PERANCANGAN DAN PEMBUATAN PENGIRIM GAMBAR DARI HELICOPTER MODEL YANG DIKONTROL OLEH PERSONAL COMPUTER (PC)

SKRIPSI

Disusun oleh :

ANGGA WIDYA OKTAVIAN

02. 17. 140

JURUSAN TEKNIK ELEKTRO S-1

KONSENTRASI TEKNIK ELEKTRONIKA

FAKULTAS TEKNOLOGI INDUSTRI

INSTITUT TEKNOLOGI NASIONAL

MALANG

2008

LEMBAR PERSETUJUAN

PERANCANGAN DAN PEMBUATAN PENGIRIM GAMBAR DARI HELICOPTER MODEL YANG DIKONTROL OLEH PERSONAL COMPUTER (PC)

SKRIPSI

Diajukan Untuk Memenuhi salah satu Syarat Memperoleh Gelar Sarjana Teknik Pada
Jurusan Teknik Elektro S-1 Konsentrasi Teknik Elektronika

Oleh :

Angga Widya Oktavian
02 17 140

Diperiksa dan Disetujui,

Dosen Pembimbing I	Dosen Pembimbing II
I Komang Somawirata, ST, MT	**Ir. Eko Nurcahyo**
NIP.P. 1030100361	**NIP.Y. 1028700172**

Mengetahui,
Ketua Jurusan Teknik Elektro S-1

Ir. F. Yudi Limpraptono, MT
NIP. Y. 1039500274

JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK ELEKTRONIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL
MALANG
2008

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK ELEKTRONIKA

BERITA ACARA UJIAN SKRIPSI
FAKULTAS TEKNOLOGI INDUSTRI

Nama Mahasiswa : Angga Widya Oktavian

NIM : 02.17.140

Jurusan : Teknik Elektro S-1

Konsentrasi : Teknik Elektronika

Judul Skripsi :

" PERANCANGAN DAN PEMBUATAN PENGIRIM GAMBAR DARI HELICOPTER MODEL YANG DIKONTROL OLEH PERSONAL COMPUTER (PC) "

Dipertahankan dihadapan Tim Penguji Skripsi Jenjang Strata Satu (S-1) pada:

Hari : Selasa

Tanggal : 23 September 2008

Dengan Nilai : 80,8 (A)

Panitia Ujian Skripsi

Ketua

(Ir. Mochtar Asroni, MSME.)
NIP. Y. 1018100036

Sekretaris

(Ir. F. Yudi Limpraptono, MT.)
NIP. Y. 1039500274

Anggota Penguji

Penguji Pertama

(Sotyohadi, ST, MSc.)
NIP. Y. 1039700309

Penguji Kedua

(Ir. Mimien Mustikawati, MT.)
NIP. P. 1030000352

ABSTRACK

PERANCANGAN DAN PEMBUATAN PENGIRIM GAMBAR DARI HELICOPTER MODEL YANG DIKONTROL OLEH PERSONAL COMPUTER (PC)

Angga Widya Oktavian, 02 17 140, Electrical Engineering S.1/Electronica
Adviser I : I Komang Somawirata, ST. MT
Adviser II : Ir. Eko Nurcahyo

- Keywords: Remote Control (R/C) Helicopter Model, Mikrokontroller AT89s8252, TLP 434A, RLP 434A, Motor DC, Wireless CCTV, TV TUNER

Mengontrol sebuah helicopter dari jarak jauh bukanlah hal yang mustahil kita lakukan saat ini. Di saat teknologi elektronika berkembang pesat, mengontrol sebuah helicopter dari jarak jauh akan dapat terwujud. Secara umum system R/C terdiri dari sebuah *transmitter* atau lebih, *receiver* dan beberapa buah motor dc sebagai actuator. Baterai sebagai sumber daya diperlukan oleh bagian pemancar maupun bagian penerima *Transmitter* bertugas menerima perintah kendali dari orang yang mengendalikan dan merubahnya menjadi kode-kode elektronik dan mengirimkannya melalui gelombang radio ke udara. Bagian *receiver* bertugas menerima informasi gelombang radio, menerjemahkan kode-kode elektriknya menjadi perintah gerak yang dikirimkan ke motor dc, selanjutnya motor dc bertugas sebagai aktuator ke posisi tertentu yang diinginkan. Disini penulis mempunyai ide untuk merancang suatu alat yang dapat mengontrol remote control sebagai alat pemantau dari udara melalui media *wireless* (remote control) yang mungkin dapat digunakan untuk mengintai suatu hal yang mengancam hidup seseorang misalnya untuk mengamati suatu kejadian dimana manusia tidak memungkinkan atau berbahaya untuk mendekatinya, atau juga dapat digunakan sebagai alat intelejen. Helicopter remote control ini dirancang utuk mengurangi suatu resiko yang besar. Peralatan yang digunakan satu set PC (personal computer) kemudian data dipancarkan oleh TLP434A dan selanjutnya data diterima oleh RLP434A kemudian diteruskan ke mikrokontroller AT89S8252 sebagai inputan yang akan mengaktifkan motor dc (servo).

Abstract

Controlling a helicopter of long distance is not impossible matter of us do conduct in this time. In technological moment of electronics rapidly grow, controlling a helicopter of long distance will be able to form. In general R/C system consist of a transmitter or more, receiver and some motor fruit of dc as actuator. Battery as resource needed by part of shares and also transmitter receiver of transmitter undertake to accept command conduct from one who control and change it become electronic codes and delivering it pass through radio wave into the air. Part of receiver undertake to accept radio wave information, translating electrical code of him become command of delivered motion to motor of dc, hereinafter motor of dc undertake as actuator to certain position which wanted. Here writer have an idea to design an appliance able to control remote as a means of watcher from the air pass through media of wireless what possible can be used to peep an matter menacing someone life for example to perceive an occurrence where human being do not enable or dangerous to coming near it, or also can be used as by appliance of intelligent. this Remote control Helicopter is designed by lessen an big risk. used equipments a set of PC (personal of computer) later; then data transmitted by TLP434A and hereinafter

data accepted by RLP434A is later; then distribute to AT89S8252 microcontroller as inputting to activate motor of dc (servo).

KATA PENGANTAR

Alhamdulillah, puji syukur kehadirat-Mu Ya Allah yang telah memberikan rahmat dan hidayah-Nya, sehingga dapat menyelesaikan skripsi yang berjudul "Perancangan dan Pembuatan Pengirim Gambar Dari Helicopter Model Yang dikontrol Oleh Personal Competer (PC)" ini dengan lancar. Skripsi ini merupakan persyaratan kelulusan Studi di Jurusan Teknik Elektro S-1 Konsentrasi Teknik Elektronika ITN Malang dan untuk mencapai gelar Sarjana Teknik.

Keberhasilan peyelesaian laporan skripsi ini tidak lepas dari dukungan dan bantuan berbagai pihak. Untuk itu penyusun menyampaikan terima kasih kepada :

1. Bapak Ir. F. Yudi Limpraptono, MT selaku Ketua Jurusan Teknik Elektro S-1.
2. Bapak I Komang Somawirata, ST. MT selaku Dosen Pembimbing I.
3. Bapak Ir. Eko Nurcahyo selaku Dosen Pembimbing II.
4. Ayah dan Ibu serta keluarga besar yang telah memberikan do'a restu, dorongan, semangat, dan biaya.
5. Rekan-rekan mahasiswa/i Elektronika S-1.
6. Semua yang telah membantu dalam penyelesaian penyusunan skripsi ini.

Penyusun telah berusaha semaksimal mungkin dan meyadari sepenuhnya akan keterbatasan pengetahuan dalam menyelasaikan laporan ini. Untuk itu penyusun mengharapkan saran dan kritik yang membangun dari pembaca demi kesempurnaan laporan ini.

Harapan penyusun semoga laporan skripsi ini memberikan manfaat bagi perkembangan ilmu pengetahuan dan pembaca.

