECEIVED DATA (RD, PIN 3, KE DTE)

Pin ini berfungsi sebagai saluran penerimaan data pada DTE. Unjuk kerja dari RD tidak tergantung pada fungsi RS-232 yang lainnya. Standar unjuk kerja ini ditentukan pada kemampuan RD berada pada keadaan MARK ketika saluran tidak terdapat *carrier*.

REQUEST TO SEND (RTS, PIN 4, KE DCE)

CLEAR TO SEND (CTS, PIN 5, KE DTE)

Standar RS-232 menyatakan bahwa *Request to Send* mengondisikan modem untuk pengiriman. Pada kenyataan, fungsi pin ini hanya mengatur modem *half duplex* untuk berada pada mode pengiriman atau mode penerimaan. Selama modem *half duplex* sedang dalam mode penerimaan, DTE akan menahan pengiriman *Request to Send*. Ketika DTE berubah menjadi mode pengiriman, maka DTE akan menyatakan keinginan untuk mengirim data ke modem *half duplex* dengan mengirimkan *Request to Send*. DTE tidak dapat memulai mengirim data dengan segera karena modem tidak dapat berubah secara tiba-tiba. Setelah mengirimkan *Request to Send*, DTE mulai memonitor sinyal *Clear to Send* dari modem, dimana sinyal ini dinyatakan dengan sinyal *low* saat modem dalam mode penerimaan. Setelah modem selesai menerima data, modem mengirimkan sinyal

Clear to Send ke DTE untuk menginformasikan bahwa saat ini bisa dilakukan pengiriman data dengan aman. *RTS/CTS handshaking* ini ditujukan dalam perubahan mode pengiriman ke mode penerimaan dan sebaliknya.

Pada hubungan *full duplex*, *RTS/CTS handshaking* ini tidak diperlukan. Hal ini disebabkan pada hubungan *full duplex* terdapat dua buah saluran.

DATA SET READY (DSR, PIN 6, KE DTE)

Data Set Ready dikirimkan bila keadaan berikut ini terjadi secara bersamaan. Keadaan yang dimaksud yaitu :

- Modem dalam keadaan terhubung dengan saluran transmisi dan dalam keadaan *off hook*, tetapi tidak berada dalam mode tes, suara, dan dial.
- Modem harus telah melakukan proses dial, proses pengawasan terhadap panggilan yang masuk, dan segala sesuatu yang diperlukan dalam pelayanan panggilan melalui jaringan telepon.
- Modem telah mulai pengiriman *discrete answer tone*. *Answer tone* dan *Data Set Ready* dikirimkan dua detik setelah telepon dalam keadaan *off-hook*.

Model *Data Set* adalah nama untuk modem dari perusahaan telepon. Standar EIA menyebutnya DCE.

DATA CARRIER DETECT (DCD, PIN 8, KE DTE)

Pin ini, yang memiliki nama resmi *Received Line Signal Detect*, mengirimkan sinyal ketika modem menerima *remote carrier* dan siap melakukan pertukaran data dengan *host*. DCD tetap dinyatakan selama hubungan. Pada modem *half duplex*, DCD hanya dikirimkan oleh modem penerima.

DATA TERMINAL READY (DTR, PIN 20, KE DCE)

Pada mode original, *Data Terminal Ready* harus dikirimkan supaya terjadi *auto-dial* (pada *answer mode*), dan juga supaya terjadi *auto-answer*. Begitu modem terhubung pada saluran transmisi, *Data Terminal Ready* harus tetap dinyatakan untuk mempertahankan hubungan ini. Jika tidak akan terjadi pemutusan hubungan dari saluran transmisi.

RING INDICATOR (RI, PIN 22, KE DTE)

Pin ini mengirimkan sinyal ke DTE selama terdapat ring dalam saluran transmisi. Jadi *Ring Indicator* dapat dianggap sebagai fasilitas penjawab otomatis. Sinyal yang dikirim ke DTE sesuai dengan nada yang terdapat pada saluran.

II.2.2 Karakteristik Sinyal Listrik

Meskipun karakteristik sinyal listrik dari *interface* RS-232 tidak relevan dengan *programmer*, beberapa pengetahuan tentang karakteristik ini perlu diketahui untuk memahami topik dalam tugas akhir ini.

SPEED and POWER

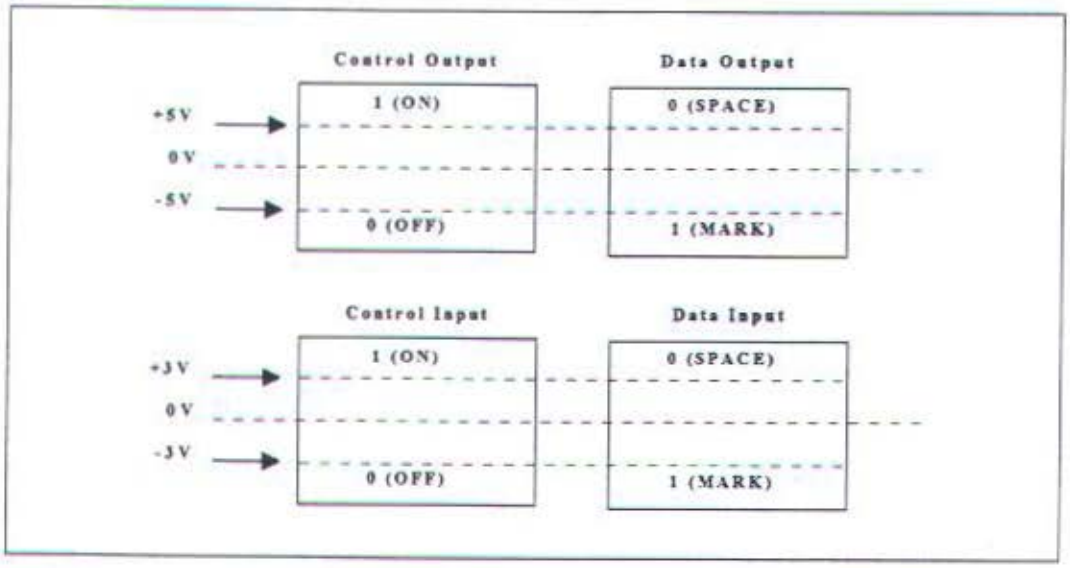
Standar RS-232 menyatakan bahwa kelajuan transfer data yang diijinkan antara nol dan 20000 bit per detik (bps). Namun pada kenyataannya kelajuan transfer data dibatasi hingga 19200 bps. Untuk EIA 561 (8 pin) kelajuan transfer datanya dapat mencapai 38400 bps. Selain itu panjang kabel yang digunakan harus kurang dari *50 feet* dan total kapasitansi kabel harus lebih kecil dari 2500 pikofarad. Sedangkan arus yang digunakan tidak boleh melebihi 0,5 ampere.

LOGIC LEVELS

Standar RS-232 menetapkan *bipolar logic level*. Oleh karena itu, level logika tidak hanya dinyatakan oleh besar tegangan tetapi juga oleh polaritasnya. Tegangan maksimum yang diijinkan yaitu ±15 V.

Standar RS-232 menetapkan empat level logika. Level logika input mempunyai definisi yang berbeda dengan level logika output. Gambar 2.1 menunjukkan definisi level logika dari RS-232 untuk bagian input dan output.

Level logika biner untuk output adalah +5 V hingga +15 V dan -5 V hingga -15 V. Tegangan antara -5 V dan +5 V tidak didefinisikan.Sedangkan level logika biner untuk input adalah +3 V hingga +15 V dan -3 V hingga -15 V. Tegangan antara -3 V dan +3 V tidak didefinisikan.

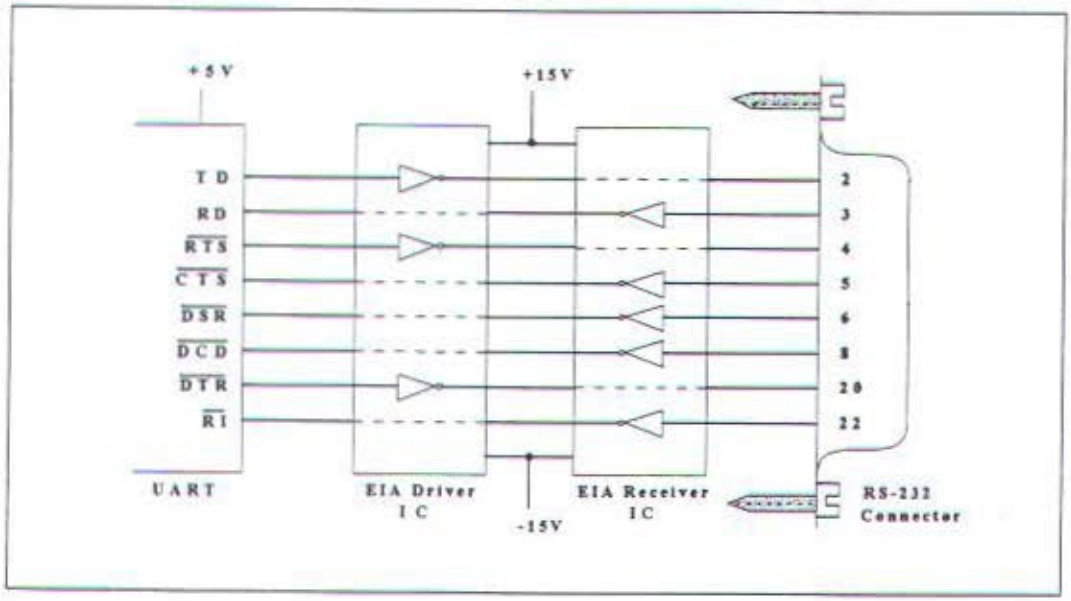


GAMBAR 2.1[2)]
LEVEL LOGIKA RS-232 UNTUK FUNGSI KONTROL

RS-232 LEVEL CONVERSION

Karena tegangan RS-232 dan level logikanya tidak sesuai dengan tegangan maupun level logika yang digunakan pada rangkaian komputer, maka diperlukan pengubahan level-level tersebut. Hal ini dapat dilakukan oleh IC khusus yang disebut *EIA (RS-232) line drivers* dan *line receivers*. Untuk alasan elektronik, peralatan ini biasanya berupa *inverter*, yang menunjukkan input atau output dari *asynchronous I/O controller IC* (UART). Gambar 2.2 menunjukkan pengubahan level logika secara perangkat keras.

2) Ibid., p.164



GAMBAR 2.2[3)]
PENGUBAHAN LEVEL LOGIKA RS-232 SECARA PERANGKAT KERAS

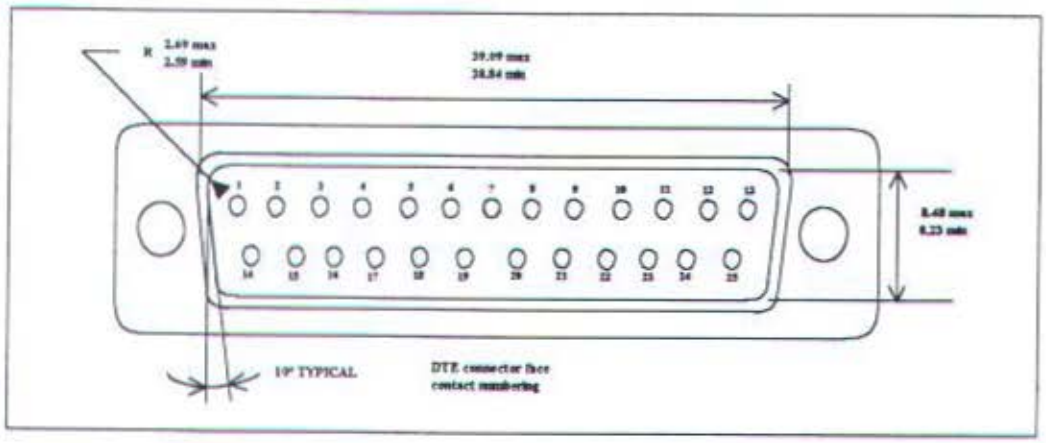
II.2.3 Deskripsi Mekanis dari Rangkaian Antar Muka

Konektor standar, yang memenuhi spesifikasi sesuai revisi D, adalah DB-25, yang kadang-kadang dikenal dengan konektor RS-232. Standar RS-232 menetapkan *female connector* untuk DCE dan *male connector* untuk DTE. Hal ini memberikan garis pedoman tentang dimana konektor harus ditempatkan. Gambar 2.3 merupakan gambar dari konektor EIA/TIA 232-E dengan 25 pin.

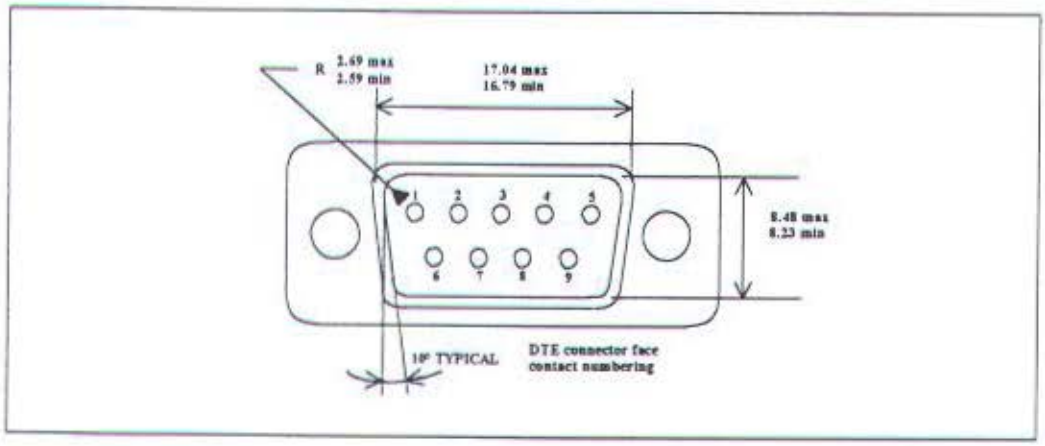
IBM PC AT memutuskan transisi DB-25 dengan menggunakan konektor 9 pin. Gambar 2.4 merupakan gambar dari konektor EIA/TIA 574 dengan 9 pin. Sedangkan untuk kebutuhan konektor yang lebih kecil diproduksi konektor EIA/TIA 561 dengan 8 pin, ditunjukan dengan gambar 2.5.

---

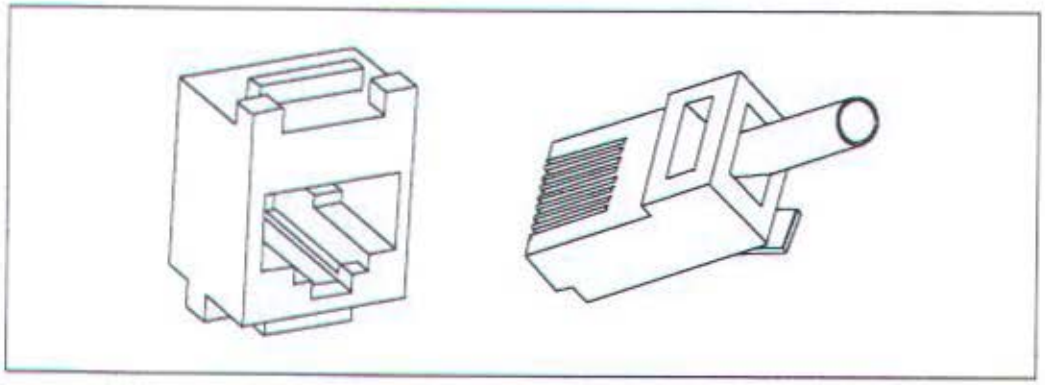
3) Ibid., p. 165



GAMBAR 2.3[4)]
EIA/TIA 232-E DENGAN 25 PIN



GAMBAR 2.4[5)]
EIA/TIA 574 DENGAN 9 PIN



GAMBAR 2.5[6)]
EIA/TIA 561 DENGAN 8 PIN

4) Ibid, p.166
5) Ibid, p.166
6) Ibid, p.166

II.3 UART

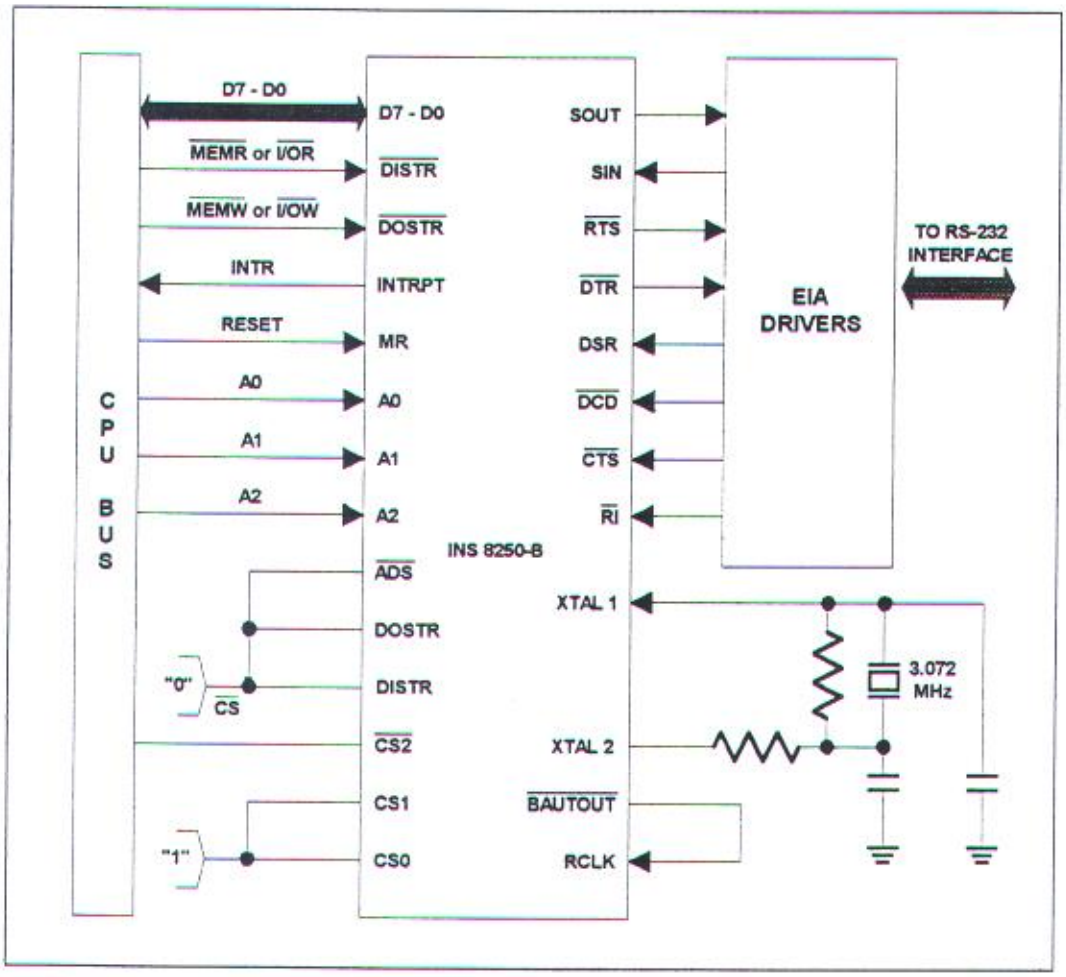
UART merupakan kependekan dari *Universal Asynchronous Receiver Transmitter*. UART adalah suatu alat yang berfungsi membantu kerja pemrogram maupun *processor* dalam melakukan proses yang berhubungan dengan *asynchronous serial I/O*. Untuk menerima atau mengirimkan data, program cukup membaca atau menulis data (byte) ke UART, yang tampak oleh *processor* seperti salah satu lokasi memori atau *port* masukan/keluaran. Bentuk nyata UART berupa *integrated circuit*. Ada beberapa contoh UART, antara lain keluarga IC National 8250 dan Zilog Z80SIO. Namun yang akan digunakan dalam pembahasan adalah keluarga IC National 8250.