Malang, September 2008

Penyusun

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR GRAFIK

BAB I

PENDAHULUAN

1.1 Latar Belakang

Pernahkah kita membayangkan jika itu ada di kehidupan nyata. Mengontrol sebuah helicopter dari jarak jauh bukanlah hal yang mustahil kita lakukan saat ini. Di saat teknologi elektronika berkembang pesat, disertai dengan kemudahan memperoleh peralatan-peralatan dengan biaya yang relative murah serta tersedianya buku-buku tutorial yang mendukung, mengontrol sebuah helicopter dari jarak jauh akan dapat terwujud.

Secara umum system R/C terdiri dari sebuah pemancar atau *transmitter,* sebuah atau lebih penerima atau *receiver* dan beberapa buah motor dc sebagai actuator(http://www.geocities.yahoo.com/p=remote+control+system).

Baterai sebagai sumber daya diperlukan oleh bagian pemancar maupun bagian penerima. Pemancar atau *transmitter* bertugas menerima perintah kendali dari orang yang mengendalikan dan merubahnya menjadi kode-kode elektronik dan mengirimkannya melalui gelombang radio ke udara. Bagian penerima atau *receiver* bertugas menerima informasi gelombang radio, menerjemahkan kode-kode elektriknya menjadi perintah gerak yang dikirimkan ke motor dc, selanjutnya motor dc bertugas sebagai aktuator ke posisi tertentu yang diinginkan.

Disini penulis mempunyai ide untuk merancang suatu alat yang dapat mengontrol remote control sebagai alat pemantau dari udara melalui media *wireless* (remote control) yang mungkin dapat digunakan untuk mengintai suatu

hal yang mengancam hidup seseorang misalnya untuk mengamati suatu kejadian dimana manusia tidak memungkinkan atau berbahaya untuk mendekatinya, atau juga dapat digunakan sebagai alat intelejen. Maka dari itu helicopter remote control ini dirancang utuk mengurangi suatu resiko yang besar. Peralatan yang digunakan satu set PC (personal computer) kemudian data dipancarkan oleh TLP434A dan selanjutnya data diterima oleh RLP434A kemudian diteruskan ke mikrokontroller AT89S8252 sebagai inputan yang akan mengaktifkan motor dc (servo).

1.2 Rumusan Masalah

Berdasarkan latar belakang yang telah diuraikan pada bagian sebelumnya, maka dapat dirumuskan beberapa masalah yang akan dibahas yaitu :

- Bagaimana merencanakan membuat instalasi hardware pada mikrokontroller AT89S8252
- Bagaimana system pengaturan remote control
- Bagaimana pengendalian motor dc menggunakan mikrokontroller

1.3 Tujuan

Tujuan dari skripsi ini adalah merancang dan membuat control pengendali helicopter model remote control (R/C) yang semula memakai hand control dirubah dengan menggunakan keyboard PC.

1.4 Batasan Masalah

Dalam menyusun skripsi ini diperlukan suatu batasan masalah agar tidak menyimpang dari ruang lingkup yang akan dibahas, batasan masalahnya meliputi:

- Tidak melakukan pembahasan pada pengiriman data secara wireless
- Pengendalian motor dc pada helicopter remote control menggunakan mikrokontroller AT89S8252
- Tidak membahas isi wireless tv tuner internal dan cctv
- Tidak membahas pengiriman gambar video

1.5 Metodologi

Metodologi yang dipakai dalam pembuatan skripsi ini adalah:

1. Studi Literatur

   Dengan mencari referensi-referensi yang berhubungan dengan perencanaan dan pembuatan alat yang akan dibuat.

2. *Field Research*

   Dengan melakukan penelitian secara langsung mengenai objek-objek yang berhubungan langsung dengan perencanaan alat yang akan dibuat.

3. *Design* dan Pembuatan Alat

   Yaitu meliputi pembuatan, perakitan komponen serta penyolderan dan pembuatan perangkat lunak.

4. Pengujian Alat

   Dengan melakukan pengujian perblok rangkaian dan kerja seluruh sistem pada alat tersebut.

1.6 Sistematika

Penulisan skripsi ini terbagi menjadi lima bab dengan sistematika sebagai berikut:

- **BAB I PENDAHULUAN**

  Membahas tentang latar belakang, rumusan masalah, tujuan, batasan masalah, metodologi dan sistematika penulisan pada penulisan skripsi ini.

- **BAB II LANDASAN TEORI**

  Berisikan tentang penjelasan dan teori-teori yang berhubungan dengan komponen-komponen yang digunakan dalan perancangan alat.

- **BAB III PERENCANAAN DAN PEMBUATAN ALAT**

  Membahas tentang perancangan alat yang terdiri dari perancangan perangkat keras dan perancangan perangkat lunak.

- **BAB IV PENGUJIAN ALAT**

  Membahas tentang pengujian peralatan secara keseluruhan dan analisa hasil pengujian.

- **BAB V PENUTUP**

  Berisikan kesimpulan yang didapat selama perancangan dan pembuatan alat serta saran-saran.

BAB II

LANDASAN TEORI

Pada bab ini akan dibahas mengenai dasar-dasar teori yang dapat menunjang dalam merencanakan dan membuat sistem kendali motor dc pada helicopter model *remote control (R/C)* menggunakan mikrokontroller AT89s8252. Teori penunjang ini akan membahas komponen dan peralatan pendukung pada alat yang dibuat. Untuk dapat memahami alat yang akan dirancang, maka dalam bab ini akan dijelaskan mengenai teori dasar yang akan berkaitan dengan sistem.

Uraian teori-teori dalam bab ini meliputi:

- Mikrokontroller AT89s8252
- *Transmitter* TLP434A dan *Receiver* RLP434A
- Motor DC
- Transistor
- RS 232
- Personal Computer
- LED (*Light Emitting Dioda*)
- WIRELESS-CAMERA

2.1. Mikrokontroler AT89s8252

2.1.1. Fitur Mikrokontroler AT89s8252

Perbedaan mendasar antara mikrokontroler dan mikroprosesor adalah mikrokontroler selain memiliki CPU juga dilengkapi memori dan input output yang merupakan kelengkapan sebagai sistem minimum mikrokomputer sehingga

sebuah mikrokontroler dapat dikatakan sebagai mikrokomputer dalam keeping tunggal (*Single Chip Microcomputer*) yang dapat berdiri sendiri.

Mikrokontroler AT89s8252 adalah mikrokontroler ATMEL yang kompatibel penuh dengan Mikrokontroler keluarga MCS-51, membutuhkan daya rendah, memiliki performance yang tinggi dan merupakan mikrokomputer 8 bit yang dilengkapi dengan *8 Kilobyte Flash memori untuk program, 2 Kilobyte EEPROM (Electrical Eraseable And Programmable Read Only Memory) dan 256 Byte RAM internal.* Program memori yang dapat diprogram ulang dalam sistem atau menggunakan programmer nonvolative Memori konvensional. Dalam system mikrokontroler terdapat dua hal mendasar yaitu: perangkat lunak dan perangkat keras yang keduanya saling terkait dan mendukung.