II.3.1 Perangkat Keras UART 8250

Dari gambar 2.6 tampak bahwa UART 8250 mempunyai tiga *interface* dasar, yaitu: *I/O bus*, *clock* UART, dan *interface* ke RS-232. UART 8250 berhubungan dengan 8 bit data dari CPU melalui *data bus*. Pada bagian ini data dapat masuk (*write*) dan keluar (*read*) dari UART. Proses pembacaan dan penulisan data dibedakan melalui status dari pin DISTR dan DOSTR. Selain itu, UART 8250 terdiri dari beberapa register internal, dimana alamat dari register-register tersebut ditentukan dengan status pada pin A0 - A2.

Berikut ini proses-proses yang dilakukan dalam mengirimkan sebuah byte data :

- CPU menempatkan byte data yang akan dikirim pada *data bus* (D0 - D7).



GAMBAR 2.6[7]
KONFIGURASI DASAR DARI NATIONAL 8250

- CPU mengeset alamat *transmitter buffer register* pada *line* A0 - A2.
- CPU mengeset *line* DISTR dan DOSTR sedemikian sehingga data dari D0 - D7 masuk ke *transmitter buffer*. Kemudian UART 8250 memindahkan byte data dari *transmitter buffer register* ke *transmitter shift register*, saat register ini kosong.

[7] Ibid, p. 213

Untuk menerima sebuah byte data, proses-proses yang dilakukan serupa dengan proses pengiriman data. Proses-proses berikut diasumsikan bahwa sebuah byte data yang akan diterima telah berada di *receiver buffer register*.

- CPU mengeset alamat *receiver buffer register* pada *line* A0 - A2.
- Operasi *read* dilakukan dengan mengeset *line* DISTR dan DOSTR.
- Data kemudian pindah dari *receiver buffer* ke *data bus* (D0 -D7) sehingga CPU dapat memprosesnya.

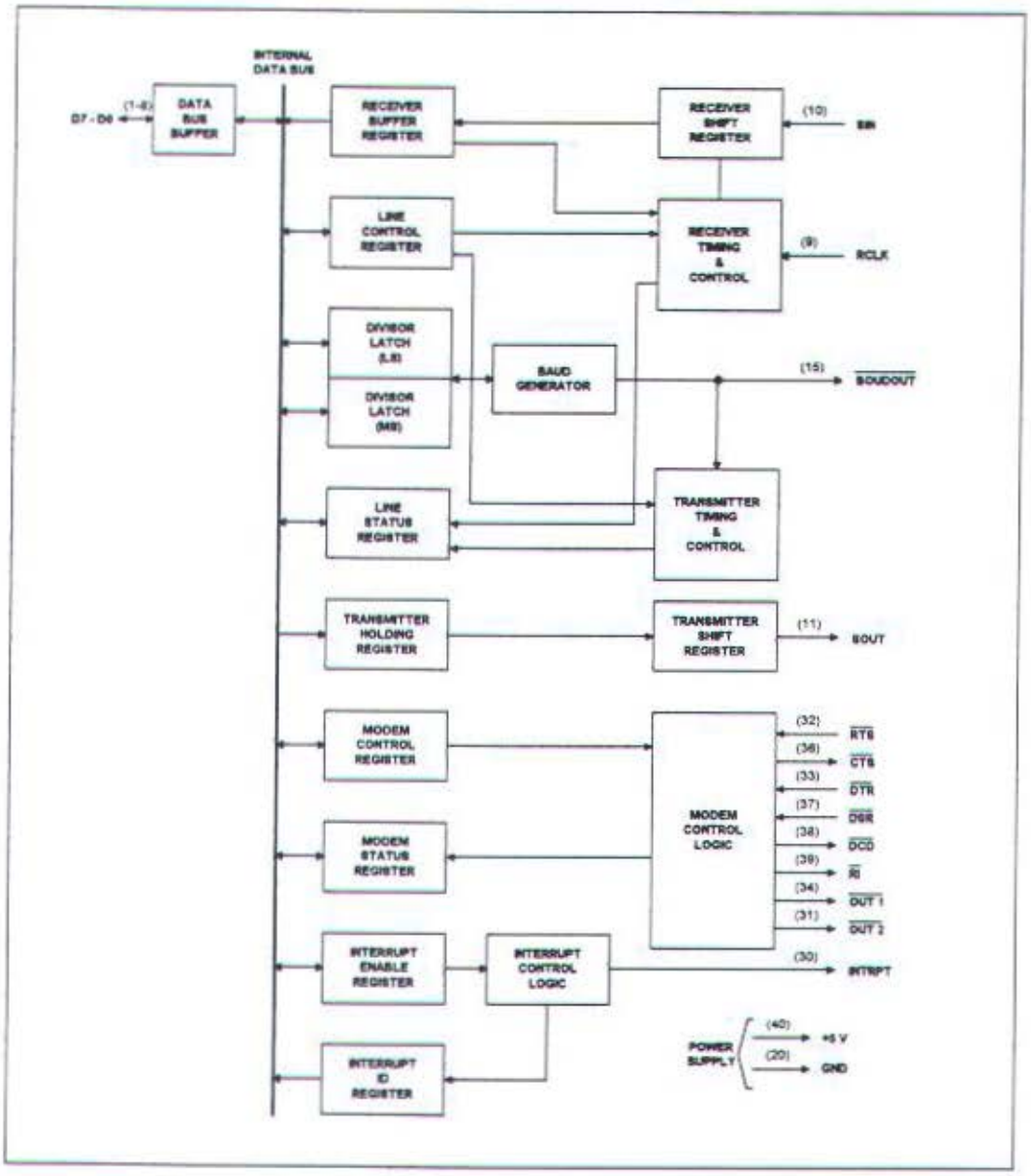
Hal terakhir dari *interface* dengan *I/O bus* adalah *interrupt* (INTRPT). Output INTRPT akan *TRUE* ketika suatu keadaan terjadi dalam UART 8250 sesuai dengan program yang dikehendaki.

Sistem berikutnya yang berhubungan dengan UART adalah sistem *clock*. Sinyal *clock* UART 8250 dapat diperoleh secara external maupun secara internal dengan menggunakan kristal. Dalam contoh di sini sinyal *clock* UART 8250 diperoleh dengan menggunakan kristal. Kristal ini dihubungan dengan input XTAL1. Sebelum dihubungkan dengan input XTAL1, sinyal dari kristal harus dilewatkan rangkaian pembagi, yang dapat diprogram, untuk menghasilkan *master data clock*. *Master data clock* ini 16 kali lebih tinggi dari *baud rate* yang diinginkan. *Master data clock* ini kemudian dikeluarkan kembali melalui pin BAUDOUT. Jika pin BOUDOUT dan RCLK dihubungkan maka proses pengiriman dan penerimaan data akan dilakukan pada *baud rate* yang sama.

II.3.2 Pemrograman UART 8250

UART 8250 memiliki beberapa register yang dapat digunakan untuk mengontrol ataupun mengetahui kondisi dari UART. Masing-masing register memiliki fungsi dan alamat tersendiri. Jadi dengan menulis (*write*) atau membaca (*read*) register-register tersebut UART 8250 dapat diprogram sesuai dengan keadaan yang dikehendaki. Secara umum register-register pada UART 8250 dapat dilihat pada gambar 2.7.

Sebelum membahas fungsi dari masing-masing register tersebut, ada satu hal yang harus diketahui yaitu alamat memori masing-masing register. Pengodean alamat register-register tersebut dapat dilihat pada tabel 2.2. Seperti yang terlihat pada tabel 2.2, UART 8250 memiliki 11 macam register, namun hanya ada 10 alamat memori. Hal ini bisa terjadi karena *transmitter buffer register* dan *receiver buffer register* diletakkan pada alamat memori yang sama. Selain itu, pada bagian perangkat keras UART 8250, terdapat 3 buah pin alamat memori (A0-A2). Hal ini berarti hanya ada 8 alamat memori yang dapat dikodekan, padahal UART 8250 memiliki 10 alamat memori yang harus dikodekan. Untuk mengatasi masalah ini digunakan bit ke-7 dari *data format* atau *line control register* sebagai dekoder tambahan. Bit ke-7 ini disebut dengan "*Divisor Latch Access Bit*" atau DLAB. Ilustrasi penggunaan DLAB dalam pengodean alamat register dapat dilihat pada tabel 2.2.



GAMBAR 2.7[8)]
BLOK DIAGRAM UART 8250

Berikut ini akan diuraikan fungsi masing-masing register UART 8250 :

RECEIVER BUFFER REGISTER

Receiver buffer register terdiri dari satu byte memori. Register ini

8) Ibid, p.215

TABEL 2.2[9]
PENGALAMATAN REGISTER DAN DLAB

DLAB[1]	A0	A1	A2	Read/ Write	Register
0	0	0	0	-	Receiver (read) Transmitter (write)
0	0	0	1	R/W	Interrupt Enable
X	0	1	0	READ	Interrupt Identification
X	0	1	1	R/W	Data Format (Line Control)
X	1	0	0	R/W	RS-232 Output (Modem Control)
X	1	0	1	R/W	Serialization Status (Line Status)
X	1	1	0	R/W	RS-232 Input Status (Modem Status)
X	1	1	1	R/W	Scratch Pad
1	0	0	0	R/W	LSB Baud Rate Divisor Latch
1	0	0	1	R/W	MSB Baud Rate Divisor Latch

[1] *"Divisor Latch Access Bit"*, bit ke-7 dari *Line Control Register*.

terisi bila UART menerima data. Meskipun data yang diterima kurang dari 8 bit, CPU akan selalu mengambil 8 bit data dari register ini. Alamat register ini adalah $3F8 untuk COM 1 dan $2F8 untuk COM 2.

TRANSMITTER BUFFER REGISTER

Untuk mengirim sebuah byte data, CPU menulis byte data tersebut ke dalam register ini. *Transmitter buffer register* memiliki alamat yang sama dengan *receiver buffer register* yaitu $3F8 untuk COM 1 dan $2F8 untuk COM 2.

INTERRUPT ENABLE REGISTER

Rincian fungsi dari bit-bit yang terdapat pada register ini tampak pada gambar 2.8.

[9] Ibid, p. 215

7	6	5	4	3	2	1	0
Always 0	Always 0	Always 0	Always 0	RS-232 Input	Receiver Error or BREAK	Transmitter Buffer Empty (TBE)	Receiver Data (RxRDY)

GAMBAR 2.8[10)]
INTERRUPT ENABLE REGISTER 8250

Bit 0 Bit ini bernilai 1, jika UART telah menerima data. Kemudian UART mengaktifkan interupsi ke CPU.

Bit 1 Bit ini bernilai 1, jika *transmitter buffer register* kosong. Kemudian UART mengaktifkan interupsi ke CPU.

Bit 2 Bit ini bernilai 1, UART mengaktifkan interupsi ke CPU karena salah satu keadaan dari *line status register*, seperti : *parity error*, *overrun error*, *framing error*, atau kondisi *BREAK* terdeteksi oleh penerima selama penerimaan sebuah byte data.

Bit 3 Bit ini bernilai 1, UART mengaktifkan interupsi ke CPU karena salah satu keadaan dari *modem status register* atau terjadi perubahan keadaan dari input RS-232.

Bit 4-7 Bit-bit ini selalu bernilai 0.

Seperti yang terlihat pada tabel 2.2, register ini aktif jika bit ke-7 dari *line control register* (DLAB) bernilai salah (0). *Interrupt enable register* memiliki alamat $3F9 untuk COM 1 dan $2F9 untuk COM 2.

INTERRUPT IDENTIFICATION REGISTER

Interrupt identification register menampung pemrograman agar dapat

[10)] Ibid, p. 217

menentukan bagian mana yang diberi urutan prioritas khusus untuk dapat melangsungkan interupsi ke CPU. Fungsi bit-bit register ini dapat dilihat pada gambar 2.9. Bila bit 0 dari register ini bernilai salah (0) berarti sebuah interupsi sedang menunggu. Pada tabel 2.3 tampak bahwa masing-masing interupsi mempunyai prioritas sendiri-sendiri tergantung dari nilai dari bit 0 sampai bit 2.

Interrupt identification register memiliki alamat $3FA untuk COM 1 dan $2FA untuk COM 2.

DATA FORMAT (LINE CONTROL) REGISTER

Line control register menampung ketentuan yang dipilih untuk menentukan berapa jumlah bit untuk setiap data, berapa jumlah *stop bit*-nya, apakah menggunakan *parity check*. Di samping itu juga melayani apakah akan menentukan/mengubah *baud rate divisor*. Fungsi bit-bit pada register ini dapat dilihat pada gambar 2.10 dan uraian berikut ini:

Bit 2 Jika setiap data terdiri dari 5 bit, maka *stop bit* 1½ akan terpilih secara otomatis.

Bit 3-5 Banyak penjelasan tentang ketiga bit ini, termasuk *data sheet*, menyebut "*Parity Enable*" untuk bit 3, "*Parity Select*" untuk bit 4, dan "*Stick Parity*" untuk bit 5.

Bit 6 Bit ini bernilai 1 berarti *break control* menjadi aktif. Bila hal ini terjadi maka output *transmitter* "dipaksa" menghasilkan nilai logika nol (*SPACE*). *Transmitter* akan tetap pada kondisi ini

TABEL 2.3 [11]
PENGODEAN INTERRUPT IDENTIFICATION REGISTER 8250

Bit 2	Bit 1	Bit 0	Prioritas[1]	ID Interupsi
0	0	1	Tidak ada	Tidak ada
1	1	0	0	Keadaan *line status register*
1	0	0	1	Penerimaan data
0	1	0	2	*Transmitter buffer* kosong
0	0	0	3	Keadaan *modem status register*

[1] 0 prioritas tertinggi

7	6	5	4	3	2	1	0
Reserved	Reserved	Always 0	Always 0	Reserved	Lihat Tabel 2.3		Interrupt Pending

GAMBAR 2.9[12]
INTERRUPT IDENTIFICATION REGISTER 8250

7	6	5	4	3	2	1	0
DLAB	BREAK Control 0 = off 1 = on	Parity 000 = NONE 001 = ODD 011 = EVEN 101 = MARK 111 = SPACE			Number of STOP Bits : 0 = 1 1 = 2	Number of Data Bits : 00 = 5 01 = 6 10 = 7 11 = 8	

GAMBAR 2.10[13]
LINE CONTROL REGISTER 8250

sampai bit ini diisi atau ditulis dengan nilai logika nol.

Bit 7 Bit ini merupakan DLAB (*Divisor Latch Access Bit*). Bila bit ini berisi 1, berarti *baud rate divisor* akan diakses dan bila bit ini berisi 0, berarti *baud rate divisor* tidak akan diakses.

Line control register mempunyai alamat $3FB untuk COM 1 dan $2FB untuk COM 2.

---

[11] Ibid, p.218
[12] Ibid, p.217
[13] Ibid, p.218

RS-232 OUTPUT CONTROL (MODEM CONTROL) REGISTER

Modem control register menampung pemrograman untuk mengatur modem, terutama menggunakan saluran DTR (pin 33) dan saluran RTS (pin 32) dari UART ke perangkat modem. UART dapat diprogram untuk melakukan *loop back*, yaitu data yang dikirim dapat diterima sendiri oleh UART. Kemampuan ini sangat bermanfaat untuk melakukan tes mengirim dan tes menerima data pada UART di dalam IBM PC, sehingga tidak memerlukan perangkat lain yang merepotkan. UART memberi ekstra 2 saluran OUT1 dan OUT2 untuk disambung ke komponen lain sebagai pengatur *enable/disable*. Penerapannya di IBM PC adalah untuk mengatur saluran interupsi INTRPT (pin 30) di UART. Saluran INTRPT itu diberi *tri-state buffer* yang diatur oleh OUT2. Oleh karena itu agar UART dapat beroperasi berdasar interupsi dari peralatan lain di luar, maka bit 3 (OUT2) pada *modem control register* harus diberi nilai logika 1.

Fungsi bit-bit register ini secara lengkap dapat dilihat pada gambar 2.11 dan uraian berikut ini:

Bit 0 Jika bit ini diisi 1, maka saluran DTR di pin 33 UART diaktifkan (level 0).

Bit 1 Jika bit ini diisi 1, maka saluran RTS di pin 32 UART diaktifkan (level 0).

Bit 2 Bit ini merupakan saluran pertama yang dapat dihubungkan ke perangkat lain.

7	6	5	4	3	2	1	0
Always 0	Always 0	Always 0	Local Loopback Test	General Purpose Output #2 (GPO2)	General Purpose Output #1 (GPO1)	RTS (4)	DTR (20)

GAMBAR 2.11[14)]
MODEM CONTROL REGISTER 8250

Bit 3 Bit ini merupakan saluran kedua yang dapat dihubungkan ke perangkat lain.

Bit 4 Jika bit ini diisi 1, maka *loop back* internal diaktifkan, artinya data yang dikirim dapat diterima kembali oleh UART.

Bit 5-7 Bit-bit ini selalu bernilai 0.

Modem control register memiliki alamat $3FC untuk COM 1 dan $2FC untuk COM 2.

SERIALIZATION STATUS (LINE STATUS) REGISTER

Line status register menampung bit-bit yang menyatakan keadaan kehadiran data dan keadaan kesalahan operasi. Rincian fungsi bit-bit register ini dapat dilihat pada gambar 2.12 dan uraian berikut ini.

Bit 0 Bit ini bernilai 1 bila telah ada data yang masuk ke *receiver buffer register*.

Bit 1 Bit ini bernilai 1 bila data yang masuk ke *receiver buffer register* saling tumpuk (*overrun error*).

14) Ibid, p.219

7	6	5	4	3	2	1	0
Always 0	Transmitter Empty (TXE)	Transmitter Buffer Empty (TBE)	BREAK Detect	Framing Error	Parity Error	Overrun Error	Data Ready (RxRDY)

GAMBAR 2.12[15]
LINE STATUS REGISTER 8250

Bit 2 Bit ini bernilai 1 bila terjadi kesalahan *parity* pada data yang diterima.

Bit 3 Bit ini bernilai 1 bila jumlah bit pada setiap data yang masuk tidak benar (*framing error*).

Bit 4 Bit ini bernilai 1 bila terjadi interupsi yang menghentikan (*break interrupt*).

Bit 5 Bit ini bernilai 1 bila isi dari *transmitter buffer register* telah kosong.

Bit 6 Bit ini bernilai 1 bila *transmitter shift register* telah kosong. Bit ini disebut juga bit TXE atau *Transmitter Empty*.

Bit 7 Bit ini selalu bernilai 0.

Line status register memiliki alamat $3FD untuk COM 1 dan $2FD untuk COM 2.

RS-232 INPUT STATUS (MODEM STATUS) REGISTER

Modem status register menampung bit-bit yang menyatakan keadaan tentang hubungan dengan modem atau perangkat lain yang dihubungkan dengan UART di IBM PC. Fungsi bit-bit register ini dapat dilihat pada gambar 2.13.