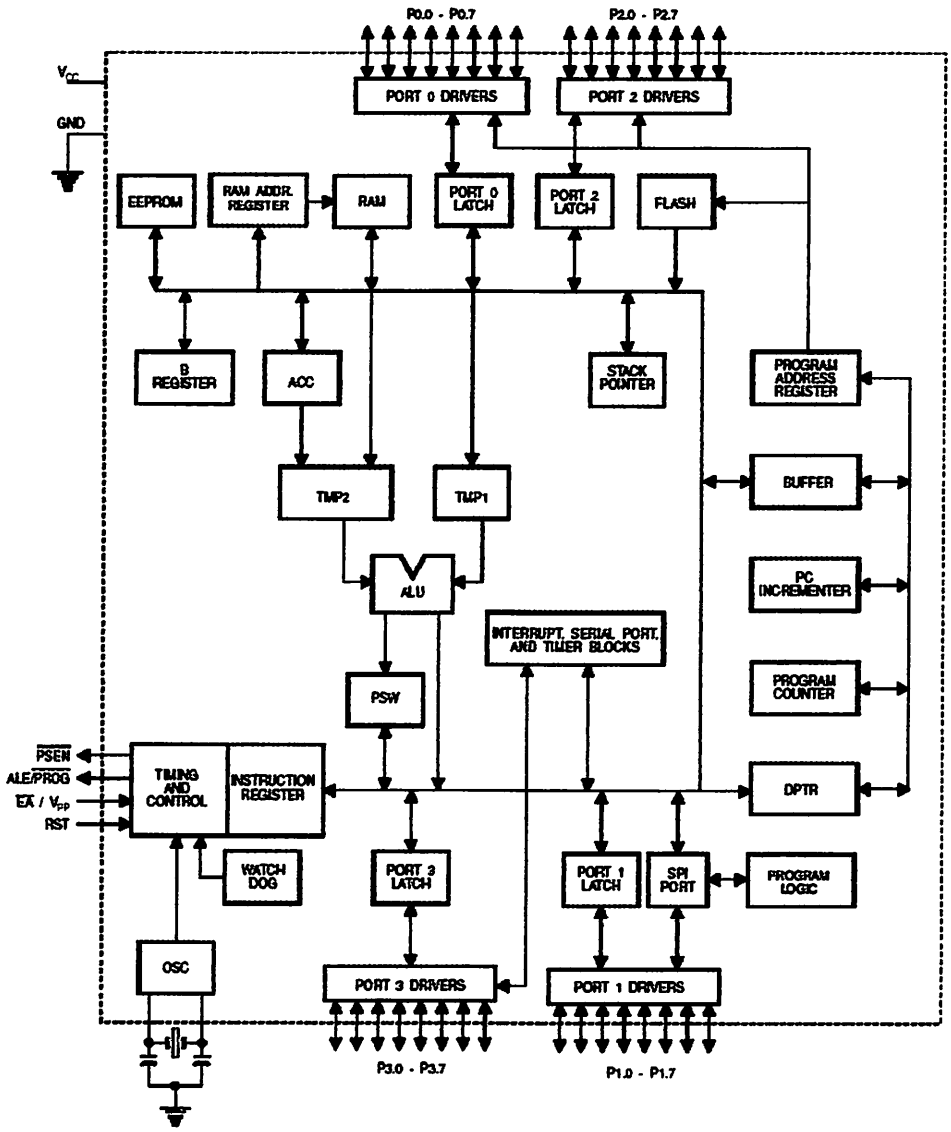
2.1.2 Arsitektur Mikrokontroler AT89s8252

Secara umum Mikrokontroler AT89s8252 memiliki :

- CPU 8 bit termasuk keluarga MCS-51
- 8 Kbyte *Flash* Memori
- 256 Byte *Internal* Memori
- 32 Port I/O, masing-masing terdiri atas 6 jalur I/O
- 3 *Timer/Counter* 16 Bit
- 2 *Serial Port Full Duplek*
- Kecepatan pelaksanaan instruksi per siklus 1 µS pada frekuensi *clock 12* Nhz
- 2 DPTR (*Data Pointer*)
- *Watchdog timer*
- Fleksibel ISP Programing

Dengan keistimewaan diatas pembuatan alat menggunakan AT89s8252 menjadi lebih sederhana dan tidak memerlukan IC pendukung yang banyak.

Adapun blok diagram dari Mikrokontroler AT89s8252 adalah sebagai berikut :

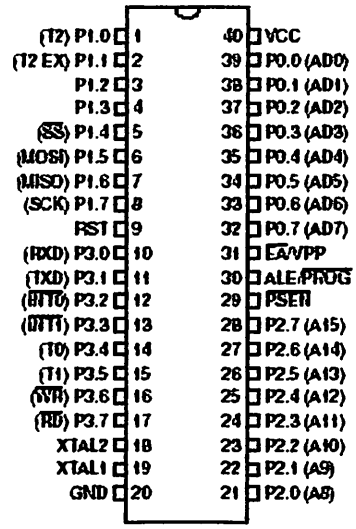


Gambar 2-1 Diagram Blok Mikrokontroler AT89s8252

(Sumber: Data Sheet Atmel AT89s8252)

2.1.3 Konfigurasi Pena-Pena Mikrokontroler AT89s8252

Mikrokontroler AT89s8252 terdiri dari 40 pin dengan konfigurasi sebagai berikut:



Gambar 2-2 Konfigurasi Pin-Pin AT89s8252

(Sumber: Data Sheet Atmel AT89s8252)

Fungsi-fungsi tiap pinnya adalah sebagai berikut:

- VCC (Supply tegangan), pin 40
- GND (Ground) , pin 20
- Port 0, pin 32-39

  Merupakan port input-output dua arah, tanpa internal pull-up dan konfigurasi sebagai multipleks bus alamat rendah (A_0-A_7) dan data selain pengaksesan program memory dan data memory eksternal.

- Port 1, pin 1-8

  Merupakan port input-output dua arah dengan internal pull-up.

- Port 2, pin 21-28

Merupakan port input-output dengan internal pull-up. Mengeluarkan alamat tinggi selama pengambilan program memori eksternal.

- Port 3, pin 10-17

  Merupakan port input-output dengan internal pull-up, dimana port 3 juga memiliki fungsi khusus dan dapat dilihat pada table berikut:

Tabel 2-1 Fungsi Khusus Pada Port 3

Nama Penyemat	Fungsi Khusus
Port 3.0	RxD (Port Masukan khusus)
Port 3.1	TxD (Port Keluaran Khusus)
Port 3.2	/INT0 (Masukan Interupsi Eksternal 0)
Port 3.3	/INT1 (Masukan Interupsi Eksternal 1)
Port 3.4	T0 (Masukan Pewaktu Eksternal 0)
Port 3.5	T1 (Masukan Pewaktu Eksternal 1)
Port 3.6	/WR (Sinyal Tulis Memori Data Eksternal)
Port 3.7	/RD (Sinyal Baca Memori Data Eksternal)

(Sumber: Data Sheet Atmel AT89s8252)

- RST (Reset), pin 9

  Input reset merupakan reset master untuk AT89s8252

- ALE /Prog (*Address Lacth Enable*), pin 30

  Digunakan untuk menahan alamat memori eksternal selama pelaksanaan intruksi.

- PSEN *(Program Store Enable)*, pin 29

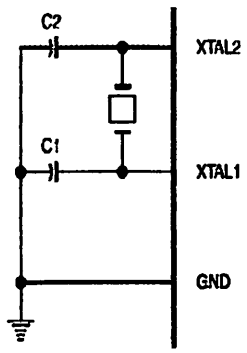
  Merupakan sinyal pengontrol yang memperbolehkan program memori eksternal masuk ke dalam bus.

- EA/VPP *(External Access)*, pin 31

  Dapat diberikan logika rendah (Ground) atau logika tinggi (+5 Volt). Jika diberikan logika tinggi maka mikrokontroler akan mengakses program dari ROM internal (EEPROM/Flash Memori), dan jika diberikan logika rendah maka mikrokontroler akan mengakses program dari memori eksternal.

- X-TAL 1 dan X-TAL 2,pin 18,19

  Pin ini dihubungkan dengan kristal bila menggunakan osilator internal. X-TAL 1 merupakan masukan ke rangkaian osilator internal sedangkan X-TAL 2 keluaran dari rangkaian osilator internal. Untuk keperluan ini diperlukan kapasitor penstabil sebesar 30 pF. Dan nilai dari X-TAL tersebut antara 4-24 Mhz. untuk lebih jelasnya dapat dilihat gambar pemasangan X-TAL serta kapasitor yang digunakannya.