---

[15] Ibid, p.220

7	6	5	4	3	2	1	0
DCD	RI	DSR	CTS	Delta DCD	Delta RI	Delta DSR	Delta CTS

GAMBAR 2.13[16)]
MODEM STATUS REGISTER 8250

Bit 0-3 Bit-bit ini menyatakan perubahan keadaan yang terjadi pada pin-pin RS-232. Nilai logika 1 pada masing-masing bit ini berarti keadaan pin-pin RS-232 telah berubah dari pembacaan register ini sebelumnya. Pembacaan register ini menyebabkan bit 0-3 bernilai logika 0.

Bit 4-7 Bit-bit ini menyatakan keadaan sebenarnya dari pin-pin RS-232. Jika bit-bit ini bernilai 1, maka pin-pin RS-232 dalam keadaan aktif.

Modem status register mempunyai alamat $3FE untuk COM 1 dan $2FE untuk COM 2.

SCRATCH PAD

Register ini tidak mempunyai fungsi apapun. Register ini dapat digunakan seperti penggunaan RAM, tetapi register ini tidak selalu ada pada seluruh versi dari UART 8250.

BAUD RATE DIVISOR LATCH

Ada 2 macam *baud rate divisor latch* yaitu :

16) Ibid, p. 221

- **LSB BAUD RATE DIVISOR LATCH**

  LSB (Least Significant Byte) baud rate divisor latch menampung angka byte level rendah untuk pembagi *clock* yang dimasukkan ke UART agar didapat *baud rate* yang diinginkan. Angka pembagi ini dapat dipilih antara $00 hingga $FF.

- **MSB BAUD RATE DIVISOR LATCH**

  MSB (Most Significant Byte) baud rate divisor latch menampung angka byte level tinggi untuk pembagi *clock* yang dimasukkan ke UART agar didapat *baud rate* yang diinginkan. Angka pembagi ini dapat dipilih antara $00 hingga $FF.

Dengan demikian angka pembagi *baud rate* dapat dipilih antara angka $00 hingga $FFFF. Frekuensi dari input *clock* yang biasa digunakan adalah 1,8432 MHz.Untuk menentukan *baud rate*, frekuensi ini harus dibagi dengan angka 1 word dengan byte tinggi pada *MSB baud rate divisor* dan byte rendah pada *LSB baud rate divisor*. Perhitungan ini harus sesuai dengan persamaan berikut ini :

$$\text{Pembagi} = \frac{\text{Frekuensi Clock UART}}{16 \times \text{Baud Rate}} \qquad (2.1)$$

Dengan menggunakan persamaan 2.1 di atas, dapat diperoleh angka-angka pembagi untuk masing-masing *baud rate*. Tabel 2.4 menunjukkan nilai angka pembagi untuk nilai *baud rate* tertentu.

TABEL 2.4[17]
ANGKA PEMBAGI BAUD RATE UNTUK CLOCK 1,8432 MHz

Baud Rate	Angka Pembagi
50	$0900
75	$0600
110	$0417
134.5	$0359
150	$0300
300	$0180
600	$00C0
1,200	$0060
1,800	$0040
2,000	$003A
2,400	$0030
3,600	$0020
4,800	$0018
7,200	$0010
9,600	$000C
12,000	-
14,400	$0008
19,200	$0006
28,800	$0004
38,400	$0003
57,600	$0002
115,200	$0001

[17] Ibid, p.223

II.4 MODULATOR DEMODULATOR

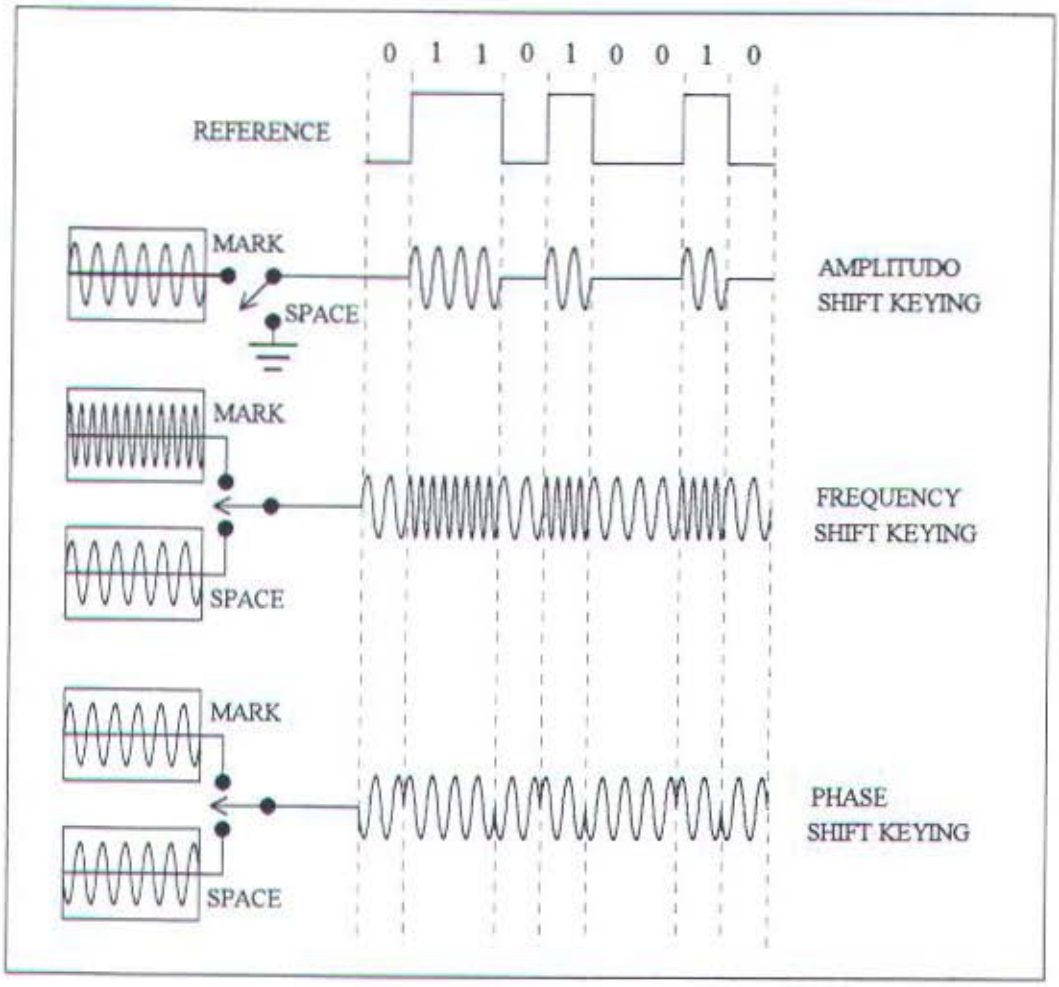
Modulator demodulator sering dikenal dengan sebutan modem. Sesuai dengan namanya, modem adalah suatu alat yang berfungsi sebagai modulator sinyal digital (dari peralatan seperti komputer) menjadi nada-nada audio dan demodulator nada-nada audio menjadi sinyal digital. Jadi dengan menggunakan modem komunikasi data dapat dilakukan melalui saluran sinyal analog, seperti saluran telepon, gelombang radio, dan sebagainya. Berikut ini akan dibahas tentang dasar-dasar modem dan bagaimana pemrograman modem-modem cerdas (*smart modem*).

II.4.1 Dasar-dasar Modem

Seperti yang telah diuraikan sebelumnya, modem merupakan suatu alat untuk mengubah sinyal digital, seperti sinyal digital komputer , menjadi sinyal analog frekuensi rendah (frekuensi percakapan manusia). Hal ini dilakukan agar sinyal digital dapat ditransmisikan melalui saluran telepon yang memiliki jangkauan lebih jauh dan luas. Ada beberapa hal yang menyangkut dengan proses modulator dan demodulator, yaitu :

- **Teknik Modulasi**

  Ada beberapa teknik modulasi yang dapat digunakan yaitu *amplitudo shift keying*, *frequency shift keying*, dan *phase shift keying*. Ilustrasi mengenai ketiga jenis modulasi ini dapat dilihat pada gambar 2.14.



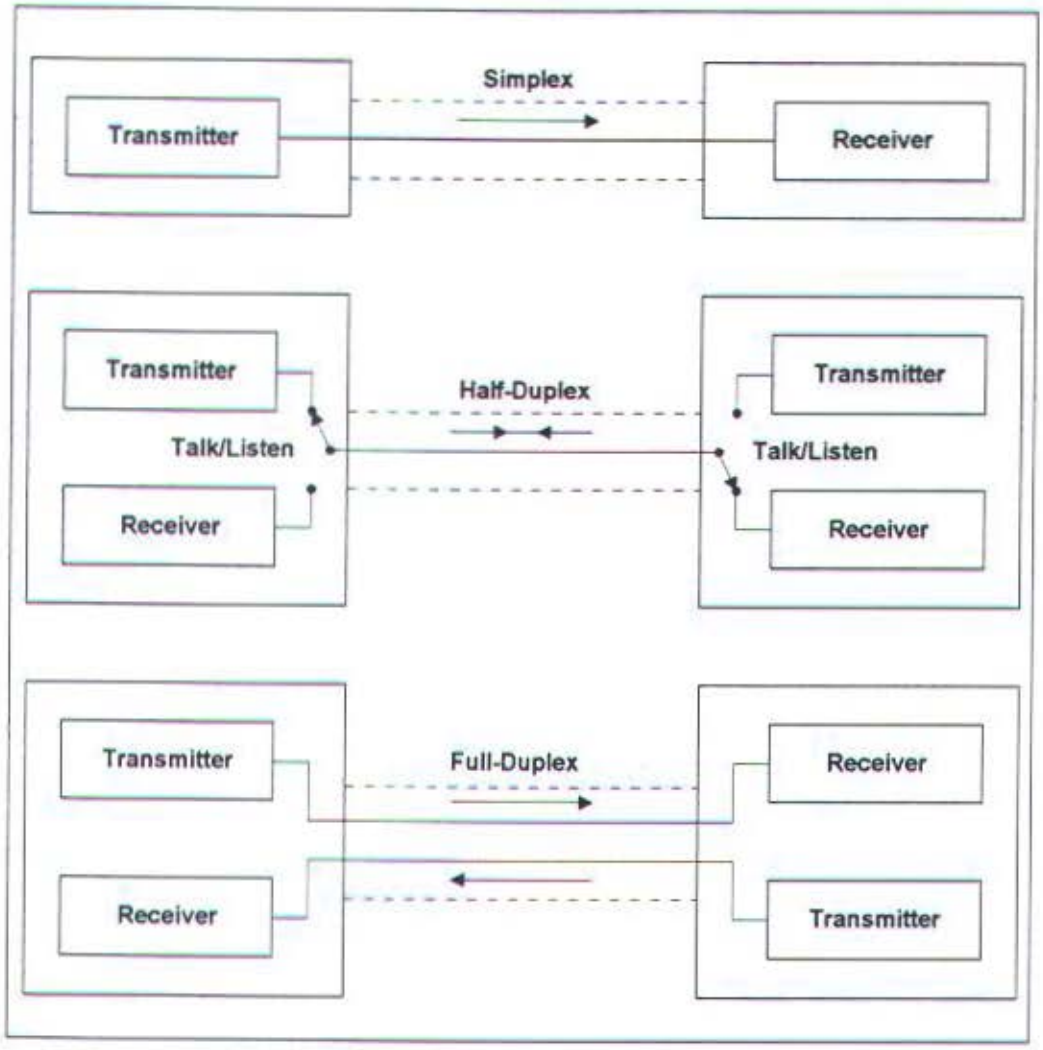
GAMBAR 2.14[18]
MODULASI PADA MODEM

Dari ketiga jenis modulasi ini, *amplitudo shift keying* merupakan modulasi yang paling peka terhadap *noise*. Hal ini menyebabkan kemungkinan terjadi kesalahan pengiriman semakin besar.

➢ **Hubungan Antar Modem**

Ada 3 jenis hubungan antar modem yaitu hubungan *simplex*, hubungan *half-duplex*, dan hubungan *full-duplex*. Gambar 2.15 menunjukkan ketiga jenis hubungan ini.

[18] Ibid, p.138, p.143, p.145



GAMBAR 2.15[19]
PENGGUNAAN BANDWIDTH DALAM HUBUNGAN MODEM

Pada hubungan *simplex*, modem pengirim terus menerus mengirimkan data tanpa memperdulikan apakah modem penerima menerima data tersebut dengan benar. Pada hubungan *half-duplex*, pihak pengirim dan penerima dapat berkomunikasi, tetapi harus dilakukan secara bergantian. Sedangkan pada hubungan *full-duplex*, kedua belah pihak dapat mengirim maupun menerima data pada saat yang bersamaan.

[19] Ibid, p.140

II.4.2 Pemrograman Smart Modem

Smart modem adalah modem yang dapat dikontrol dengan mengirimkan perintah, dalam bentuk deretan karakter ASCII, dari komputer melalui saluran RS-232. Untuk memberikan perintah ke *smart modem* harus diawali oleh karakter "AT", singkatan dari *attention*. Dengan memberikan perintah AT ke *smart modem*, modem dapat diatur sesuai keperluan seperti: melakukan proses *dialing* (ATD), mengubah kelajuan transfer data (AT), memutuskan hubungan (ATH) dan sebagainya. Perintah-perintah modem secara keseluruhan dapat dilihat pada tabel 2.5.

Selain itu, *smart modem* juga mempunyai register-register yang dapat diinisialisasi ulang dan isi register ini tidak terhapus ketika aliran listrik mati. Untuk mengatur dan mengecek isi register modem digunakan perintah ATS. Fungsi masing-masing register untuk modem 9624 dapat dilihat pada tabel 2.6. Untuk modem yang lebih tinggi teknologinya, register yang ada akan lebih banyak.

Dari sekian banyak perintah modem, perintah yang paling berhubungan dalam mengerjakan tugas akhir ini adalah perintah mengatur kelajuan transfer data. Prosedur mengatur kecepatan modem adalah sebagai berikut:

- menginisialisasi *baud rate* UART sesuai dengan *baud rate* yang diinginkan.
- mengirimkan perintah AT ke modem. Setelah DTE menerima respon dari modem, kecepatan modem berubah sesuai kecepatan UART.

TABEL 2.5[20)]
PERINTAH AT PADA SMART MODEM

AT Command	Default	Function
A/	none	Repeat last command
A	none	Answer
Bn	1	Select CCITT or Bell
Cn	1	Carrier control option
D	none	Dial command
En	1	Command echo
Fn	1	On-Line echo
Hn	0	Switch hook control
In	0	Identification/checksum
Ln	2	Speaker volume control
Mn	1	Speaker control
Nn	1	Connection data rate control
On	0	Go on line
P	none	Select pulse dialing
Qn	0	Result code display control
Sn	none	Selects an S-Register
Sn=x	none	Write to an S-Register
Sn ?	none	Read from an S-Register
?	none	Read from an S-Register
T	none	Select DTMF dialing
Vn	1	Result code form
Xn	4	Result code type
Yn	0	Long space disconnect
Zn	0	Recall stored profile
&Cn	1	DCD option
&Dn	2	DTR option
&F	none	Load factory defaults
&Gn	0	Guard tone option
&Ln	0	Dial-up/leased line option
&Mn	0	Communication mode option
&Pn	0	Dial pulse ratio
&Qn	0	Communication mode option
&Rn	0	RTS/CTS option
&Sn	0	DSR option
&Tn	0	Self test commands
&Vn	0	View active and stored configuration
&Wn	0	Stored active profile
&Xn	0	Sync transmit clock source option
&Yn	0	Select stored profile on power up
&Zn=x	none	Store telephone number
%En	1	Auto-retain control

20) _____, *Modem Instruction Manual*, p.1

TABEL 2.6[21)]
REGISTER S PADA SMART MODEM

S Register	Default	Function
S0	0	No. of rings to auto answer on
S1	0	Ring count
S2	43	Escape character
S3	13	Carriage return character
S4	10	Line feed character
S5	8	Backspace character
S6	2	Wait before dialing
S7	30	Wait for carrier
S8	2	Pause time for dial modifier
S9	6	Carrier recovery time
S10	14	Lost carrier hang up delay
S11	70	DTMF dialing speed
S12	50	Guard Time
S13	none	Reserved
S14	none	Bit-mapped options
S15	none	Reserved
S16	none	Modem test options
S17	none	Reserved
S18	0	Modem test timer
S19	none	Reserved
S20	none	Reserved
S21	none	Bit-mapped options
S22	none	Bit-mapped options
S23	none	Bit-mapped options
S24	none	Reserved
S25	5	Detect DTR change
S26	1	RTS to CTS delay interval
S27	none	Bit-mapped options
S30	10	Sleep mode timer

21) Ibid., p.3

II.5 PROTOKOL TRANSFER DATA

Bagian terakhir dari komunikasi data serial adalah protokol transfer data. Protokol inilah yang memungkinkan terjadi perpindahan data dari *transmitter* ke *receiver* secara benar. Ada beberapa macam protokol transfer data, antara lain:

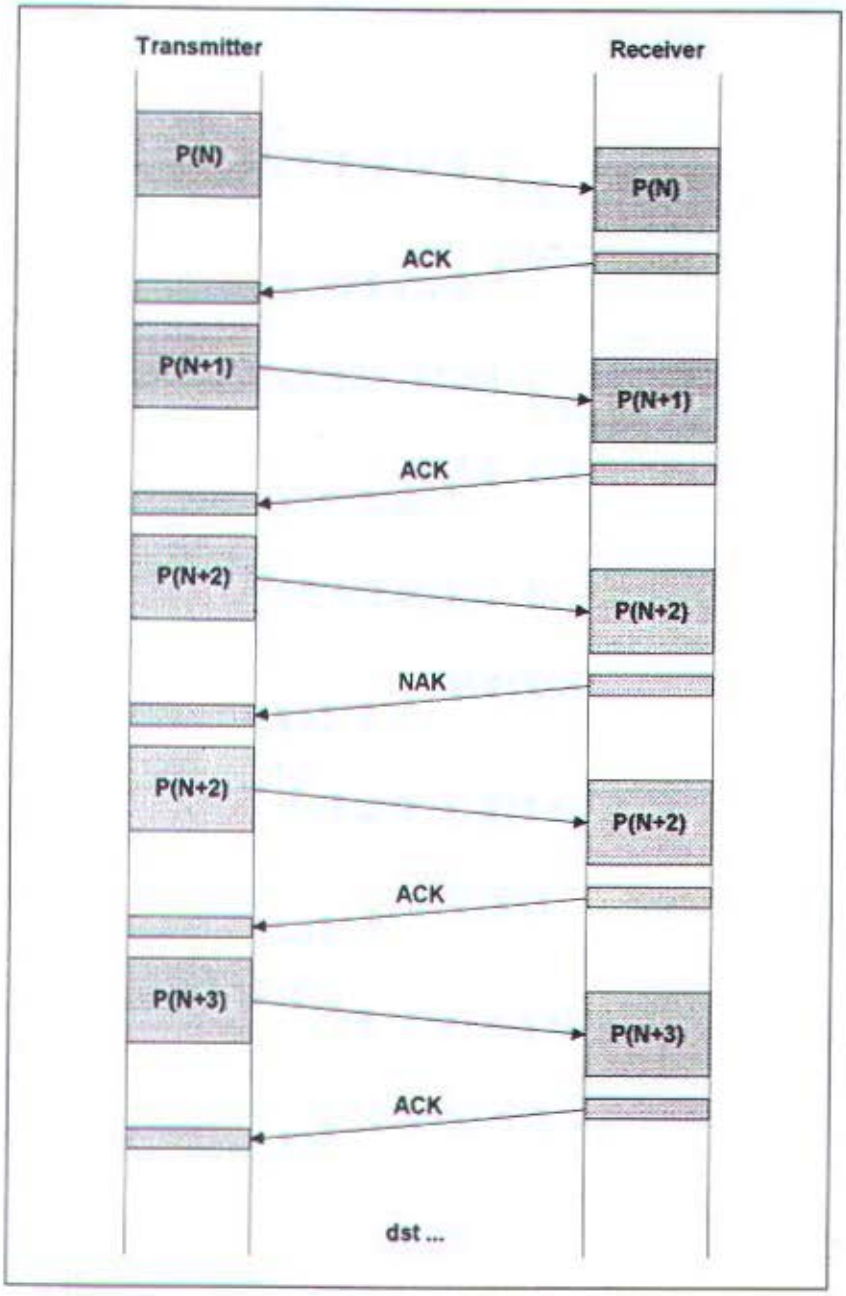
II.5.1 Automatic Repeat Request (ARQ) Protocol

Tipe protokol paket data yang paling umum digunakan adalah protokol jenis ini. Jika terjadi kesalahan penerimaan pada penerima maka *unacknowledged packet* secara otomatis menyebabkan pengiriman ulang dari paket yang salah. Ada beberapa jenis dari *ARQ protocol*, antara lain:

- **Send-and-Wait ARQ**

  Ilustrasi cara kerja protokol jenis ini dapat dilihat pada gambar 2.16. Pada protokol jenis ini pengirim mengirim paket ke penerima kemudian menunggu jawaban dari penerima. Jika jawaban yang diterima adalah ACK (*Acknowledge*) maka pengirim melakukan pengiriman paket berikutnya. Jika jawaban yang diterima adalah NAK (*Negative Acknowledge*), maka pengirim diminta untuk mengirim kembali paket sebelumnya karena terjadi kesalahan penerimaan.

  Bagian penerima mengirimkan jawaban ACK bila paket yang diterima, setelah melalui proses pengecekan kesalahan, sesuai dengan paket yang dikirimkan. Jika pada proses pengecekan kesalahan terdeteksi kesalahan paket maka penerima mengirimkan jawaban NAK. Demikian seterusnya



GAMBAR 2.16
SEND AND WAIT ARQ

proses ini berlangsung hingga pengiriman paket berakhir dengan benar.

- **Continuous ARQ**

Pada *continuous ARQ protocol*, pengirim tidak menunggu jawaban dari penerima. Pengirim melakukan pengiriman paket terus menerus hingga

seluruh paket telah terkirim. Di lain pihak, penerima melakukan pengiriman jawaban seperti biasa, ACK bila paket yang diterima benar dan NAK bila paket yang diterima salah, namun jawaban NAK ini disertai nomor paket yang salah.

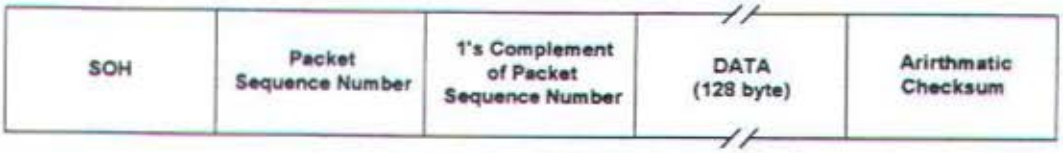
Setelah pengirim mengirim semua paket, pengirim mulai melihat jawaban dari penerima. Bila ditemui jawaban NAK, pengirim mengirim ulang paket sesuai dengan nomor paket yang diminta.

Desain paket data yang dapat digunakan untuk *ARQ protocol* bermacam-macam. Ada 3 pendekatan dasar yang dapat digunakan untuk menyusun sebuah paket data, yaitu:

- Menandai awal dan akhir dari data dengan menggunakan karakter kontrol.
- Memasukkan panjang data pada paket yang digunakan.
- Menggunakan panjang data tertentu.

II.5.2 XMODEM PROTOCOL

XMODEM file transfer protocol ditemukan pada tahun 1977 oleh Ward Christensen. Protokol XMODEM merupakan salah satu bentuk sederhana dari *send-and-wait protocol* yang menggunakan panjang paket data tertentu. Model paket dari protokol XMODEM tampak pada gambar 2.17. Masing-masing *field* kecuali *field* data mempunyai panjang 1 byte. Jadi secara keseluruhan panjang paket XMODEM adalah 132 byte.



GAMBAR 2.17[20)]
MODEL PAKET PROTOKOL XMODEM

SOH *Field* ini berisi byte *Start-of-Header* yang menunjukkan byte pertama dalam paket.

Packet sequence *Field* ini berisi nomer paket dengan modulo 256. Nomer paket pertama adalah 1.

1's complement of packet sequence *Field* ini berisi komplemen pertama dari nomer paket.

DATA *Field* ini berisi byte-byte data yang akan dikirimkan.

Aritmatic Checksum *Field* ini berisi hasil penjumlahan dari seluruh isi *field* data, dengan modulo 256.

[20)] Ibid, p.101

BAB III
PERENCANAAN DAN PEMBUATAN PROGRAM PENGONTROL BAUD RATE

Pada bab ini akan dibahas tentang perencanaan dan pembuatan program pengontrol *baud rate*. Pada bagian pertama akan dibahas tentang model sistem komunikasi data yang akan digunakan dan pada bagian kedua akan dibahas tentang penulisan model sistem komunikasi data tersebut ke dalam bahasa pemrograman C++.

III.1 MODEL SISTEM KOMUNIKASI DATA

Seperti yang telah diuraikan pada bab-bab sebelumnya, sistem komunikasi data yang akan dibuat adalah sistem komunikasi data asinkron. Pada sistem komunikasi data jenis ini diperlukan suatu protokol. Protokol berfungsi untuk mengatur bagaimana cara pertukaran data dapat berlangsung dengan benar. Jadi penerima dapat mengerti apa yang dikirimkan pengirim dan sebaliknya pengirim dapat mengerti tanggapan penerima. Hal ini berhubungan dengan *error checking*.

Selain protokol yang digunakan, hal lain yang perlu diperhatikan adalah saluran yang digunakan dalam melakukan komunikasi data. Dalam tugas akhir ini, saluran yang akan dipakai adalah saluran telepon. Namun untuk

mempermudah pembuatan program, digunakan saluran RS-232 sebagai saluran uji coba awal dan dilanjutkan pada saluran telepon. Berikut ini uraian dari dua masalah di atas :

III.1.1 Protokol Transfer Data

Protokol yang digunakan dalam sistem komunikasi data ini adalah *Send And Wait ARQ Protocol*, dengan menggunakan paket data yang mempunyai panjang tertentu. Protokol *send and wait ARQ* digunakan sebagai model sistem komunikasi data karena sesuai dengan metode optimasi *baud rate* yang digunakan. Metode optimasi *baud rate* ini menggunakan parameter ACK untuk menaikkan *baud rate* dan parameter NAK untuk menurunkan *baud rate*. Hal ini sesuai dengan pernyataan "Semakin padat trafik pada saluran, kelajuan optimal pengiriman data akan semakin turun". Jika pada trafik padat, kelajuan pengiriman data melebihi kelajuan optimal, maka akan terjadi *collision* atau tubrukan. Hal ini menyebabkan timbulnya kesalahan data pada bagian penerima, sehingga penerima mengirimkan data NAK untuk memberi tahu pengirim bahwa data diterima dengan salah. Jika hal ini berulang berkali-kali maka kelajuan pengiriman data harus diturunkan. Demikian sebaliknya, jika pengiriman berlangsung baik maka kelajuan pengiriman dinaikkan. Dengan demikian kelajuan pengiriman data dapat berlangsung dengan kelajuan optimal.

Model paket yang digunakan dalam sistem komunikasi data ini memiliki panjang paket 131 byte dan dapat dilihat pada gambar 3.1.

GAMBAR 3.1
MODEL PAKET DATA

Fungsi masing-masing *field* pada paket dapat dilihat pada uraian berikut ini.

- **SOH** Byte ini menunjukkan awal dari paket data.
- **Nomer Paket** Byte ini berisi nomer paket data dengan modulo 256 dan dimulai dari nomer 1.
- **Data** *Field* ini berisi data-data yang memiliki panjang 128 byte.
- **Checksum** Byte ini berisi jumlah dari seluruh data pada *field* data dengan modulo 256.

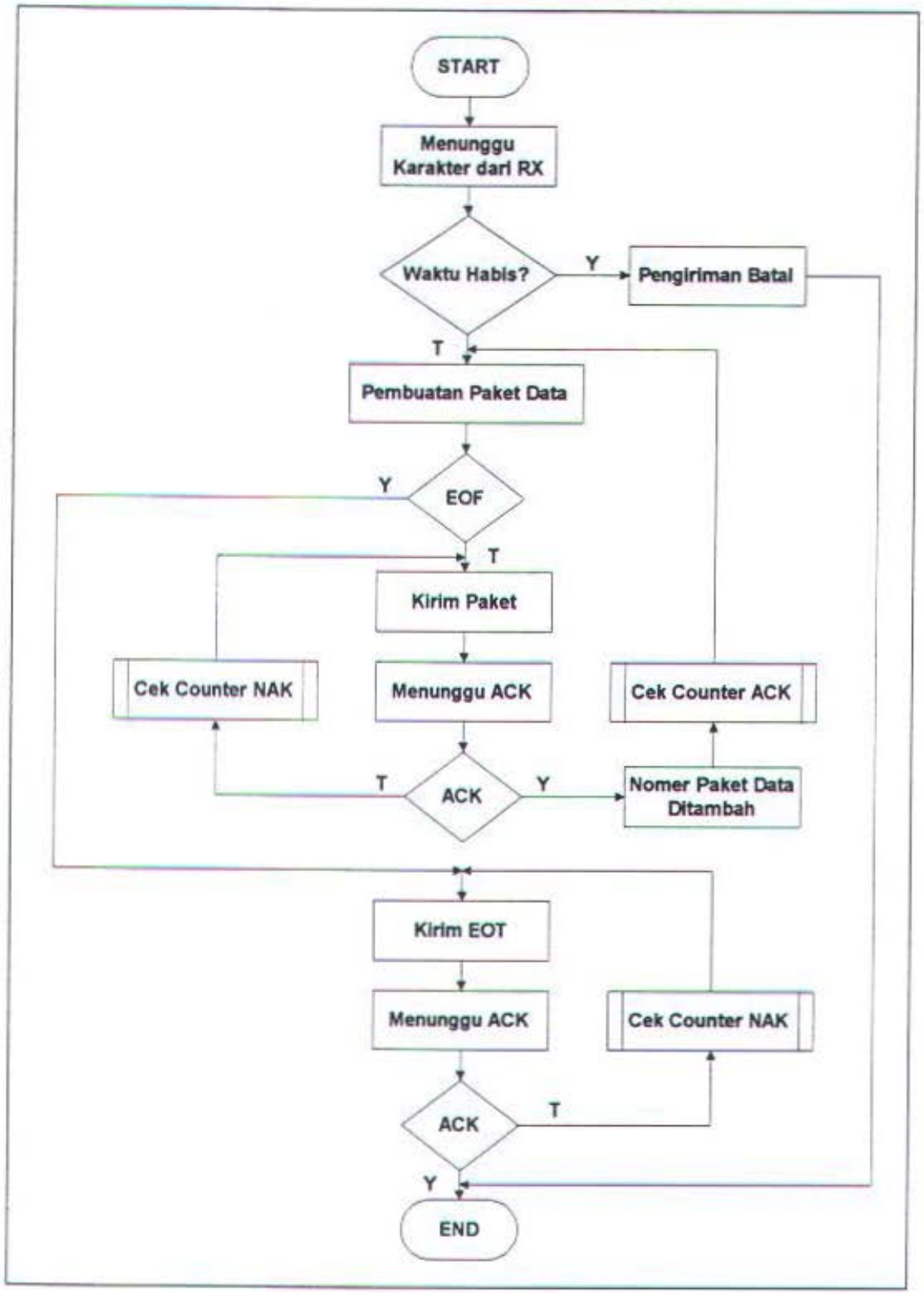
Setelah mengetahui model paket yang digunakan maka uraian berikutnya adalah tentang pengiriman dan penerimaan paket data.

PENGIRIMAN PAKET DATA

Diagram alir dari pengiriman paket dapat dilihat pada gambar 3.2. Tahap-tahap proses pada pengiriman paket data ada 3 tahap, yaitu :

- **Tahap Awal Pengiriman Paket Data**

  Tugas pertama dari semua protokol adalah membangun hubungan antara pengirim dan penerima. Untuk membangun hubungan maka pengirim harus menunggu sebuah byte data dari penerima.



GAMBAR 3.2
DIAGRAM ALIR PROTOKOL PENGIRIM

Jika pengirim sudah menerima sebuah byte, maka antara pengirim dan penerima sudah terjadi hubungan.

- **Tahap Menengah Pengiriman Paket Data**

  Pada tahap menengah, pengirim membuat paket yang berisi 128 byte data kemudian mengirimkannya. Setelah itu, pengirim menunggu jawaban dari penerima. Jika jawaban ACK yang diterima oleh pengirim, maka paket dapat diterima tanpa kesalahan di penerima. Jika jawaban NAK yang diterima oleh pengirim, maka terdapat kesalahan pada pengiriman paket dan meminta pengirim melakukan pengiriman ulang. Jika tidak ada lagi data yang akan dikirim maka tahap menengah sudah selesai.

- **Tahap Akhir Pengiriman Paket Data**

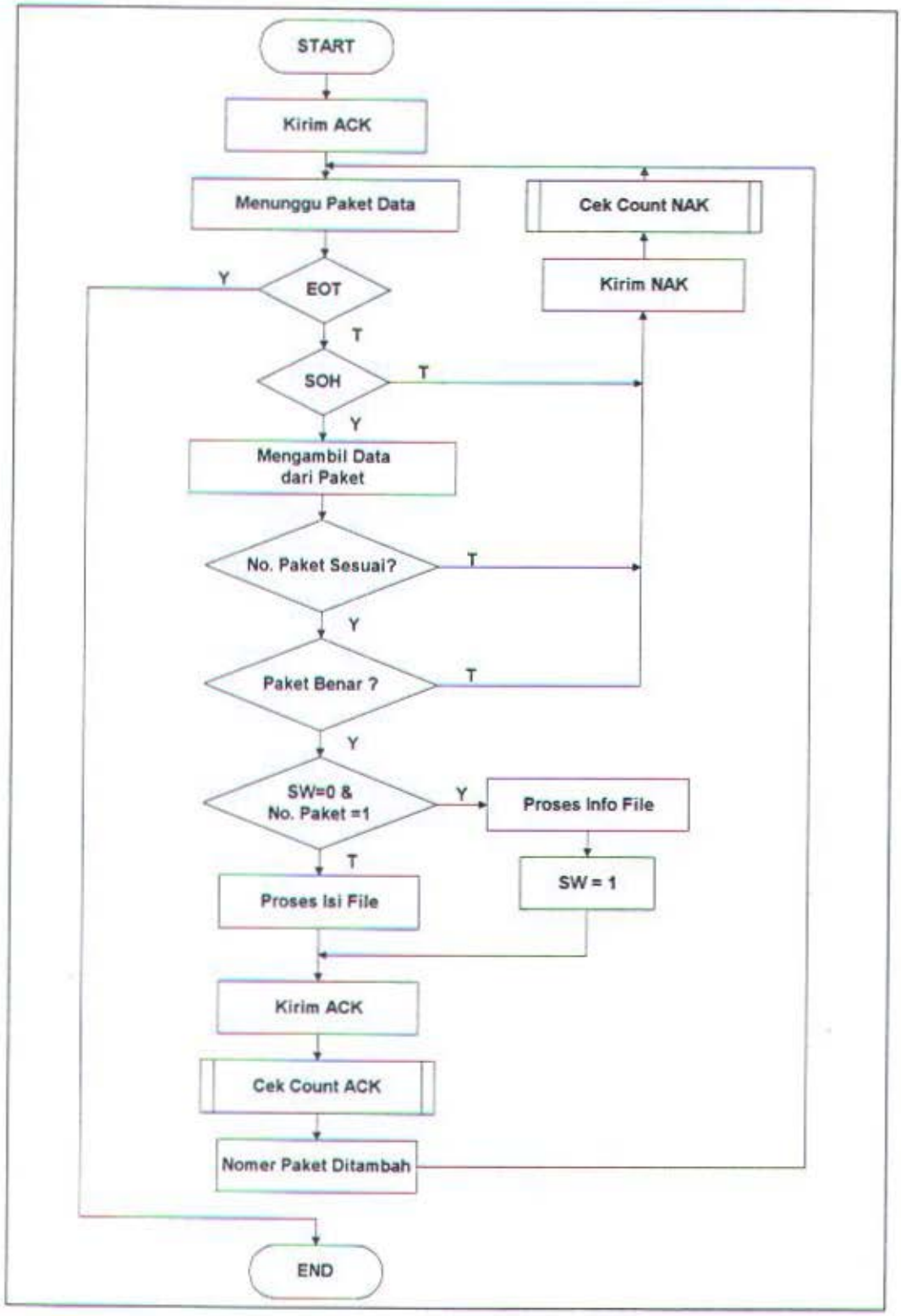
  Jika tahap menengah berakhir secara normal, maka pengirim akan mengirim karakter EOT. Selanjutnya pengirim menunggu jawaban dari penerima. Jika jawaban ACK yang diterima, maka pengiriman data telah selesai.

PENERIMAAN PAKET DATA

Diagram alir penerimaan paket data dapat dilihat pada gambar 3.3. Tahap-tahap penerimaan paket data juga dibagi menjadi 3 tahap, yaitu :

- **Tahap Awal Penerimaan Paket Data**

  Pada tahap ini penerima mengirim byte ACK untuk memberi tahu pengirim bahwa penerima sudah siap menerima paket data.



GAMBAR 3.3
DIAGRAM ALIR PROTOKOL PENERIMA

➢ **Tahap Menengah Penerimaan Paket Data**

Pada tahap ini penerima menunggu kedatangan paket data yang ditandai oleh karakter SOH. Setelah paket data diterima, proses yang dilakukan adalah melakukan pengecekan terhadap nomer paket apakah sesuai dengan yang seharusnya. Selain itu juga dilakukan proses kalkulasi penjumlahan (dengan modulo 256) terhadap 128 byte data, kemudian dicocokan dengan isi dari byte *checksum*. Jika sama, maka penerima akan memberikan jawaban ACK. Jika tidak sama, maka penerima akan memberikan jawaban NAK.

➢ **Tahap Akhir Penerimaan Paket Data**

Tahap ini dimulai dengan diterimanya karakter EOT. Proses-proses yang dilakukan kemudian yaitu mengirimkan jawaban ACK ke pengirim dan melakukan proses yang berhubungan dengan data yang telah diterima seperti penutupan file data, membersihkan memori dan sebagainya.

III.1.2 Saluran Pengiriman Data

Saluran yang digunakan dalam perencanaan sistem komunikasi data ini ada dua macam, yaitu :

➢ **Saluran RS-232 atau Null Modem**

Dengan menggunakan saluran ini, 2 buah komputer yang digunakan untuk pengiriman dan penerimaan data dihubungkan secara

langsung dengan menggunakan saluran RS-232. Jadi data yang dikirim tidak mengalami perubahan sinyal. Ilustrasi dari model komunikasi data dengan menggunakan saluran ini dapat dilihat pada gambar 3.4.a.