Gambar 2-3 Osilator Eksternal AT89s8252

(Sumber: Data Sheet Atmel AT89s8252)

2.1.4 Organisasi Memory

Organisasi memori pada mikrokontroler AT89s8252 dapat dibagi menjadi dua bagian besar yaitu memori program dan memori data. Pembagian tersebut didasarkan atas fungsi dari penyimpanan data maupun program. Memori program digunakan untuk menyiimapn instruksi-instruksi yang akan dijalankan oleh mikrokontroler, sedangkan memori data digunkan sebagai tempat yang sedang diolah mikrokontroler.

Program mikrokontroler disimpan dalam memori program berupa ROM. Mikrokontroler AT89s8252 dilengkapi dengan Rom internal, sehingg untuk menyimpan program tidak digunakan ROM Eksternal yang terpisah dari mikrokontroler. Agar tidak menggunakan memori program eksternal, penyemat /EA dihubungkan dengan Vcc (logika 1).

Memori program mikrokontroler menggunakan alamat 16 Bit mulai dari 0000_H – $0FFF_H$ sehingga kapasitas penyimpanan program maksimal adalah 4Kbyte. Sinyal /PSEN (*Program Store Enable)* tidak digunakan jika menggunakan memori program internal.

Selain program mikrokontroler AT89s8252 juga memiliki data internal sebesar 128 Byte dan mampu mengakses memori data eksternal sebear 64 Kbyte. Semua memori data internal dapat dialamati dengan data langsung atau tidak langsung. Ciri dari pengalamatan langsung adalah *operand* adalah alamat register yang berisi alamat data yang akan diolah. Sebagian memori tersebut dapat dialamati dengan pengalamatan register, dan sebagian lagi dapat dialamati dengan memori satu bit. Untuk membaca data digunalan sinyal /RD sedangkan untuk menulis digunakan sinyal /WR.

2.1.5 SFR *(Special Fuction Register)*

Register Fungsi Khusus *(Special fuction Register)* terletak pada 128 byte bagian atas memori data internal dan berisi register-register untuk pelayanan latch port, timer, program status word (PSW), control peripheral dan sebagainya. Alamat register fungsi khusus ini ditunjukkan pada table 2-2.

Tabel 2-2 *Special Fucntion Register*

Simbol	**Nama Register**	**Alamat**
ACC	Accumulator	$E0_H$
B	Register B	$F0_H$
PSW	Program Status Word	$D0_H$
SP	Stack Pointer	81_H
DPTR	Data Pointer 2 Byte	
DPL	Bit Rendah	82_H
DPH	Bit Tinggi	83_H
P0	Port 0	80_H
P1	Port 1	90_H
P2	Port 2	$A0_H$
P3	Port 3	$B0_H$
IP	Interup Periority Control	$D8_H$
IE	Interup Enable Control	$A8_H$
TMOD	Timer/Counter Mode	89_H
TCON	Control	88_H
TH0	Timer/Counter Control	$8C_H$

TL0	Timer/Counter 0 High	$8A_H$
TH1	Control	$8D_H$
TL1	Timer/Counter 0 Low	$8B_H$
SCON	Control	98_H
SBUF	Timer/Counter 1 High	99_H
PCON	Control	87_H
	Timer/Counter 1 Low	
	Control	
	Serial Control	
	Serial Data Buffer	
	Power Control	

(Sumber: Data Sheet Atmel AT89s8252)

Beberapa macam register fungsi khusus yang sering digunakan adalah sebagai berikut :

- *Accumulator* (ACC) merupakan register untuk penambahan dan pengurangan, perintah Mneumonic untuk mengakses akumulator disederhanakan sebagai A.
- *Register B* merupakan register khusus yang berfungsi melayani operasi perkalian dan pembagian.
- *Stack Pointer* (SP) merupakan register 8 bit yang dapat diletakkan di alamat manapun pada RAM internal.

- *2 Data Pointer* (DPTR) terdiri atas dua register yaitu untuk byte tinggi (*data pointer high*, DPH) dan byte rendah (*data Pointer Low*, DPL) yang berfungsi untuk mengunci alamat 16 bit.
- *Port 0 sampai port 3* merupakan register yang berfungsi untuk membaca dan mengeluarkan data pada port 0,1,2,3. masing-masing register ini dapat dialamati per-byte maupun per-bit.
- *Control Register* terdiri dari register yang mempunyai fungsi control. Untuk mengontrol system interupsi, terdapat dua register khusus yaitu register IP (*Interrupt Priority*) dan register IE *(Interupt Enable*). Untuk mengontrol pelayanan timer/counter terdapat register khusus yaitu register TCON (*Timer/Counter Control*) serta pelayanan port srial menggunakan register SCON (*Serial Port Control).*

2.1.6 Sistem Interupsi

Mikrokontroler AT89s8252 mempunyai 5 buah sumber interupsi yang dapat membangkitkan permintaan yaitu INT0, INT1, T1, T2 dan Port Serial.

Saat terjadinya interupsi mikrokontroler secara otomatis akan menuju ke subrutin pada alamat tersebut. Setelah interupsi selesai dikerjakan, mikrokontroler akan mengerjakan program semula. Tiap-tiap sumber interupsi dapat *enable* atau *disable* secara software.

Tingkat prioritas semua sumber interrupt dapat diprogram sendiri-sendiri dengan set atau clear bit pada (*interrupt priority).* Jika dua permintaan interupsi dengan tingkat prioritas yang berbeda diterima secara bersamaan akan dilakukan polling untuk menentukan mana yang akan dilayani.

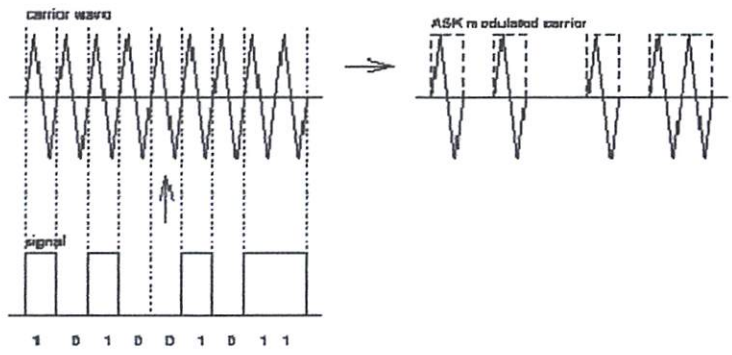
Tabel 2-3 Alamat Sumber Interupsi

Sumber Interupsi	Alamat Awal
Interupt Luar 0 (INT 0)	03_H
Pewaktu/Pencacah 0 (T0)	$0B_H$
Interrupt Luar 1 (INT 0)	13_H
Pewaktu/Pencacah 0 (T0)	$1B_H$
Port Serial	23_H

(Sumber: Data Sheet Atmel AT89s8252)

2.2. *Transmitter* TLP434A dan *Receiver* RLP434A

Transmitter TLP434A dan *receiver* RLP434A adalah sepasang modul pengiriman dan penerimaan data tanpa menggunakan kabel (*wirelles*) yang bekerja pada frekuensi 434 MHz. Sistem modulasi yang digunakan adalah modulasi digital ASK (*Amplitudo Syift Keying*) yaitu modulasi yang menyatakan sinyal digital 1 sebagai suatu nilai tegangan tertentu (misalnya 1 volt) dan sinyal digital 0 sebagai sinyal digital dengan tegangan 0 volt. sinyal ini yang kemudian digunakan untuk menyala-mati-kan pemancar, kira-kira mirip sinyal morse (affan@itb.ac.id).