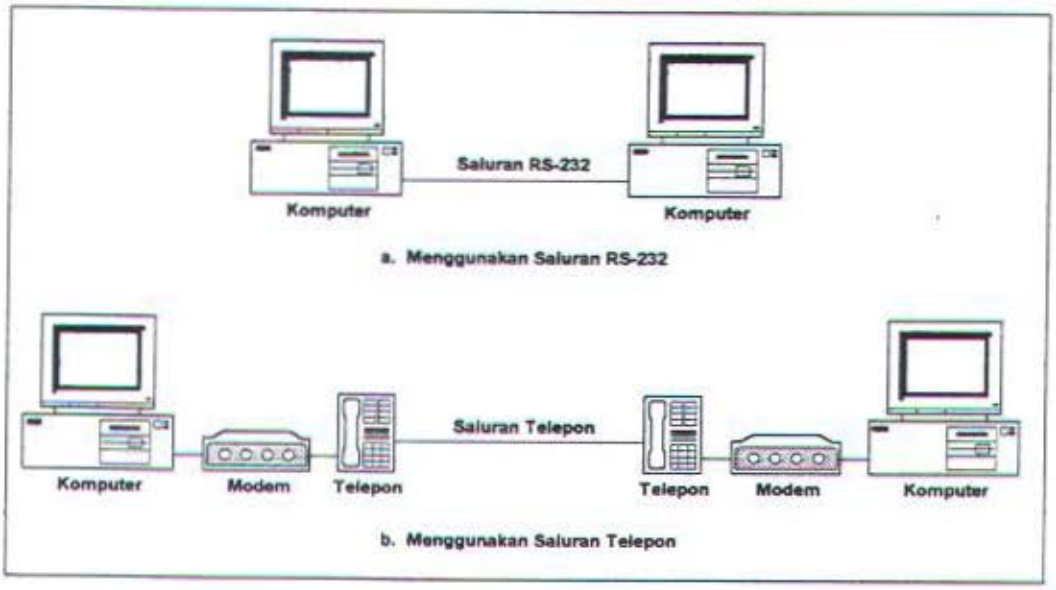
➢ **Saluran Telepon**

Dengan menggunakan saluran ini, 2 komputer yang digunakan untuk pengiriman dan penerimaan data harus dilengkapi dengan modem. Hal ini dikarenakan sinyal digital dari peralatan komputer tidak dapat ditransmisikan melalui saluran telepon secara langsung. Jadi sinyal digital ini harus diubah melalui modem. Demikian di bagian penerima, dilakukan pengubahan ke sinyal digital semula agar komputer dapat mengerti data yang diterima. Ilustrasi dari model komunikasi data dengan menggunakan saluran ini dapat dilihat pada gambar 3.4.b.

Jadi pemrograman yang akan dilakukan adalah pemrograman komunikasi data melalui RS-232 terlebih dahulu, dilanjutkan dengan penambahan program pengontrolan modem agar data dapat ditransmisikan melalui saluran telepon.

III.2 MODEL KOMUNIKASI DALAM BAHASA C++

Ada dua tahap pemrograman dalam model komunikasi data yang diinginkan, yaitu pemrograman UART dan pemrograman modem. Berikut ini uraian tentang kedua tahap pemrograman tersebut.



GAMBAR 3.4
SALURAN KOMUNIKASI DATA

III.2.1 Pemrograman UART

Dalam pemrograman UART, beberapa hal yang harus diperhatikan adalah :

- **Proses Pengiriman Data**

  Pengiriman data di sini dapat berarti dua, yaitu pengiriman sebuah byte karakter dan pengiriman sebuah paket data. *Listing program* dari prosedur kedua proses ini dapat dilihat pada uraian berikut ini :

  - **Prosedur pengiriman karakter**

```
void SendChar(char tx)
{
    while((inportb(LSR)&0x60) != 0x60);
    outportb(Tx,tx);
}
```

➢ **Prosedur pengiriman paket data**

```
void SendPacket()
{
    unsigned char j;
    for (j=0;j<131;j++)
    {
        SendChar(tpacket[j]);
    }
}
```

➢ **Proses Penerimaan Data**

Penerimaan data di sini dapat berarti dua, yaitu penerimaan sebuah byte karakter dan penerimaan sebuah paket data. *Listing program* dari prosedur kedua proses ini dapat dilihat pada uraian berikut ini :

➢ **Prosedur penerimaan karakter**

```
void ReceiveChar()
{
    char rx;
    while((inportb(LSR)&0x01) != 0x01);
    rx = inportb(Rx);
    return(rx);
}
```

➢ **Prosedur penerimaan paket data**

```
void ReceivePacket()
{
    unsigned char j;
    for (j=0;j<131;j++)
    {
        rpacket[j] = ReceiveChar();
    }
}
```

➢ **Proses Inisialisasi Port UART dan Pengubahan Baud Rate**

Proses pengubahan *baud rate* dilakukan dengan menginisialisasi ulang UART dengan *baud rate* yang diinginkan. Berikut merupakan *listing program* dari prosedur inisialisasi port UART dan prosedur pengubahan *baud rate* UART.

➢ **Prosedur inisialisasi port UART**

```
void initport()
{
   int addcom;

   Rx       = 0X2E8;
   Tx       = 0X2E8;
   LOBaud   = 0X2E8;
   HIBaud   = 0X2E9;
   IER      = 0X2E9;
   IIR      = 0X2EA;
   LCR      = 0X2EB;
   MCR      = 0X2EC;
   LSR      = 0X2ED;
   MSR      = 0X2EE;

   switch(NoCom)
   {
     case 0 :  addcom = 0X110;  /* Alamat Memory COM 1 --> 3F8 */
               break;
     case 1 :  addcom = 0X010;  /* Alamat Memory COM 2 --> 2F8 */
               break;
     case 2 :  addcom = 0X100;  /* Alamat Memory COM 3 --> 3E8 */
               break;
     case 3 :  addcom = 0X000;  /* Alamat Memory COM 4 --> 2E8 */
               break;
   }
```

```
    Rx      = Rx + addcom;
    Tx      = Tx + addcom;
    LOBaud  = LOBaud + addcom;
    HIBaud  = HIBaud + addcom;
    IER     = IER + addcom;
    IIR     = IIR + addcom;
    LCR     = LCR + addcom;
    MCR     = MCR + addcom;
    LSR     = LSR + addcom;
    MSR     = MSR + addcom;

    outportb(LCR,0X03);          /* parity disable, 1 stop bit, 8 bit data */
    outportb(IER,0X00);          /* interrupt disable */
}
```

➢ **Prosedur pengubahan baud rate**

```
void baud(char rate)
{
   switch (rate)
   {
      case 1  : HiRate    = 0x01;
                LoRate    = 0x80;
                BaudPrint = 300;
                break;
      case 2  : HiRate    = 0x00;
                LoRate    = 0x60;
                BaudPrint = 1200;
                break;
      case 3  : HiRate    = 0x00;
                LoRate    = 0x30;
                BaudPrint = 2400;
                break;
      case 4  : HiRate    = 0x00;
                LoRate    = 0x18;
                BaudPrint = 4800;
                break;
```

```
        case 5  : HiRate    = 0x00;
                  LoRate    = 0x0c;
                  BaudPrint = 9600;
                  break;
    }
    outportb(LCR,0x80);          /* mengaktifkan perubahan baud rate */
    outportb(LOBaud,LoRate);
    outportb(HIBaud,HiRate);
    outportb(LCR,0x03);          /* parity disable, 1 stop bit, 8 bits data */
}
```

➢ **Proses Pengontrolan Baud Rate**

Setelah mengetahui cara mengubah *baud rate*, selanjutnya diperlukan suatu prosedur yang berguna untuk mengontrol *baud rate*. Fungsi prosedur ini adalah menentukan bilamana *baud rate* harus dinaikkan atau diturunkan. Untuk mempermudah, prosedur ini dibagi menjadi dua prosedur baru yaitu prosedur untuk menaikkan *baud rate* dan prosedur untuk menurunkan *baud rate*.

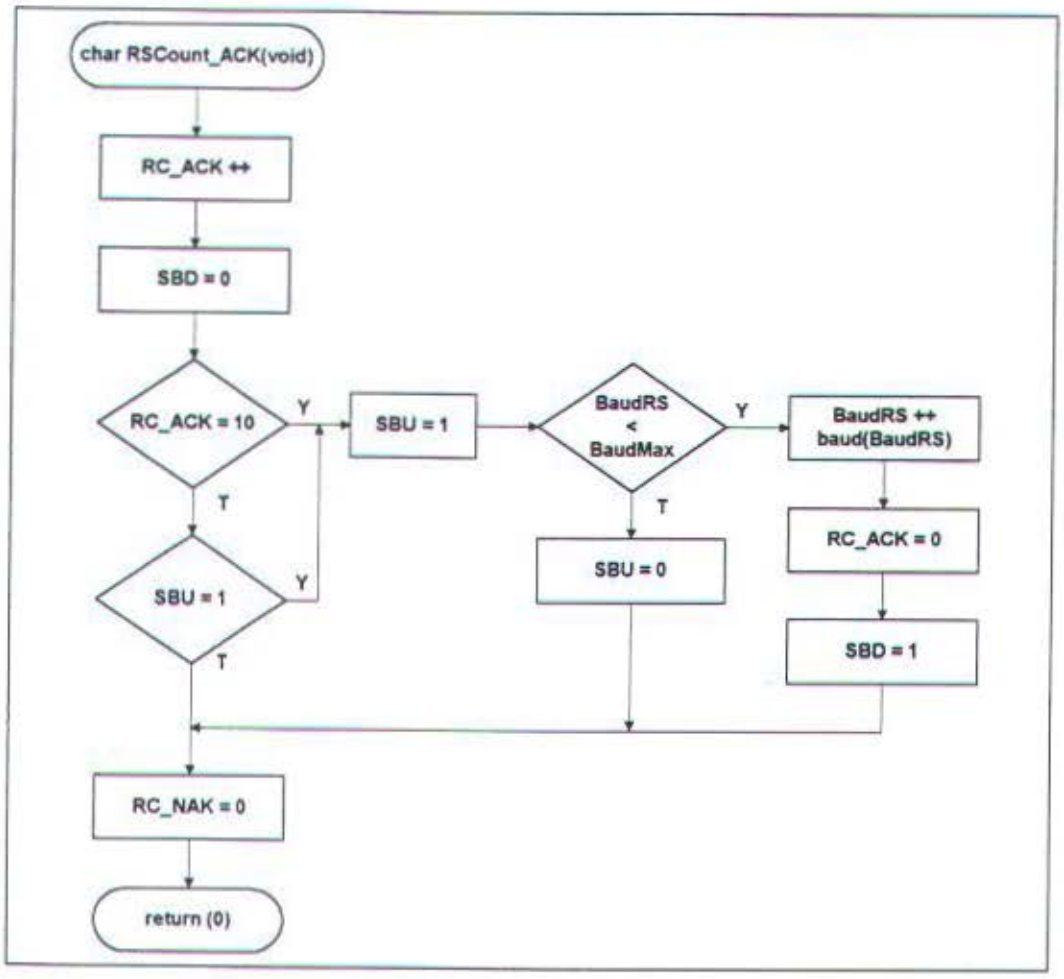
➢ **Prosedur untuk menaikkan baud rate**

Prosedur ini dijalankan bila kondisi paket data yang diterima tidak terjadi kesalahan. Diagram alir dari prosedur ini dapat dilihat pada gambar 3.5. Secara umum *baud rate* dinaikkan jika sepuluh paket data secara berurutan diterima dengan baik oleh penerima. Namun adakalanya *baud rate* dinaikkan tanpa harus menunggu sepuluh paket data secara berurutan diterima dengan benar. Hal ini terjadi bila :

- *Baud rate* mengalami kenaikkan karena sepuluh paket data secara berurutan diterima secara benar.

Listing program dari prosedur pengontrol kenaikan *baud rate* dapat dilihat pada uraian berikut ini :

```
char RSCount_ACK(void)
{
  RC_ACK ++;
  SBD = 0;
  if ( (RC_ACK == 10) || (SBU == 1) )
  {
    SBU = 1;
    if (BaudRS < BaudMaxRS)
    {
      BaudRS++;
      baud(BaudRS);
      RC_ACK = 0;
      SBD = 1;
    }
    else
    {
      SBU = 0;
    }
  }
  RC_NAK = 0;
  return(0);
}
```



GAMBAR 3.5
PROSEDUR PENGONTROL KENAIKAN BAUD RATE

➢ **Prosedur untuk menurunkan baud rate**

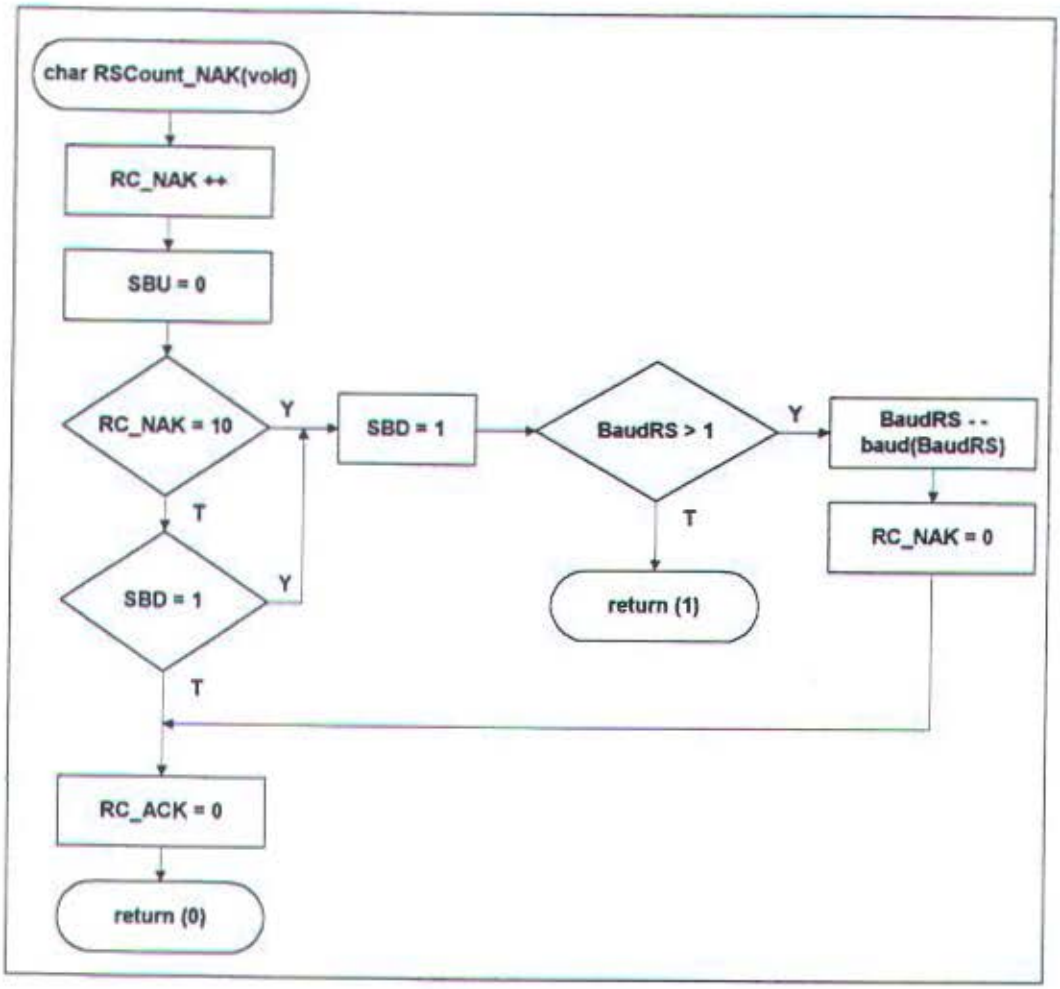
Prosedur ini dijalankan bila kondisi paket data yang diterima terjadi kesalahan. Diagram alir dari prosedur ini dapat dilihat pada gambar 3.6. Secara umum *baud rate* diturunkan jika sepuluh paket data secara berurutan diterima dengan kesalahan oleh penerima. Namun adakalanya *baud rate* diturunkan tanpa harus menunggu sepuluh paket data secara berurutan diterima dengan kesalahan. Hal ini terjadi bila :

- *Baud rate* baru mengalami kenaikan dan terjadi kesalahan penerimaan paket data berikutnya.
- *Baud rate* mengalami penurunan karena sepuluh paket data secara berurutan diterima dengan kesalahan.

Bila penerimaan paket data terus menerus mengalami kesalahan, prosedur ini akan memberi nilai balik '1' (pengiriman tidak berhasil).

Listing program dari prosedur pengontrol penurunan *baud rate* dapat dilihat pada uraian berikut ini :

```
char RSCount_NAK(void)
{
  RC_NAK ++;
  SBU = 0;
  if ( (RC_NAK == 10) || (SBD == 1) )
  {
    SBD = 1;
    if (BaudRS > 1)
    {
      BaudRS--;
      baud(BaudRS);
      RC_NAK = 0;
    }
    else
    return(1);
  }
  RC_ACK = 0;
  return(0);
}
```



GAMBAR 3.6
PROSEDUR PENGONTROL PENURUNAN BAUD RATE

III.2.2 Pemrograman Modem

Dalam pemrograman modem, beberapa hal yang perlu diperhatikan adalah:

- **Pengiriman Perintah AT**

Perintah AT merupakan sederetan karakter ASCII yang diawali oleh karakter AT dan diakhiri oleh karakter ENTER (13). *Listing program* prosedur pengiriman perintah AT dapat dilihat pada uraian berikut ini:

```
void SendCommand(char command[])
{
  char a;
  for (a=0; a<strlen(command); a++)
  {
    SendChar(command[a]);
  }
  SendChar(13);
}
```

➢ **Proses Pembangunan Hubungan Antar Modem**

Ada beberapa motode untuk membangun hubungan antar modem. Salah satu metode tersebut yaitu metode pembangunan hubungan dengan menginisialisasi *answer mode* dan *originate mode*. *Originate mode* berlangsung pada modem pengirim dan *answer mode* pada modem penerima. *Automatic Originate Mode* diinisialisasi dengan mengirimkan perintah ATD<nomer telepon>, sedangkan *automatic answer mode* disiapkan dengan menginisialisasi register S0 dengan suatu bilangan bukan nol. Apabila sinyal *ringing* terdeteksi dan nilai dari register S0 bukan nol, modem akan *off hook* dan akan mulai membangun hubungan dengan modem pengirim.

➢ **Proses Pengubahan Kelajuan Transfer Data Modem**

Proses pengubahan kelajuan transfer data (*baud rate*) modem diawali dengan pengubahan *baud rate* DTE (UART). Kemudian UART mengirim perintah AT<CR> atau perintah AT lainnya. Setelah UART menerima respon <OK> dari modem, *baud rate* modem telah berubah sesuai

dengan *baud rate* UART.

- **Proses Pemutusan Hubungan Antar Modem**

  Untuk memutuskan hubungan antar modem, masing-masing modem harus dalam keadaan *hang up*. Keadaan ini dimulai dengan pengiriman karakter *escape* oleh UART ke modem dilanjutkan dengan pengiriman perintah ATH.

BAB IV
PENGUJIAN PROGRAM

Pada bab III telah dibahas tentang perencanaan dan pembuatan program pengontrol *baud rate*. Untuk mengetahui kemampuan program tersebut diperlukan suatu uji coba. Pada bab ini akan diuraikan tentang uji coba program yang dilakukan. Uji coba akan dilakukan dengan menggunakan saluran RS-232 (*null modem*) dalam dua kondisi yang berlainan, yaitu kondisi saluran RS-232 tanpa gangguan dan kondisi saluran RS-232 dengan gangguan. Berikut ini uraian uji coba dari kedua kondisi saluran tersebut.