Gambar 2-4 Modulasi ASK (*Amplitudo Syift Keying*)

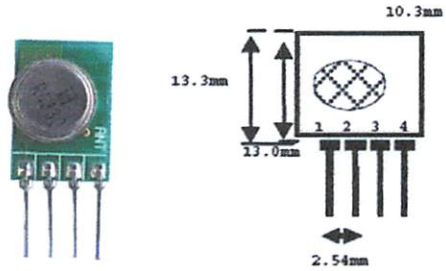
Sumber: Elektonika Komunikasi jilid 2

2.2.1. *Transmitter*

TLP 434A adalah suatu modul RF *(Radio Frekuensi)* dari Laipac Tecnology, Inc. yang biasa digunakan pada aplikasi *mobile robot*, alarm, *remote control*, dan transfer data. Jarak pancar yang mampu dijangkau tergantung dari berbagai faktor diantaranya faktor antena, kebisingan, tegangan kerja dari pemancar dan juga resistansi yang dimiliki suatu obyek penghalang. Jarak efektif kurang lebih 30 meter untuk pemakaian *indoor* dan 100 meter untuk pemakaian *outdoor* (www.futurlec.com dan Datasheet TLP/RLP434A).

Spesifikasi dan penjelasan dari *transmitter* TLP434A antara lain sebagai berikut:

- *Operation Voltage* antara 2-12 Volt DC
- Bekerja pada frekuensi 315, 418, dan 433,92 MHz
- *Case Temperature* = 25°C
- Data *Rate* sampai 4.8Kbps



Gambar 2-5 Modul dan Ukuran Dimensi TLP434A

(Sumber: Datasheet TLP/RLP434A)

Tabel 2-4 Konfigurasi Pin TLP434A

Pin 1	GND
Pin 2	Data in
Pin 3	Vcc
Pin 4	Antena (RF Output)

(Sumber: Datasheet TLP/RLP 434A)

2.2.2. *Receiver*

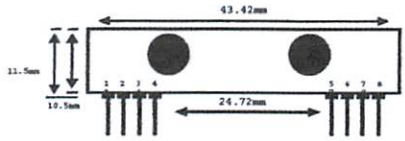
Receiver adalah suatu *hardware* yang berfungsi menerima sinyal informasi yang dikirim oleh suatu pemancar yang memiliki frekuensi yang sama dengan antara pemancar dan penerima. Spesifikasi dari RLP434A antara lain:

- *Operation voltage* 3,3 sampai 6,0 Volt DC
- Bekerja pada frekuensi 315, 418, dan 433,92 MHz
- *Operation temperature* antara -20°C sampai 80°C
- Data *Rate* sampai 4.8Kbps

Untuk ukuran fisiknya, RLP 434A memiliki dimensi yang lebih lebar dibandingkan dengan TLP 434A. Untuk lebih jelas dapat dilihat dari penjelasan berikut ini :

Gambar 2-6 Modul RLP 434A

(Sumber: *Datasheet* TLP/RLP 434A)



Gambar 2-7 Ukuran Fisik RLP 434A

(Sumber: *Datasheet* TLP/RLP 434A)

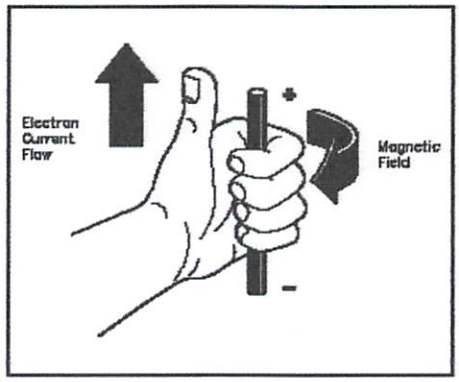
Tabel 2-5 Konfigurasi Pin RLP 434A

Pin 1	Ground
Pin 2	Digital Data Output
Pin 3	Linier Output/Test
Pin 4	Vcc
Pin 5	Vcc
Pin 6	Ground
Pin 7	Ground
Pin 8	Antena

(Sumber: *Datasheet* TLP/RLP 434A)

2.3. Motor DC

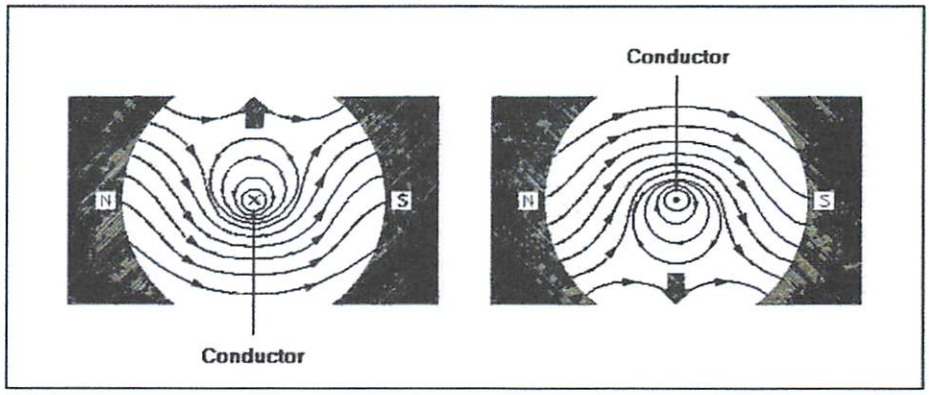
Setiap arus yang mengalir melalui sebuah konduktor akan menimbulkan medan magnet. Arah medan magnet dapat ditentukan dengan kaidah tangan kiri. Ibu jari tangan menunjukkan arah aliran arus listrik sedangkan jari-jari yang lain menunjukkan arah medan magnet yang timbul, seperti yang ditunjukkan oleh gambar 2-8 berikut ini.



Gambar 2-8 Kaidah Tangan Kiri

(Sumber: www.CoolCircuit.com)

Jika suatu konduktor yang dialiri arus listrik ditempatkan dalam sebuah medan magnet, kombinasi medan magnet akan ditunjukkan oleh gambar 2-9. Arah aliran arus listrik dalam konduktor ditunjukkan dengan tanda "x" atau ".". Tanda "x" menunjukkan arah arus listrik mengalir menjauhi pembaca gambar, tanda "." menunjukkan arah arus listrik mengalir mendekati pembaca gambar.

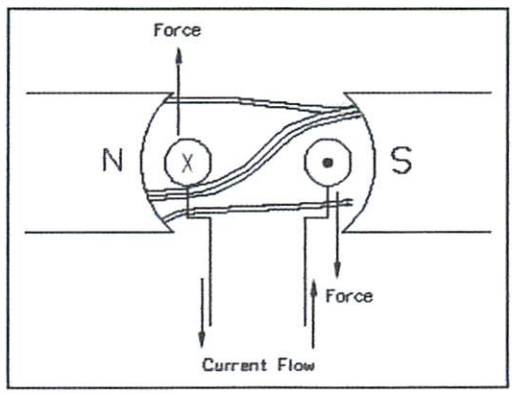


Gambar 2-9 Konduktor Berarus Listrik Dalam Medan Magnet

(Sumber: www.CoolCircuit.com)

Pada gambar sebelah kiri, arah medan magnet pada sisi atas yang dihasilkan oleh konduktor berlawanan dengan arah medan magnet yang dihasilkan oleh magnet permanen. Sementara pada sisi sebelah bawah, arah medan magnet yang dihasilkan oleh konduktor searah dengan arah medan magnet yang dihasilkan oleh magnet permanen. Dengan kata lain, pada sisi sebelah atas kerapatan *fluks* magnet lebih sedikit dari pada sisi sebelah bawah. Sebuah gaya dorong akan menyebabkan konduktor bergerak ke sisi sebelah atas. Pada gambar sebelah kanan, arah medan magnet pada sisi atas yang dihasilkan oleh konduktor searah dengan arah medan magnet yang dihasilkan oleh magnet permanen. Sementara pada sisi sebelah bawah, arah medan magnet yang dihasilkan oleh konduktor berlawanan dengan arah medan magnet yang dihasilkan oleh magnet permanen. Dengan kata lain, pada sisi sebelah bawah kerapatan *fluks* magnet lebih sedikit dari pada sisi sebelah atas. Sebuah gaya dorong akan menyebabkan konduktor bergerak ke sisi sebelah bawah. Pada