IV.1 Saluran RS-232 Tanpa Gangguan

Yang dimaksud dengan saluran RS-232 tanpa gangguan dalam tugas akhir ini adalah suatu saluran RS-232 yang mempunyai keadaan sebagai berikut :

- Panjang saluran RS-232 tidak melebihi *50 feet*.
- Saluran RS-232 tidak dipengaruhi *noise*.
- Saluran RS-232 tidak berada di dalam pengaruh medan magnet maupun medan listrik (pengaruh dari kedua medan tersebut bisa diabaikan).

Secara teoritis, jika pengiriman data dilakukan dengan menggunakan saluran RS-232 dengan kondisi seperti ini maka kelajuan transfer data dapat berlangsung dengan kelajuan maksimum terkecil.

Pada uji coba yang dilakukan, kelajuan transfer data dapat berlangsung dalam kelajuan maksimum terkecil.

IV.2 Saluran RS-232 Dengan Gangguan

Yang dimaksud dengan saluran RS-232 dengan gangguan dalam tugas akhir ini adalah suatu saluran RS-232 yang mempunyai keadaan sebagai berikut :

- Panjang saluran RS-232 tidak melebihi *50 feet*.
- Saluran RS-232 tidak berada di dalam pengaruh medan magnet maupun medan listrik (pengaruh dari kedua medan tersebut bisa diabaikan).
- Saluran RS-232 dipengaruhi oleh *noise* yang berasal dari *white noise generator*.

Peralatan-peralatan yang digunakan dalam uji coba adalah sebagai berikut:

- 2 (dua) set Komputer PC AT 486.
- 1 (satu) *Synchroscope* merk TELI dengan tipe TS10067.
- 1 (satu) *White Noise Generator* merk ANDO dengan tipe GRN-3DB.

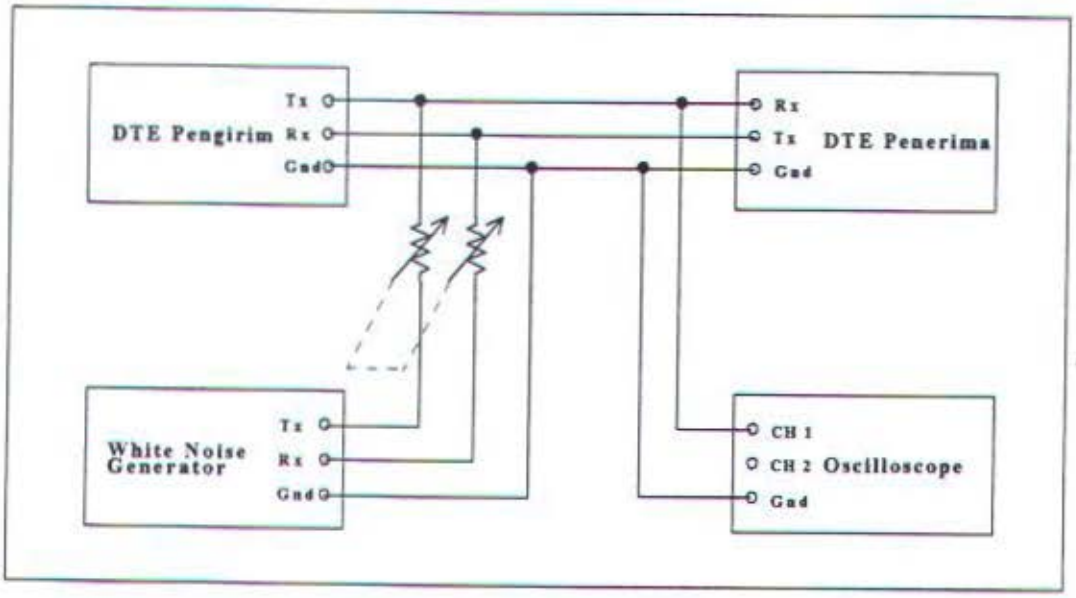
  White noise generator ini memiliki karakteristik listrik sebagai berikut:

  - Level output : -15 dBm sampai +15 dBm
  - Deviasi spektrum kontinu : ± 1 dB pada 100 Hz sampai 10 kHz
    ± 2 dB pada 30 Hz sampai 20 kHz
  - Impedansi output : 600 Ω
  - Jangkauan temperatur : 0 °C sampai 40 °C
  - Keperluan listrik : AC 220V ± 10%, 50/60 Hz
  - Konsumsi daya : 5 VA

Instalasi peralatan untuk melakukan uji coba dapat dilihat pada gambar 4.1. Fungsi *variabel resistor* pada instalasi tersebut yaitu untuk memperbesar impedansi output *white noise generator* dan mengatur level output *noise* yang diinginkan.

Berikut ini uraian mengenai langkah-langkah uji coba program dengan menggunakan instalasi peralatan seperti gambar 4.1 :

- Merangkai peralatan seperti gambar 4.1.
- Mengatur *variabel resistor* pada kedudukan maksimum.
- Menjalankan program pengatur *baud rate* pada masing-masing komputer (DTE).
- Mengukur tegangan *peak to peak* sinyal pada input komputer penerima. Level tegangan ini merupakan level sinyal dari DTE.
- Mengatur *variabel resistor* secara perlahan-lahan hingga komunikasi data antar DTE terputus.
- Mengukur tegangan *peak to peak noise* saat komunikasi data terputus dengan menggunakan *synchroscope*. Level *noise* ini merupakan level *noise* kritis, bila level output *noise* lebih besar atau sama dengan level *noise* kritis komunikasi antar DTE tidak dapat berlangsung, sedangkan bila level output *noise* lebih kecil dari level *noise* kritis komunikasi dapat berlangsung meskipun dengan kelajuan transfer data yang berbeda.



GAMBAR 4.1
INSTALASI PERALATAN UNTUK UJI COBA PROGRAM

Dari pengukuran yang dilakukan diperoleh data sebagai berikut:

- Harga level *noise* kritis : lebih kurang 4 V
- Harga level sinyal dari DTE : 20 V

Dari data pengukuran di atas dapat diperoleh harga S/N_{kritis} sebesar 13,98 dB.

BAB V
PENUTUP

V.1 Kesimpulan

- Dengan memanfaatkan *error checking*, kelajuan transfer data dapat diatur hingga mencapai kelajuan transfer data optimum.
- Kelajuan transfer data modem (*Data Communication Equipment*) dapat diatur dengan mengubah-ubah kelajuan transfer data UART (*Data Terminal Equipment*).
- Pengontrolan kelajuan transfer data secara *software* ternyata tidak dapat digunakan secara langsung pada *smart modem*. Hal ini karena terjadi konflik antara protokol modem dengan protokol program.
- Apabila saluran yang digunakan cukup baik, seperti saluran RS-232 tanpa gangguan, kelajuan transfer data dapat berlangsung dengan kelajuan maksimum terkecil.
- Apabila pada saluran terdapat *noise* maka kelajuan transfer data akan berubah sesuai level *noise* yang diinjeksikan atau komunikasi data tidak dapat berlangsung. Jadi terdapat level *noise* yang menyebabkan komunikasi data tidak dapat berlangsung. Level *noise* ini, dalam tugas akhir ini, disebut level *noise* kritis.

V.2 Saran

Untuk meningkatkan kualitas program dapat ditambahkan fasilitas program pengontrol kecepatan *real modem*. Program-program yang dapat ditambahkan antara lain :

- Program untuk menon-aktifkan protokol modem yang terdapat pada ROM modem.
- Program *dialing* yang berguna untuk membangun hubungan antar modem.

DAFTAR PUSTAKA

1. Abdul Kadir, ***Pemrograman C++***, Cetakan Pertama, Andi Offset, Yogyakarta, 1995.

2. H.M. Deitel / P.J. Deitel, ***C++ How To Program***, Prentice Hall, New Jersey, 1994.

3. Hartono Partoharsodjo, ***Bahasa Assembly***, Cetakan Kelima, PT. Elex Media Komputindo, Jakarta, 1995.

4. Hartono Partoharsodjo, ***Contoh Penggunaan Fungsi-fungsi Turbo C***, PT. Elex Media Komputindo, Jakarta, 1993.

5. Joe Campbell, ***C Programmer's Guide to Serial Communication***, First Edition, Howard W. Sams & Company, Division of Macmillan, Inc., Indianapolis, 1987.

6. Joe Campbell, ***C Programmer's Guide to Serial Communication***, Second Edition, Sams Publishing, Division of Prentice Hall, Indianapolis, 1994.

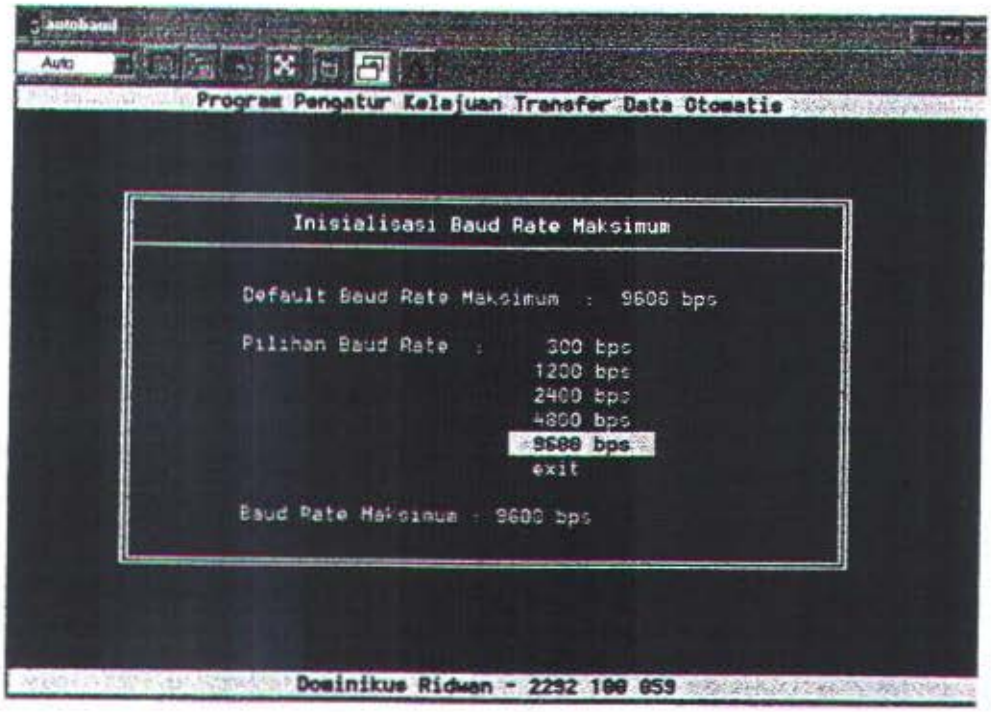
7. Les Freed dan Frank J. Derfler, Jr., ***Panduan Komunikasi Modem***, Cetakan Kedua, PT. Elex Media Komputindo, Jakarta, 1996.

8. Timothy S. Monk, ***Windows Programmer's Guide to Serial Communication***, First Edition, Sams Publishing, Division of Prentice Hall, Indianapolis, 1992.

9. _____, ***Modem Instruction Manual***.