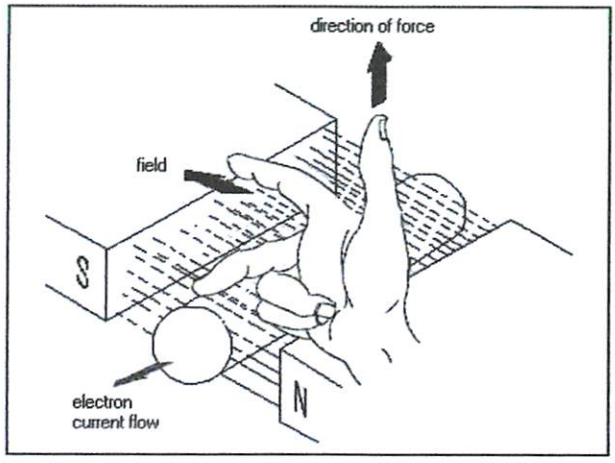
sebuah motor dc, konduktor dibentuk menjadi sebuah *loop* sehingga ada dua bagian konduktor yang berada didalam medan magnet pada saat yang sama, seperti diperlihatkan pada gambar 2-10. Konfigurasi konduktor seperti ini akan menghasilkan *distorsi* pada medan magnet utama dan menghasilkan gaya dorong pada masing-masing konduktor. Pada saat konduktor di tempatkan pada rotor, gaya dorong yang timbul akan menyebabkan rotor berputar searah dengan jarum jam, seperti diperlihatkan pada gambar 2-10.



Gambar 2-10 Bergeraknya Sebuah Motor

(Sumber: www.CoolCircuit.com)

Sebuah cara lagi untuk menunjukkan hubungan antara arus listrik yang mengalir didalam sebuah konduktor, medan magnet dan arah gerak, adalah kaidah tangan kanan untuk motor seperti yang diperlihatkan pada gambar 2-11.



Gambar 2-11 Kaidah Tangan Kanan Untuk Motor

(Sumber: www.CoolCircuit.com)

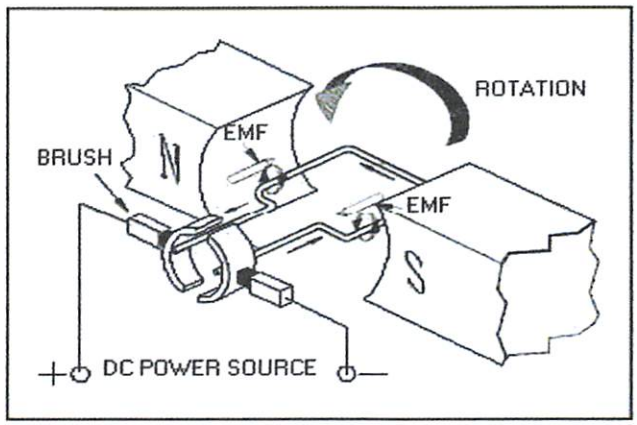
Kaidah tangan kanan untuk motor menunjukkan arah arus yang mengalir didalam sebuah konduktor yang berada dalam medan magnet. Jari tengah menunjukkan arah arus yang mengalir pada konduktor, jari telunjuk menunjukkan arah medan magnet dan ibu jari menunjukkan arah gaya putar. Adapun besarnya gaya yang bekerja pada konduktor tersebut dapat dirumuskan dengan :

F = B.L.I (Newton)

Dimana : B = kerapatan *fluks* magnet (weber)

L = panjang konduktor (meter)

I = arus listrik (ampere)



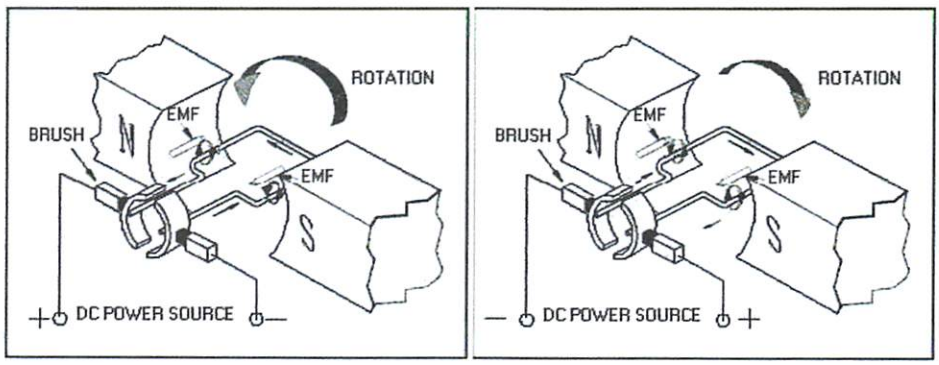
Gambar 2-12 Konstruksi Dasar Motor DC

(Sumber: www.CoolCircuit.com)

Pada gambar 2-12 diatas tampak sebuah konstruksi dasar motor dc, pada gambar diatas terlihat bahwa pada saat terminal motor diberi tegangan dc, maka arus elektron akan mengalir melalui konduktor dari terminal negatif menuju ke terminal positif. Karena konduktor berada diantara medan magnet, maka akan timbul medan magnet juga pada konduktor yang arahnya seperti terlihat pada gambar 2-12. diatas. Arah garis gaya medan magnet yang dihasilkan oleh magnet permanen adalah dari kutub utara menuju ke selatan. Sementara pada konduktor yang dekat dengan kutub selatan, arah garis gaya magnet disisi sebelah bawah searah dengan garis gaya magnet permanen sedangkan di sisi sebelah atas arah garis gaya magnet berlawanan arah dengan garis gaya magnet permanen. Ini menyebabkan medan magnet disisi sebelah bawah lebih rapat daripada sisi sebelah atas. Dengan demikian konduktor akan terdorong ke arah atas. Sementara pada konduktor yang dekat dengan kutub utara, arah garis gaya magnet disisi sebelah atas searah dengan garis gaya magnet permanen sedangkan di sisi sebelah

bawah arah garis gaya magnet berlawanan arah dengan garis gaya magnet permanen. Ini menyebabkan medan magnet disisi sebelah atas lebih rapat daripada sisi sebelah bawah. Dengan demikian konduktor akan terdorong ke arah bawah. Pada akhirnya konduktor akan membentuk gerakan berputar berlawanan dengan jarum jam seperti terlihat pada gambar 2-12 di atas.

2.3.1. Pengendalian Arah Putaran Motor DC



Gambar 2-13 Arah Putaran Motor DC

(Sumber: www.CoolCircuit.com)

Putaran motor dc sebenarnya dari kiri ke kanan, untuk mengubah putaran motor dari kiri ke kanan atau berbalik arah dari yang seharusnya dapat dilakukan dengan cara membalik arah arus motor tersebut dengan tegangan sumber, yaitu kutub positif pada tegangan sumber dan negatif pada motor disambungkan kekutub positif pada tegangan sumber, maka motor akan berputar berlawanan dari arah sebenarnya yaitu dari kiri ke kanan.

2.4. Transistor

Prinsip kerja transistor adalah arus bias *base - emitter* yang kecil mengatur besar arus *colector – emitter*. Bagian penting berikutnya adalah bagaimana caranya memberi arus bias yang tepat sehingga transistor dapat bekerja optimal.

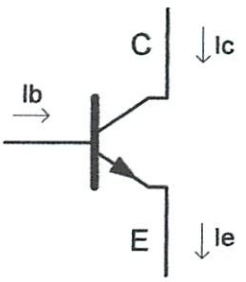
2.4.1. Arus Bias

Ada tiga cara yang umum untuk memberi arus bias pada transistor, yaitu rangkaian CE (*Common Emitter*), CC (*Common Colector*), CB (*Common Base*). Namun saat ini akan lebih detil jika dijelaskan bias transistor rangkaian CE. Dengan menganalisa rangkaian CE akan dapat diketahui beberapa parameter penting dan berguna, terutama untuk memilih transistor yang tepat untuk aplikasi tertentu. Misalnya untuk aplikasi pengolahan sinyal audio tentu saja tidak menggunakan transistor power.