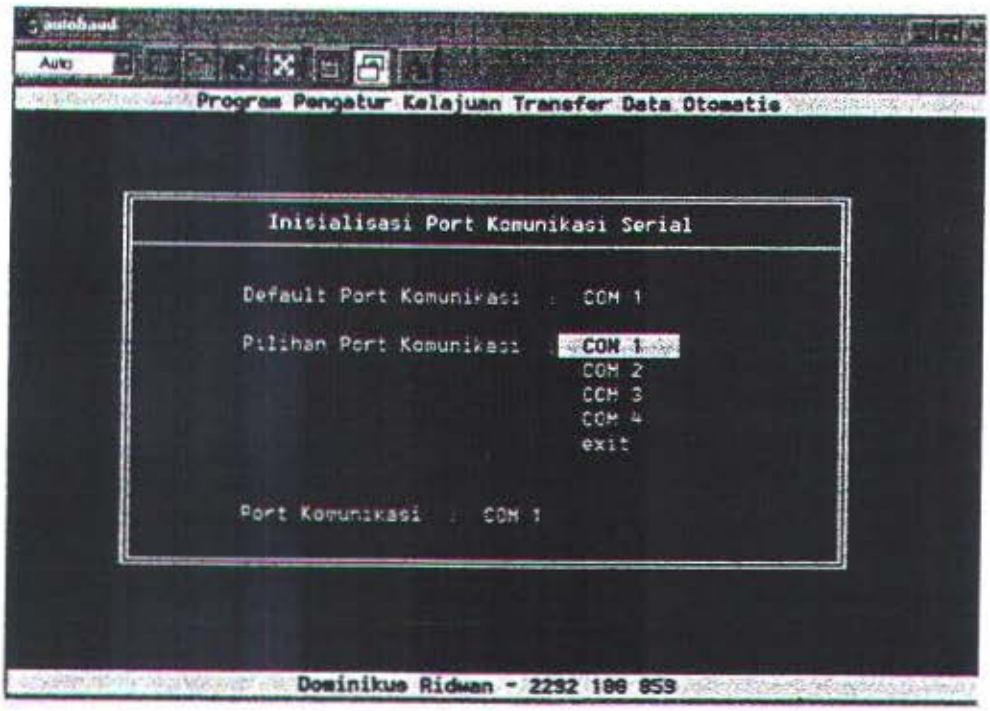
LAMPIRAN A

INISIALISASI BAUD RATE MAKSIMUM



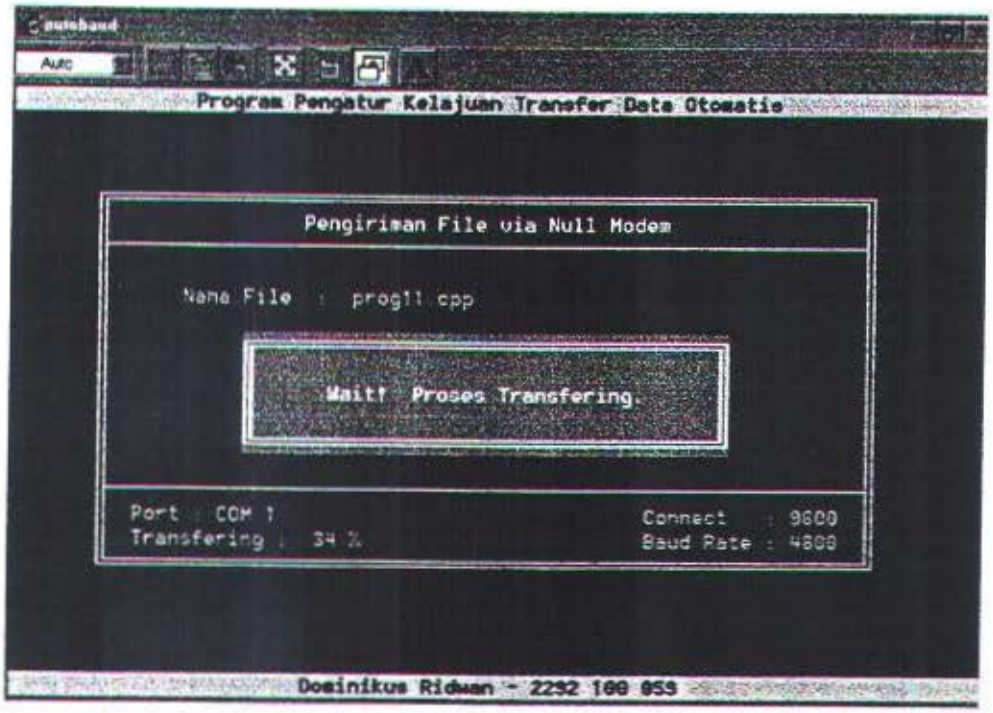
LAMPIRAN B

INISIALISASI PORT KOMUNIKASI



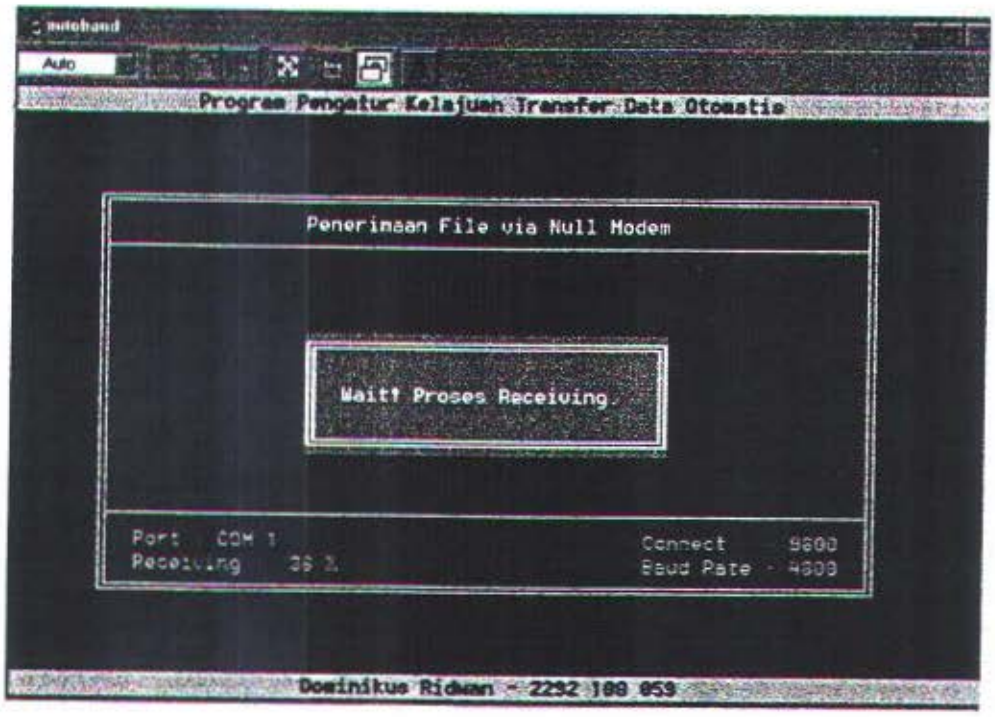
LAMPIRAN C

PROSES PENGIRIMAN DATA



LAMPIRAN D

PROSES PENERIMAAN DATA



LAMPIRAN E

LISTING PROGRAM KOMDAT.H

```
#include <dos.h>

/* Variabel-variabel global */

int  Rx,                    /* Alamat Rx Buffer */
     Tx,                    /* Alamat Tx Buffer */
     LOBaud,                /* Alamat untuk High Order Baud */
     HIBaud,                /* Alamat untuk Low Order Baud */
     IER,                   /* Interrupt Enable Register */
     IIR,                   /* Interrupt Identification Register */
     LCR,                   /* Line Control Register */
     MCR,                   /* Modem Control Register */
     LSR,                   /* Line Status Register */
     MSR;                   /* Modem Status Register */
int  NoCom;                 /* Nomer Port Komunikasi */
int  HiRate,                /* Inisialisasi High Order Baud */
     LoRate;                /* Inisialisasi Low Order Baud */
int  RBaudPrint;            /* Variabel Baud Rate RS untuk ditampilkan */
int  RMaxPrint;             /* Variabel Baud Rate Max RS untuk ditampilkan */
int  BaudMaxRS,             /* Variabel Baud Max via RS232 */
     BaudMax;               /* Variabel Baud Max Komunikasi Data */
char BaudRS;                /* Variabel Baud Rate via RS232 */
char DelayBaud[5] = {400,600,800,1000,1200};
char RC_NAK,                /* Variabel Counter NAK untuk RS */
     RC_ACK;                /* Variabel Counter ACK untuk RS */
char SBU,                   /* Variabel Switch Baud Up */
     SBD;                   /* Variabel Switch Baud Down */
unsigned char tpacket[131]; /* Variabel array Data Paket Kirim */
unsigned char rpacket[131]; /* Variabel array Data Paket Terima */
unsigned char thpacket[10]; /* Variabel array Data Paket Handshaking Kirim */
unsigned char rhpacket[10]; /* Variabel array Data Paket Handshaking Terima */

/* Fungsi-fungsi yang digunakan */
void initport()
{
 int addcom;

 Rx      = 0X2E8;
 Tx      = 0X2E8;
 LOBaud  = 0X2E8;
 HIBaud  = 0X2E9;
 IER     = 0X2E9;
 IIR     = 0X2EA;
 LCR     = 0X2EB;
 MCR     = 0X2EC;
 LSR     = 0X2ED;
 MSR     = 0X2EE;

 switch (NoCom)
 {
  case 0 : addcom  =  0X110;   /* Alamat Memory COM 1 --> 3F8 */
                      break;
  case 1 : addcom  =  0X010;   /* Alamat Memory COM 2 --> 2F8 */
                      break;
  case 2 : addcom  =  0X100;   /* Alamat Memory COM 3 --> 3E8 */
                      break;
  case 3 : addcom  =  0X000;   /* Alamat Memory COM 4 --> 2E8 */
                      break;
 }
```

```
  Rx      = Rx + addcom;
  Tx      = Tx + addcom;
  LOBaud  = LOBaud + addcom;
  HIBaud  = HIBaud + addcom;
  IER     = IER + addcom;
  IIR     = IIR + addcom;
  LCR     = LCR + addcom;
  MCR     = MCR + addcom;
  LSR     = LSR + addcom;
  MSR     = MSR + addcom;

  outportb(LCR,0x03);       /* parity disable, 1 stop bit, 8 bit data */
  outportb(IER,0x00);       /* interupt disable */
 }

 void baud(char rate)
 {
  switch (rate)
  {
   case 1 : HiRate  = 0x01;
                      LoRate = 0x80;
                      RBaudPrint = 300;
                      break;
   case 2 : HiRate  = 0x00;
                      LoRate = 0x60;
                      RBaudPrint = 1200;
                      break;
   case 3 : HiRate  = 0x00;
                      LoRate = 0x30;
                      RBaudPrint = 2400;
                      break;
   case 4 : HiRate  = 0x00;
                      LoRate = 0x18;
                      RBaudPrint = 4800;
                      break;
   case 5 : HiRate  = 0x00;
                      LoRate = 0x0c;
                      RBaudPrint = 9600;
                      break;
  }
  outportb(LCR,0x80);           /* mengaktifkan perubahan baud */
  outportb(LOBaud,LoRate);
  outportb(HIBaud,HiRate);
  outportb(LCR,0x03);           /* parity disable, 1 stop bit, 8 bit data */
 }

 void SendChar(unsigned char tx)
 {
  while((inportb(LSR)&0x60) != 0x60);
  outportb(Tx,tx);
 }

 void SendPacket()
 {
  unsigned char j;
  for (j=0;j<131;j++)
  {
   SendChar(tpacket[j]);
   delay(1);
  }
 }

 void MakePacket(unsigned char datafile,unsigned char mcbyte,char kode)
 {
  if (kode == 0)
  {
   tpacket[130] = tpacket[130] + datafile;     /* checksum value */
   tpacket[mcbyte+2] = datafile;
  }
  else
```

```
  {
   unsigned char j;
   for (j=mcbyte-2;j<130;j++)
   {
    tpacket[j] = 0;                          /* checksum value tidak berubah */
    if (j == (mcbyte-2))
    {
     if (kode == 1)
     {
      tpacket[j] = 4;    /* Kode ASCII ETX */
      tpacket[130]=tpacket[130]+tpacket[j];
     }
    }
   }
  }
}

unsigned char ReceiveChar(void)
{
 while((inportb(LSR) & 0x01) != 0x01);
 return(inportb(Rx));
}

void ReceivePacket()
{
 unsigned char jk;
 for (jk=0;jk<131;jk++)  { rpacket[jk]=ReceiveChar(); }
}

char RSCount_NAK(void)
{
 RC_NAK++;
 SBU = 0;
 if ( (RC_NAK > 9) || (SBD==1) )
 {
  SBD = 0;
  RC_NAK = 0;
  if (BaudRS > 1)
  { delay(DelayBaud[BaudRS-1]);
    BaudRS--;
    baud(BaudRS);
  }
  else
  { return(1); }
 }
 RC_ACK=0;
 return(0);
}

char RSCount_ACK(void)
{
 RC_ACK++;
 SBD = 0;
 if ( (RC_ACK > 9) || (SBU == 1) )
 {
  RC_ACK = 0;
  SBU = 1;
  if (BaudRS < BaudMax)
  {
   delay(DelayBaud[BaudRS-1]);
   BaudRS++;
   baud(BaudRS);
   SBD = 1;
  }
  else
  { SBU = 0; }
 }
 RC_NAK = 0;
 return(0);
}
```

LAMPIRAN F

LISTING PROGRAM AUTOBAUD.CPP

```
#include "mymenu.h"
#include "komdat1.h"
#include <iomanip.h>

void MenuBaudMax()
{
 char pilihan=1;
 box(10,5,16,60,0x17,"",0x17,1);
 writecenter(10,70,6,"Inisialisasi Baud Rate Maksimum",0x1f);
 fillchar(10,7,10,7,"",0x17);
 fillchar(11,7,68,7,"-",0x17);
 fillchar(69,7,69,7,"",0x17);
 gotoxy(20,9); cprintf("Default Baud Rate Maksimum  :  9600 bps");
 gotoxy(20,11);cprintf("Pilihan Baud Rate  : ");
 do
 {
  textattr(0x17);
  gotoxy(20,18); cprintf("                              ");
  gotoxy(20,18); cprintf("Baud Rate Maksimum : ");
                 cout << setw(4) << RMaxPrint << " bps";
  menu[0] = "   300 bps  ";
  menu[1] = "  1200 bps  ";
  menu[2] = "  2400 bps  ";
  menu[3] = "  4800 bps  ";
  menu[4] = "  9600 bps  ";
  menu[5] = " exit      ";
  pilihan = winmenu(1,42,10,6,1,7,pilihan,menu);
  switch(pilihan)
  {
   case 1 : BaudMaxRS =  1;                  /* Baud Max RS =  300 */
                         RMaxPrint = 300;
                         break;
   case 2 : BaudMaxRS =  2;                  /* Baud Max RS = 1200 */
                         RMaxPrint = 1200;
                         break;
   case 3 : BaudMaxRS =  3;                  /* Baud Max RS = 2400 */
                         RMaxPrint = 2400;
                         break;
   case 4 : BaudMaxRS =  4;                  /* Baud Max RS = 4800 */
                         RMaxPrint = 4800;
                         break;
   case 5 : BaudMaxRS =  5;                  /* Baud Max RS = 9600 */
                         RMaxPrint = 9600;
                         break;
  }
 }
 while(pilihan != 6);
}

void MenuPortCom()
{
 char pilihan=1;
 box(10,5,16,60,0x17,"",0x17,1);
 writecenter(10,70,6,"Inisialisasi Port Komunikasi Serial ",0x1f);
 fillchar(10,7,10,7,"",0x17);
 fillchar(11,7,68,7,"-",0x17);
 fillchar(69,7,69,7,"",0x17);
 gotoxy(20,9); cprintf("Default Port Komunikasi  :  COM 1");
 gotoxy(20,11);cprintf("Pilihan Port Komunikasi  : ");
```

```
  do
  {
   textattr(0x17);
   gotoxy(20,18);  cprintf("                    ");
   gotoxy(20,18);  cprintf("Port Komunikasi : COM ");
                   cout << setw(1) << NoCom+1;
   menu[0] = "  COM 1  ";
   menu[1] = "  COM 2  ";
   menu[2] = "  COM 3  ";
   menu[3] = "  COM 4  ";
   menu[4] = "  exit   ";
   pilihan = winmenu(1,46,10,5,1,7,pilihan,menu);
   switch(pilihan)
   {
    case 1 : NoCom  = 0;
             break;
    case 2 : NoCom  = 1;
             break;
    case 3 : NoCom  = 2;
             break;
    case 4 : NoCom  = 3;
             break;
   }
  }
  while(pilihan != 5);
  initport();
}

void MenuKirimRS()
{
 FILE *ptr_file;
 unsigned char namafile[58];
 long filen;
 float filen1, transfer;
 int k, handle, proses;

 SBU = 1;
 SBD = 0;
 baud(1);
 Mbaud(1);
 BaudRS = 1;
 BaudModem = 1;

 loadscr(scr[1]);
 box(8,5,16,64,0x17,"",0x17,1);
 writecenter(10,70,6,"Pengiriman File via Null Modem",0x1f);
 fillchar(8,7,8,7,"",0x17);
 fillchar(9,7,70,7,"-",0x17);
 fillchar(71,7,71,7,"",0x17);
 savescr(scr[3]);
 _setcursortype(_NORMALCURSOR);
 gotoxy(15,9); cprintf("Nama File  : ");cin >> namafile;
 _setcursortype(_NOCURSOR);
 loadscr(scr[3]);
 gotoxy(15,9); cprintf("Nama File  : ");cout << namafile;

 if ((ptr_file=fopen(namafile,"rb")) == NULL)
 {
  box(25,11,5,30,0x3f,"",0x3f,0);
  writecenter(25,55,13,"File Tidak Ditemukan !",0x3f);
  getch();
 }
 else
 {
  unsigned char cbyte, nopacket;
  char tanggapan, kodeproses, kodeHS;

  for (k=0;k<9;k++)
  { thpacket[k]=BaudMaxRS; }
  thpacket[9]=8*BaudMaxRS;
```

```
fillchar(8,17,8,17,"|",0x17);
fillchar(9,17,70,17,"-",0x17);
fillchar(71,17,71,17,"|",0x17);
gotoxy(11,18); cprintf("Port : COM ");cout << NoCom+1;
gotoxy(53,18); cprintf("Connect  : ");
gotoxy(11,19); cprintf("Transfering :   %");
gotoxy(53,19); cprintf("Baud Rate : ");
savescr(scr[9]);
cout << setw(4) << RBaudPrint;
box(20,11,5,40,0x3f,"",0x3f,0);
writecenter(25,55,13,"Wait!  Proses Connecting 10 detik.",0x3f);

/* Menunggu Tanggapan dari Penerima Selama 10 detik */
SendChar(67);    /* Mengirim Karakter 'C' ke Penerima */
tanggapan = 0;
do
{
 delay(1000);
 /* Mengirimkan Kode Panggilan Ke Penerima */
 tanggapan++;
 gotoxy(38,14);printf("( %d )",10-tanggapan);
 sound(4500);
 if (tanggapan==10)
 { delay(700);}
 else
 { delay(50);}
 nosound();
}
while( ((inportb(LSR) & 0x01) != 0x01) && (tanggapan < 10) );

kodeproses = 0;
if (tanggapan < 10)       /* Ada Tanggapan dari Penerima, Komunikasi Data Dapat Dilakukan */
{
 inportb(Rx);    /* Mengosongkan RX Buffer */

 /* ---- PROSES HANDSHAKING 1 ---- */

 unsigned char checkHS;
 loadscr(scr[9]);
 box(25,11,5,30,0x3f,"",0x3f,0);
 writecenter(25,55,13,"Wait! Proses Handshaking 1.",0x3f);
 do
 {  /* Mengirim Paket Handshaking Baud Rate Maksimum */
    textattr(0x17);
    gotoxy(53,19); cprintf("Baud Rate : ");cout << setw(4) << RBaudPrint;
    for(k=0;k<10;k++)
    {
     SendChar(thpacket[k]);
     delay(1);
    }
    /* Menunggu Paket Handshaking Baud Rate Maksimum */
    for(k=0;k<10;k++)
    { rhpacket[k]=ReceiveChar(); }
    checkHS=0;
    for(k=1;k<9;k++)
    { checkHS=checkHS+rhpacket[k]; }
    kodeHS = 0;
    if (checkHS == rhpacket[9])
    { /* Data Paket Benar */
     BaudMax = rhpacket[1];
     kodeHS = 1;
    }
    else
    { /* Data Paket Salah */
     if (RSCount_NAK() == 1)
     { kodeHS = 2; }
     delay(100+DelayBaud[BaudRS-1]);
    }
 }
 while(kodeHS == 0);
```

```
if(kodeHS == 1)
{  /* ---- PROSES HANDSHAKING 2 ---- */

   loadscr(scr[9]);
   box(25,11,5,30,0x3f,"",0x3f,0);
   writecenter(25,55,13,"Wait! Proses Handshaking 2.",0x3f);
   RC_ACK=0;
   RC_NAK=0;
   SBU=0;
   SBD=0;
   BaudRS=BaudMax;
   delay(1500);
   baud(BaudRS);
   for (k=0;k<9;k++)
   { thpacket[k]=BaudMax; }
  thpacket[9]=BaudMax*8;
   do
   {
    textattr(0x17);
    gotoxy(53,19); cprintf("Baud Rate : ");cout << setw(4) << RBaudPrint;
    for(k=0;k<10;k++)
    {
     SendChar(thpacket[k]);
     delay(1);
    }
    for(k=0;k<10;k++)
    { rhpacket[k]=ReceiveChar(); }
    checkHS=0;
    for(k=1;k<9;k++)
    { checkHS=checkHS+rhpacket[k]; }
    kodeHS = 0;
    if (checkHS == rhpacket[9])
    { /* Data Paket Benar */
     if (RC_ACK == 9)
     { kodeHS = 1; }
     else
     { RSCount_ACK(); }
    }
    else
    { /* Data Paket Salah */
     if (RSCount_NAK() == 1)
     {
      kodeproses = 0;
      goto batal;
     }
     delay(100+DelayBaud[BaudRS-1]);
    }
}
while(kodeHS == 0);
SBU=0;
SBD=0;
RC_ACK=0;
RC_NAK=0;

/* ---- PROSES TRANSFER DATA ---- */

loadscr(scr[9]);
box(25,11,5,30,0x3f,"",0x3f,0);
writecenter(25,55,13,"Wait! Proses Transfering.",0x3f);
textattr(0x17);
gotoxy(53,18); cprintf("Connect   : ");cout << setw(4) << RBaudPrint;
gotoxy(53,19); cprintf("Baud Rate : ");cout << setw(4) << RBaudPrint;
delay(200);

/* Inisialisasi Variabel Untuk Paket Data */

tpacket[130] = 0;
tpacket[0] = 1;
nopacket = 1;
```

```
/* Membuat Paket Data Untuk Nama File, Panjang File */

unsigned char n;
char *string, des;
int desimal,tanda;

handle = fileno(ptr_file);
filen = filelength(handle);
string = ecvt(filen,10,&desimal,&tanda);
des = desimal;
filen1=filen;
for (n=0;n<strlen(namafile);n++)
{ MakePacket(namafile[n],n,0); }
MakePacket(4,n,0);

n++;
for (k=0; k<10; k++)
{ MakePacket(*(string+k),n+k,0); }
MakePacket(des,n+k,0);
MakePacket(0,n+k+1,1);
tpacket[1] = nopacket;

do
{
 SendPacket();
 tanggapan = ReceiveChar();
 if (tanggapan == 0x06)
 { RSCount_ACK(); }
 else
 {
  if (RSCount_NAK() == 1)
  {
   kodeproses = 0;
   goto batal;
  }
  delay(100-DelayBaud[BaudRS-1]);
 }
 textattr(0x17);
 gotoxy(53,19); cprintf("Baud Rate : ");cout << setw(4) << RBaudPrint;
}
while(tanggapan != 0x06);
proses = 0;
transfer = 0;
gotoxy(11,19); cprintf("Transfering : %3d %",proses);

/* Membuat Paket Data Untuk Data File */

nopacket++;
cbyte = 0;
tpacket[130]=0;
while ((k=getc(ptr_file)) != EOF)
{
 MakePacket(k,cbyte,0);
 cbyte++;
 if (cbyte > 127)
 {
  cbyte = 0;
  tpacket[1] = nopacket;
  nopacket++;
  do
  {
   SendPacket();
   tanggapan = ReceiveChar();
   if (tanggapan == 0x06)
   { RSCount_ACK(); }
   else
   {
     if (RSCount_NAK() == 1)
     {
      kodeproses = 0;
```

```
            goto batal;
           }
           delay(100+DelayBaud[BaudRS-1]);
          }
          textattr(0x17);
          gotoxy(53,19); cprintf("Baud Rate : ");cout << setw(4) << RBaudPrint;
         }
         while(tanggapan != 0x06);
         textattr(0x17);
         transfer = transfer + 128;
         proses = transfer / filen1 * 100;
         gotoxy(11,19); cprintf("Transfering : %3d %",proses);
         tpacket[130] = 0;
        }
       }

       /* Membuat Paket Data Terakhir */

       MakePacket(k,cbyte,1);
       tpacket[1] = nopacket;
       do
       {
        SendPacket();
        tanggapan = ReceiveChar();
        if (tanggapan == 0x06)
        { RSCount_ACK(); }
        else
        {
         if (RSCount_NAK() == 1)
         {
          kodeproses = 0;
          goto batal;
         }
         delay(100+DelayBaud[BaudRS-1]);
        }
        textattr(0x17);
        gotoxy(53,19); cprintf("Baud Rate : ");cout << setw(4) << RBaudPrint;
       }
       while(tanggapan != 0x06);
       textattr(0x17);
       transfer = transfer + 128;
       proses = transfer / filen1 * 100;
       gotoxy(11,19); cprintf("Transfering : %3d % terakhir",proses);

       do
       {
        SendChar(0x04);
        tanggapan = ReceiveChar();
        if (tanggapan == 0x06)
        { RSCount_ACK(); }
        else
        {
         if (RSCount_NAK() == 1)
         {
          kodeproses = 0;
          goto batal;
         }
         delay(100+DelayBaud[BaudRS-1]);
        }
        textattr(0x17);
        gotoxy(53,19); cprintf("Baud Rate : ");cout << setw(4) << RBaudPrint;
       }
       while(tanggapan!=0x06);
       kodeproses = 1;
       loadscr(scr[9]);
       box(25,11,5,30,0x3f,"",0x3f,0);
       writecenter(25,55,13,"Komunikasi Berhasil.",0x3f);
       getch();
      }
     }
```

```
batal:
   if (kodeproses == 0)
   { /* Penerima Tidak Dapat Dihubungi */
    loadscr(scr[9]);
    box(20,12,3,40,0x3f,"",0x3f,0);
    gotoxy(25,13);cout << "Komunikasi Gagal (Disconnect)!";
    getch();
   }
   fclose(ptr_file);
  }
}

void MenuKirimModem()
{ getch(); }

void MenuTerimaRS()
{
 FILE *ptr_file;
 unsigned char namafile[13],
               checkHS,
               cbyte,
               counter,
               nopacketRx,
               checksumRx;
 char tanggapan,
      kodeHS;
 long filen,
      receive;
 float filen1,
      receive1;
 int proses;

 baud(1);
 Mbaud(1);
 BaudRS = 1;
 BaudModem = 1;
 SBU = 1;
 SBD = 0;

 inportb(Rx);
 loadscr(scr[1]);
 box(8,5,17,64,0x17,"",0x17,1);
 writecenter(10,70,6,"Penerimaan File via Null Modem",0x1f);
 fillchar(8,7,8,7,"",0x17);
 fillchar(9,7,70,7,"-",0x17);
 fillchar(71,7,71,7,"",0x17);
 savescr(scr[3]);
 fillchar(8,18,8,18,"",0x17);
 fillchar(9,18,70,18,"-",0x17);
 fillchar(71,18,71,18,"",0x17);
 gotoxy(11,19); cprintf("Port : COM ");cout << NoCom+1;
 gotoxy(53,19); cprintf("Connect  : ");
 gotoxy(11,20); cprintf("Receiving :    %");
 gotoxy(53,20); cprintf("Baud Rate : ");
 savescr(scr[9]);
 cout << setw(4) << RBaudPrint;
 box(20,11,6,40,0x3f,"",0x3f,0);
 writecenter(25,55,13,"Mode Terima Diaktifkan!",0x3f);
 writecenter(25,55,14,"Tekan Sembarang Tombol --> QUIT",0x3f);

 /* Menunggu Panggilan dari Pengirim Hingga Penekanan Tombol [ESC]
   atau Disebut Mode Terima */

 tanggapan = 0;
 do
 {
  if ((inportb(LSR) & 0x01) == 0x01)
  {
   cbyte = inportb(Rx);
   printf("%d",cbyte);
```

```
   tanggapan = 1;
  }
  if (kbhit())
  { tanggapan =2; }
 }
 while(tanggapan == 0);

 if (tanggapan != 2) /* Mode Terima Tetap Aktif */
 { /* Ada Panggilan Dari Pengirim Komunikasi Mulai dilakukan */

  /* ----- PROSES HANDSHAKING 1 ----- */

  /* Memberikan Tanggapan Atas Panggilan Pengirim */

  loadscr(scr[9]);
  box(25,11,5,30,0x3f,"",0x3f,0);
  writecenter(25,55,13,"Wait! Proses Handshaking 1.",0x3f);
  SendChar(0x06);

  do
  { /* Menunggu Karakter Handshaking Baud Rate Maksimum */

   textattr(0x17);
   gotoxy(53,20); cprintf("Baud Rate : ");cout << setw(4) << RBaudPrint;
   for (cbyte=0;cbyte<10;cbyte++)
   { rhpacket[cbyte]=ReceiveChar(); }
   checkHS=0;
   for (cbyte=1;cbyte<9;cbyte++)
   { checkHS=checkHS+rhpacket[cbyte]; }
   kodeHS=0;
   if (checkHS == rhpacket[9])
   {
     kodeHS=1;
     if (rhpacket[1] < BaudMaxRS)
     { BaudMax = rhpacket[1]; }
     else
     { BaudMax = BaudMaxRS; }
     for (cbyte=0;cbyte<9;cbyte++)
     { thpacket[cbyte]=BaudMax; }
     thpacket[9]=BaudMax*8;

     /* Mengirim Karakter Handshaking Baud Rate Maksimum */

     for (cbyte=0;cbyte<10;cbyte++)
     {
      SendChar(thpacket[cbyte]);
      delay(1);
     }
   }
   else
   { /* Mengirim Karakter Handshaking Baud Rate Maksimum */

     for(cbyte=0;cbyte<10;cbyte++)
     {
      thpacket[cbyte]=cbyte;
      SendChar(thpacket[cbyte]);
      delay(1);
     }
     if (RSCount_NAK() == 1)
     { kodeHS=2; }
   }
  }
  while(kodeHS == 0);

 if (kodeHS == 1)
 { /* ---- PROSES HANDSHAKING 2 ---- */

  loadscr(scr[9]);
  box(25,11,5,30,0x3f,"",0x3f,0);
  writecenter(25,55,13,"Wait! Proses Handshaking 2.",0x3f);
```

```
RC_ACK = 0;
RC_NAK = 0;
SBU = 0;
SBD = 0;
BaudRS=BaudMax;
delay(1500);
baud(BaudRS);

do
{
  textattr(0x17);
  gotoxy(53,20); cprintf("Baud Rate : ");cout << setw(4) << RBaudPrint;
  for (cbyte=0;cbyte<10;cbyte++)
  { rhpacket[cbyte]=ReceiveChar(); }
  checkHS=0;
  for (cbyte=1;cbyte<9;cbyte++)
  { checkHS=checkHS+rhpacket[cbyte]; }
  kodeHS=0;
  if (checkHS == rhpacket[9])
  {
   for (cbyte=0;cbyte<9;cbyte++)
   { thpacket[cbyte]=BaudMax; }
   thpacket[9]=BaudMax*8;
   for (cbyte=0;cbyte<10;cbyte++)
   {
    SendChar(thpacket[cbyte]);
    delay(1);
   }
   if (RC_ACK == 9)
   { kodeHS=1; }
   else
   { RSCount_ACK(); }
  }
  else
  { /* Mengirim Karakter Handshaking Baud Rate Maksimum */

   for(cbyte=0;cbyte<10;cbyte++)
   {
    thpacket[cbyte]=cbyte;
    SendChar(thpacket[cbyte]);
    delay(1);
   }
   if (RSCount_NAK() == 1)
   {
    loadscr(scr[9]);
    box(25,11,5,30,0x3f,"",0x3f,0);
    writecenter(25,55,13,"Komunikasi Gagal.",0x3f);
    goto batal1;
   }
  }
}
while(kodeHS == 0);

/* ----- PROSES RECEIVING ----- */

SBU=0;
SBD=0;
RC_ACK=0;
RC_NAK=0;

/* Inisialisasi Variabel Untuk Paket Data Terima */

nopacketRx = 1;
loadscr(scr[9]);
box(25,11,5,30,0x3f,"",0x3f,0);
writecenter(25,55,13,"Wait! Proses Receiving,",0x3f);
textattr(0x17);
gotoxy(53,19); cprintf("Connect  : ");cout << setw(4) << RBaudPrint;
gotoxy(53,20); cprintf("Baud Rate : ");cout << setw(4) << RBaudPrint;
```

```
/* Menerima Paket Data */

char kirim = 0,
     sw = 0,
     validitas;
do
{
  ReceivePacket();
  checksumRx = 0;
  cbyte=0;
  for (counter=2;counter<130;counter++)
  {
   checksumRx = checksumRx + rpacket[counter];
   if ((sw == 0) && (nopacketRx==1))
   {
    if ( (rpacket[counter] == 4) && (rpacket[counter+1] == 0) )
    { cbyte=counter; }
   }
  }

  validitas = 0;
  if (rpacket[0] == 1)
  {
   if (rpacket[1] == nopacketRx)
   {
    if (rpacket[130] == checksumRx)
    { validitas = 1; }
   }
  }

  /* Melakukan Pemrosesan Data */

  if (validitas == 1)  /* Data Benar dan Dilakukan Pemrosesan */
  {
   if ((sw == 0) && (nopacketRx==1))
   { /* Memroses Nama File, Panjang File */
    sw = 1;


    /*  Nama file */
    for (counter=0;counter<13;counter++)
    { namafile[counter]=0; }
    for(counter=2;counter<cbyte-11;counter++)
    { namafile[counter-2] = rpacket[counter]; }
    namafile[cbyte-11]='\0';
    ptr_file = fopen(namafile,"wb");
    receive = 0;
    receive1 = 0;

    /* Panjang File */
    char string[10];
    int desimal;
    desimal = rpacket[cbyte-1];
    for (counter=0;counter < desimal; counter++)
    { string[counter] = rpacket[cbyte-11+counter]; }
    string[desimal]='\0';
    filen = atol(string);
    filen1=filen;
   }
   else
   { /* Memroses Isi File */
    counter = 0;
    do
    {
     putc(rpacket[counter+2],ptr_file);
     receive++;
     receive1=receive;
     counter++;
    }
    while((counter<128) && (receive<filen));
```

```
            if (receive == filen) {kirim = 1;}  /* Telah Akhir File */
           }
           textattr(0x17);
           proses = receive1/filen1*100;
           gotoxy(11,20); cprintf("Receiving : %3d %",proses);
           SendChar(0x06);
           RSCount_ACK();
           nopacketRx++;
          }
          else
          { /* Data Salah  dan Dikirimkan NAK*/

           SendChar(0x15);
           if (RSCount_NAK() == 1)
           {
            kirim=1;
            loadscr(scr[9]);
            box(25,11,5,30,0x3f,"",0x3f,0);
            writecenter(25,55,13,"Komunikasi Gagal.",0x3f);
            goto batal1;
           }
          }
          textattr(0x17);
          gotoxy(53,20); cprintf("Baud Rate : ");cout << setw(4) << RBaudPrint;
      }
      while(kirim == 0);

      do
      {
        tanggapan=ReceiveChar();
        if (tanggapan == 4)
        {
         SendChar(0x06);
         RSCount_ACK();
        }
        else
        {
         SendChar(0x15);
         if (RSCount_NAK() == 1)
         {
          loadscr(scr[9]);
          box(25,11,5,30,0x3f,"",0x3f,0);
          writecenter(25,55,13,"Komunikasi Gagal.",0x3f);
          goto batal1;
         }
        }
      }
      while(tanggapan!=4);

      fclose(ptr_file);
      loadscr(scr[9]);
      box(25,11,5,30,0x3f,"",0x3f,0);
      writecenter(25,55,13,"Komunikasi Berhasil.",0x3f);
     }
     else
     {
      loadscr(scr[9]);
      box(25,11,5,30,0x3f,"",0x3f,0);
      writecenter(25,55,13,"Komunikasi Gagal.",0x3f);
     }
batal1:
    getch();
   }
}

void MenuTerimaModem()
{ getch(); }
```

```
void main()
{
 _setcursortype(_NOCURSOR);

 textattr(0x70);
 clrscr();
 textattr(0x07);
 writecenter(1,80,25,"Dominikus Ridwan - 2292 100 059",0x70);
 fillchar(1,2,80,24,"_",0x07);
 writecenter(1,80,1,"Program Pengatur Kelajuan Transfer Data Otomatis",0x70);


 savescr(scr[1]);

 NoCom = 0;          /* Nilai Default */
 BaudMaxRS = 5;
 BaudMaxModem = 5;
 RMaxPrint = 9600;
 MMaxPrint = 9600;

 initport();

 baud(1);
 Mbaud(1);
 BaudRS = 1;
 BaudModem = 1;
 char pilih1=1;

 do
 {
  loadscr(scr[1]);
  box(13,6,8,35,0x17," Menu Utama ",0x17,1);
  menu[0] = " 1. Inisialisasi ";
  menu[1] = " 2. Mode Kirim   ";
  menu[2] = " 3. Mode Terima  ";
  menu[3] = " 4. Selesai      ";
  pilih1 = winmenu(1,21,7,4,1,7,pilih1,menu);

  savescr(scr[2]);

  char pilih2=1;
  switch(pilih1)
  {
   case 1 : do
            {
              loadscr(scr[2]);
              box(38,8,7,35,0x17," Menu Inisialisasi ",0x17,1);
              menu[0] = " 1. Baud Rate Maksimum ";
              menu[1] = " 2. Port Komunikasi    ";
              menu[2] = " 3. Selesai            ";
              pilih2 = winmenu(1,44,9,3,1,7,pilih2,menu);

              switch(pilih2)
              {
               case 1 : loadscr(scr[1]);
                        MenuBaudMax();
                        break;
               case 2 : loadscr(scr[1]);
                        MenuPortCom();
                        break;
              }
            }
           while(pilih2 != 3);
           break;

   case 2 : do
            {
              loadscr(scr[2]);
              box(38,9,7,35,0x17," Menu Mode Kirim ",0x17,1);
              menu[0] = " 1. Melalui Null Modem ";
```

```
                menu[1] = " 2. Melalui Real Modem ";
                menu[2] = " 3. Selesai          ";
                pilih2 = winmenu(1,44,10,3,1,7,pilih2,menu);

                switch(pilih2)
                {
                 case 1 : loadscr(scr[1]);
                          MenuKirimRS();
                          break;
                 case 2 : loadscr(scr[1]);
                          MenuKirimModem();
                          break;
                }
              }
              while(pilih2 != 3);
              break;

    case 3 : do
              {
                loadscr(scr[2]);
                box(38,9,7,35,0x17," Menu Mode Terima ",0x17,1);
                menu[0] = " 1. Melalui Null Modem ";
                menu[1] = " 2. Melalui Real Modem ";
                menu[2] = " 3. Selesai          ";

                pilih2 = winmenu(1,44,10,3,1,7,pilih2,menu);

                switch(pilih2)
                {
                 case 1 : loadscr(scr[1]);
                          MenuTerimaRS();
                          break;
                 case 2 : loadscr(scr[1]);
                          MenuTerimaModem();
                          break;
                }
              }
              while(pilih2 != 3);
              break;
  }
 }
 while (pilih1 != 4);

 _setcursortype(_NORMALCURSOR);
}
```

05 FEB 1996

FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO
EL-1799 TUGAS AKHIR - 6 SKS

Nama Mahasiswa	: Dominikus Ridwan
No. Pokok	: 292 220 1888
Bidang Studi	: Telekomunikasi
Tugas Diberikan	: 31 Januari 1996
Tugas Diselesaikan	: 31 Juli 1996
Dosen Pembimbing	: 1. DR. Ir. Moch. Salehudin, MEng.Sc 2. Ir. Hanny Budinugroho
Judul Tugas Akhir	: Perencanaan dan Pembuatan Pengatur Kecepatan Modem Otomatis
Uraian Tugas Akhir	:

Dewasa ini informasi mempunyai peranan yang sangat penting. Untuk memperoleh informasi secara cepat dan tepat diperlukan teknologi komunikasi yang handal; di antaranya kecepatan pengiriman data (modem). Sementara itu perkembangan modem juga berlangsung cepat. Banyak modem-modem baru muncul dengan menawarkan teknologi yang lebih baik. Hal ini menyebabkan modem-modem terdahulu tergeser kedudukannya, padahal teknologinya belum tentu tertinggal seperti kecepatan transfer datanya cukup tinggi namun tidak bisa mengatur kecepatannya secara otomatis, dan modem jenis ini masih cukup banyak di pasaran.

Dengan tugas akhir ini akan dibuat suatu program pengatur kecepatan transfer data secara otomatis dengan memanfaatkan modem tersebut, sehingga kecepatannya bisa optimum.

Mengetahui,

Pembimbing I,

31/1/96

DR.Ir. Moch. Salehudin,MEng.Sc.
NIP. 130 532 026

Pembimbing II,

Ir. Hanny Budinugroho
NIP. 131 651 433

Bidang Studi Teknik Telekomunikasi
Koordinator,

Ir. M. Aries Purnomo
NIP. 130 532 040

31/1/96

Jurusan Teknik Elektro
Ketua,

DR.Ir. Moch. Salehudin,MEng.Sc.
NIP. 130 532 026

DEPARTEMEN PENDIDIKAN DAN KEBUDAYAAN INSTITUT TEKNOLOGI SEPULUH NOPEMBER FAKULTAS TEKNOLOGI INDUSTRI JURUSAN TEKNIK ELEKTRO

1. Judul : Perencanaan dan Pembuatan Pengatur Kecepatan Modem Otomatis.

2. Ruang Lingkup : - Pemrograman Komputer
- Sistem Komunikasi Data

3. Latar Belakang : Dewasa ini informasi mempunyai peranan yang sangat penting. Untuk memperoleh informasi secara cepat dan tepat diperlukan teknologi komunikasi yang handal; di antaranya kecepatan pengiriman data (modem). Sementara itu perkembangan modem juga berlangsung cepat. Banyak modem-modem baru muncul dengan menawarkan teknologi yang lebih baik. Hal ini menyebabkan modem-modem yang terdahulu tergeser kedudukannya oleh modem yang lebih baru.

4. Permasalahan : Jaringan telepon yang dipakai dalam komunikasi antar komputer kurang baik sehingga kecepatan transfer data tidak optimum. Oleh karena itu kecepatan transfer data harus diubah-ubah untuk disesuaikan dengan kondisi trafik pada saluran yang dipakai. Saat ini telah ada modem yang mampu mengatur kecepatannya secara otomatis, namun dengan harga yang cukup tinggi. Sementara itu, masih terdapat modem dengan harga yang relatif murah dan kecepatan yang cukup tinggi, tetapi tidak bisa mengatur kecepatannya secara otomatis.

Dalam hal ini akan dibuat suatu program pengatur kecepatan transfer data secara otomatis untuk modem-modem tersebut.

5. Pembatasan Masalah : 1. Protokol yang digunakan modem.
2. Jenis saluran transmisi yang digunakan, yaitu saluran telepon digital umum.
3. Jenis bahasa pemrograman yang dipakai.

6. Tujuan : Menghasilkan perangkat lunak yang dapat menyesuaikan kecepatan modem agar data dapat dikirim dengan kecepatan optimum.

7. Penelaahan Studi : Komunikasi data antar komputer dengan menggunakan modem lewat saluran telepon merupakan jenis komunikasi serial. Di bagian pengirim, sinyal digital komputer dimodulasi oleh modem agar bisa dikirim melalui saluran analog (kabel telepon). Sebaliknya di bagian penerima, sinyal analog tersebut didemodulasi modem agar dapat diterima oleh komputer.

Dalam sistem komunikasi di atas terdapat interaksi antara komputer dengan modem, modem dengan modem, dan modem dengan komputer. Protokol komunikasi antara komputer dengan modem menggunakan protokol RS 232, sedangkan protokol komunikasi antara modem dengan modem bisa bermacam-macam tergantung dari jenis modem yang digunakan.

Dengan menggunakan protokol RS 232, modem dapat dikontrol oleh komputer. Salah satu karakteristik modem yang dapat dikontrol oleh komputer yaitu kecepatan transfer data. Selain itu dapat diketahui juga karakteristik data yang dikirim, apakah diterima dengan baik atau cacat.
Jadi dengan mengetahui cacat atau tidaknya data yang dikirim maka kecepatan transfer data modem dapat dikontrol dengan menggunakan program komputer.

8. Langkah-langkah : 1. Studi literatur tentang protokol, komunikasi serial, dan jenis modem.
2. Perencanaan dan pembuatan perangkat lunak.
3. Uji coba program.
4. Pembuatan dan penyusunan laporan tugas akhir.

9. Jadwal Kegiatan :

Kegiatan	Bulan ke :					
	1	2	3	4	5	6
1. Studi Literatur						
2. Perencanaan & pembuatan program						
3. Uji coba program						
4. Penyusunan laporan tugas akhir						

10. Relevansi : Dengan adanya pengatur kecepatan modem secara otomatis diharapkan transfer data dapat dilakukan dengan kecepatan optimum.

RIWAYAT HIDUP

Nama : Dominikus Ridwan

NRP : 2292.100.059

Agama : Katholik

Tempat, Tgl. lahir : Surabaya, 14 - 11 - 1974

Alamat : Kapas Krampung 74 Surabaya

Penyusun adalah anak kedua dari dua orang bersaudara dari pasangan Jhon Lesmana dan July Wirjany.

Riwayat Pendidikan:

1. SD Kristen Cahaya Jember ; Lulus tahun 1986.
2. SMPK Maria Fatima Jember ; Lulus tahun 1989.
3. SMAK Santo Paulus Jember ; Lulus tahun 1992.
4. Diterima di Jurusan Teknik Elektro FTI - ITS, melalui UMPTN pada tahun 1992 dengan NRP. 292 220 1888.

Kegiatan Kampus:

1. Bendahara Studi Excursive Tahap Madya 1994.
2. Asisten Praktikum Dasar Sistem Komunikasi tahun 1995-1996.
3. Asisten Praktikum Sistem Komunikasi I dan II tahun 1996.
4. Bendahara Lomba Cipta Elektroteknik Nasional 1996.