2.4.2. Arus Emitor

Dari Hukum Kirchoff diketahui bahwa jumlah arus yang masuk ke satu titik akan sama dengan arus yang keluar. Jika teorema tersebut diaplikasikan pada transistor, maka hukum itu menjelaskan hubungan:

$$I_E = I_C + I_B$$



Gambar 2–14 Arus Emitor

(Sumber: www.electroniclab.com,rubrik elka analog, aswan hamonangan)

Persamaan di atas mengatakan arus emitor I_E adalah jumlah dari arus kolektor I_C dengan arus base I_B. karena arus I_B sangat kecil sekali atau disebutkan $I_B << I_C$, maka dapat dinyatakan :

$$I_E = I_C$$

2.4.3. Alpha (a)

Pada tabel data transistor (*databook*) sering dijumpai spesifikasi a_{dc} (alpha dc) yang tidak lain adalah :

$$a_{dc} = I_C / I_E$$

Definisinya adalah perbandingan arus kolektor terhadap arus emitor. Karena besar arus kolektor umumnya hampir sama dengan besar arus emitor, maka idealnya besarnya a_{dc} adalah = 1 (satu). Namun pada umumnya transistor yang ada memiliki a_{dc} kurang lebih antara 0,95 sampai 0,99.

2.4.4. Bheta (b)

Bheta didefinisikan sebagai besar perbandingan antara arus kolektor dengan arus base.

$$b = I_C / I_B$$

Dengan kata lain, b adalah parameter yang menunjukkan kemampuan penguatan arus (*current gain*) dari suatu transistor. Parameter ini ada tertera di *databook* transistor dan sangat membantu para perancang rangkaian elektronika dalam merencanakan rangkaiannya.

Misalnya jika suatu transistor diketahui besar b = 250 dan diinginkan arus kolektor sebesar 10 mA, maka berapakah arus bias base yang diperlukan. Tentu saja jawabannya sangat mudah, yaitu :

$$I_B = \frac{Ic}{b} = \frac{10mA}{250} = 40\ \mu A$$

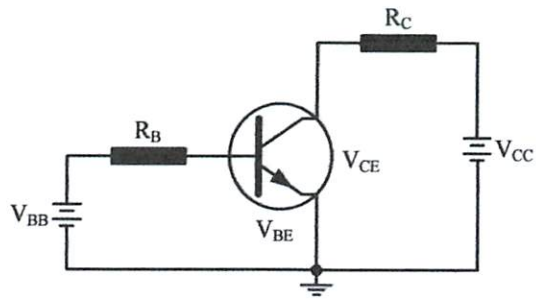
Arus yang terjadi pada kolektor transistor yang memiliki b = 200 jika diberi arus bias base sebesar 0,1 mA adalah :

$$I_C = b \,.\, I_B = 200 \times 0{,}1\ mA = 20\ mA$$

Dari rumusan di atas lebih terlihat definisi penguatan arus transistor, yang sekali lagi, arus base yang kecil menjadi arus kolektor yang lebih besar.

2.4.5. Common Emitter (CE)

Rangkaian CE adalah rangkaian yang paling sering digunakan untuk berbagai aplikasi yang menggunakan transistor. Dinamakan rangkaian CE sebab titik *ground* atau titik tegangan 0 Volt dihubungkan pada titik emitter.



Gambar 2–15 Rangkaian CE

(Sumber: www.electroniclab.com, rubrik elka analog, aswan hamonangan)

Sekilas tentang notasi, ada beberapa notasi yang sering digunakan untuk menunjukkan besar tegangan pada suatu titik maupun antar titik. Notasi dengan

satu *subscript* adalah untuk menunjukkan besar tegangan pada satu titik, misalnya V_C = tegangan kolektor, V_B = tegangan base, dan V_E = tegangan *emitter*.

Ada juga notasi dengan dua *subscript* yang dipakai untuk menunjukkan besar tegangan antar dua titik, yang disebut juga dengan tegangan jepit. Di antaranya adalah :

V_{CE} = tegangan jepit kolektor – emitor

V_{BE} = tegangan jepit base – emitor

V_{CB} = tegangan jepit kolektor base

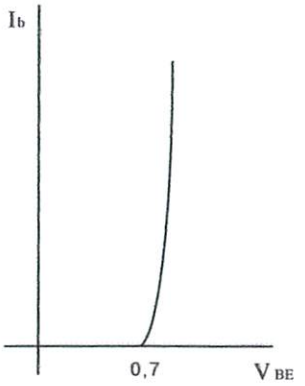
Notasi seperti V_{BB}, V_{CC}, V_{EE} berturut – turut adalah besar sumber tegangan yang masuk ke titik base, kolektor, dan emitor.

2.4.6. Kurva Base

Hubungan antara IB dan VBE tentu saja akan berupa kurva dioda. Karena memang telah diketahui bahwa *junction base – emitor* tidak lain adalah sebuah dioda. Jika hukum Ohm diterapkan pada loop base diketahui adalah :

$$I_B = \frac{(V_{BB} - V_{BE})}{R_B}$$

V_{BE} adalah tegangan jepit dioda *junction base – emitor*. Arus hanya akan mengalir jika tegangan antara base – emitor lebih besar dari V_{BE}. Sehingga arus I_B mulai aktif mengalir pada saat nilai V_{BE} tertentu.



Grafik 2–1 Kurva I_B - V_{BE}

(Sumber: www.electroniclab.com, rubrik elka dasar, aswan hamonangan)

Besar V_{BE} umumnya tercantum dalam *databook*, tetapi untuk penyederhanaan umumnya diketahui V_{BE} = 0,7 Volt untuk transistor silikon dan V_{BE} = 0,3 Volt untuk transistor germanium. Nilai ideal V_{BE} = 0 Volt.

Sampai disini akan sangat mudah mengetahui arus I_B dan arus I_C dari rangkaian berikut ini, jika diketahui besar b = 200. Misalnya yang digunakan adalah transistor yang dibuat dari bahan silikon.

$$I_B = \frac{(V_{BB} - V_{BE})}{R_B}$$

$$= \frac{(2V - 0,7V)}{100K\Omega}$$

$$= 13\ \mu A$$

Dengan b = 200, maka arus kolektor adalah :

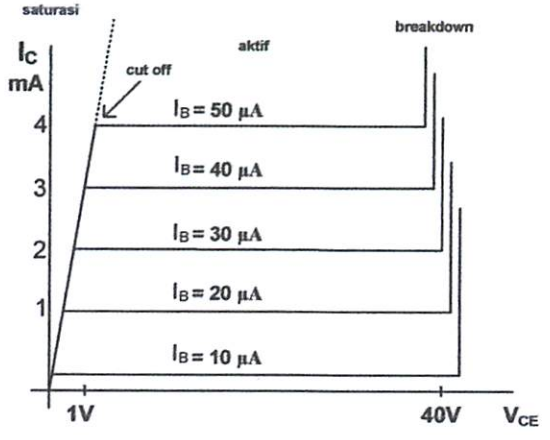
$$I_C = b \cdot I_B$$

$$= 200 \times 13\mu A$$

$$= 2,6\ mA$$

2.4.7. Kurva Kolektor

Sekarang sudah diketahui konsep arus *base* dan arus kolektor, satu hal lain yang menarik adalah bagaimana hubungan arus base I_B, arus kolektor I_C, dan tegangan kolektor kolektor – *emitter* V_{CE}. Dengan menggunakan rangkaian-01, tegangan V_{BB} dan V_{CC} dapat diatur untuk memperoleh *plot* garis – garis kurva kolektor. Pada gambar berikut telah diplot beberapa kurva kolektor arus I_C terhadap V_{CE} dimana arus I_B dibuat konstan.



Grafik 2–2 Kurva Kolektor

(Sumber: www.electroniclab.com, rubrik elka dasar, aswan hamonangan)

Dari kurva ini dapat terlihat ada beberapa region yang menunjukkan daerah kerja transistor. Pertama adalah daerah saturas*i*, lalu daerah *cut-off*, kemudian daerah aktif, dan seterusnya adalah daerah *breakdown*.

2.4.8. Daerah Aktif

Daerah kerja transistor yang normal adalah pada daerah aktif, dimana arus I_C konstan terhadap berapapun nilai V_{CE}. Dari kurva ini diperlihatkan bahwa arus

I_C hanya bergantung pada besar arus I_B. daerah kerja ini biasa disebut juga daerah linear (*linear region*).

Jika Hukum Kirchoff mengenai tegangan dan arus diterapkan pada *loop* kolektor (rangkaian CE), maka dapat diperoleh hubungan :

$$V_{CE} = V_{CC} - I_C \cdot R_C$$

Dapat dihitung dissipasi daya transistor adalah :

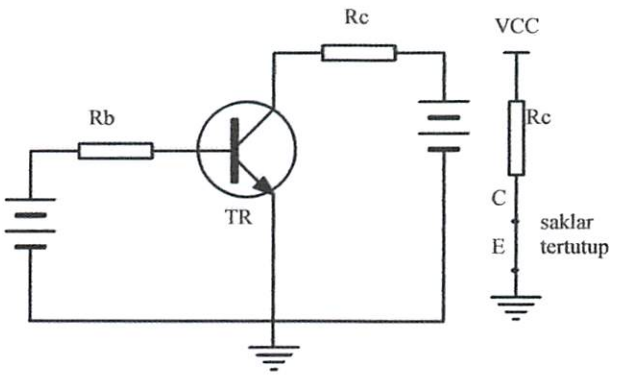
$$P_D = V_{CE} \cdot I_C$$

Rumus ini mengatakan jumlah dissipasi daya transistor adalah tegangan kolektor – *emitter* dikalikan dengan jumlah arus yang melewatinya. Dissipasi daya ini berupa panas yang menyebabkan naiknya temperatur transistor. Umumnya untuk transistor power sangat perlu untuk mengetahui spesifikasi P_{Dmax}. Spesifikasi ini menunjukkan temperatur kerja maksimum yang diperbolehkan agar transistor masih bekerja normal. Sebab jika transistor bekerja melebihi kapasitas daya P_{Dmax}, maka transistor dapat rusak atau terbakar.

2.4.9. Daerah Saturasi (Dalam Keadaan Jenuh)

Daerah Saturasi adalah mulai dari V_{CE} = 0 Volt sampai dengan kira - kira 0,7 Volt (transistor silikon), yaitu akibat dari efek dioda kolektor - base yang mana tegangan V_{CE} belum mencukupi untuk dapat menyebabkan aliran elektron. Transisitor dalam keadaan jenuh (saturasi), maka berlaku :

- Kuat arus (I_c) mencapai maksimum
- V_{ce} sama dengan 0 volt
- Tegangan pada beban sama dengan tegangan sumber (V_{cc}=V_{Rc})



Gambar 2–16 Transistor Dalam Keadaan Saturasi

(Sumber: www.electroniclab.com, rubrik elka dasar, aswan hamonangan)

Untuk menghitung resistansi pada basis menggunakan rumus :

$$V_{CC} - I_C.R_C - V_{CE} = 0$$

Karena keadaan saturasi $V_{ce} = 0$ maka rumusnya menjadi :

$$V_{CC} - I_C.R_C = 0$$

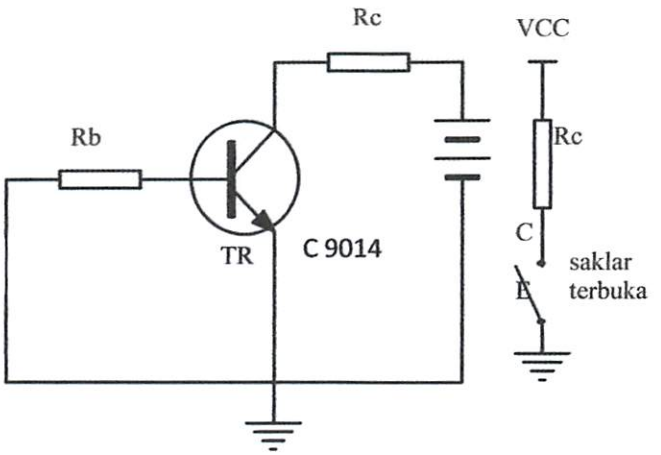
$$I_C = \beta_{dc}.\ I_B$$

$$I_B = \frac{(V_{CC} - V_{BE})}{R_B}$$

2.4.10. Daerah *Cut-Off* (Sumbat)

Transistor dalam keadaan *cut off* (sumbat) berlaku hal-hal sebagai berikut :

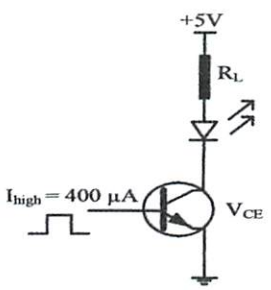
- Arus I_b sama dengan 0 volt
- Arus I_c sangat kecil sekali sehingga dapat diabaikan
- V_{cc} sama V_{ce}



Gambar 2–17 Transistor dalam Keadaan *Cutt Off* (Sumbat)

(Sumber: www.electroniclab.com, rubrik elka dasar, aswan hamonangan)

Jika kemudian tegangan V_{CC} dinaikkan perlahan - lahan sampai tegangan V_{CE} tertentu tiba - tiba arus I_C mulai konstan. Pada saat perubahan ini, daerah kerja transistor berada pada daerah *cut-off*, yaitu dari keadaan saturasi (OFF) lalu menjadi aktif (ON). Perubahan ini dipakai pada sistem digital yang hanya mengenal angka biner 0 dan 1 yang tidak lain dapat direpresentasikan oleh status transistor OFF dan ON.



Gambar 2–18 Rangkaian *Driver* LED

(Sumber: www.electroniclab.com, rubrik elka analog, aswan hamonangan)

Misalkan pada rangkaian *driver* LED di atas, transistor yang digunakan adalah transistor dengan b = 50. Penyalaan LED diatur oleh sebuah gerbang logika (*logic gate*) dengan arus *output high* = 400 μA dan diketahui tegangan *forward* LED V_{LED} = 2,4 Volt. Lalu pertanyaannya adalah berapakah seharusnya resistansi R_L yang dipakai?

$I_C = b \cdot I_B$

$= 50 \times 400\ \mu A$

$= 20\ mA$

Arus sebesar ini cukup untuk menyalakan LED pada saat transistor *cut-off*. Tegangan V_{CE} pada saat *cut-off* idealnya = 0, dan aproksimasi ini sudah cukup untuk rangkaian ini.

$R_L = (V_{CC} - V_{LED} - V_{CE}) / I_C$

$= (5 - 2{,}4 - 0) / 20\ mA$

$= 2{,}6\ V / 20\ mA = 130\ Ohm$

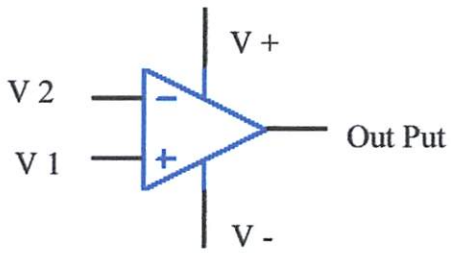
2.4.11. Daerah Breakdown

Dari kurva kolektor, terlihat jika teganngan V_{CE} lebih dari 40 V, arus I_C menanjak naik dengan cepat. Transistor pada daerah ini disebut berada pada daerah *breakdown*. Seharusnya transistor tidak boleh bekerja pada daerah ini, karena akan dapat merusak transistor tersebut. Untuk berbagai jenis transistor, nilai tegangan V_{CEmax} yang diperbolehkan sebalum *breakdown* berfariasi. V_{CEmax} pada *databook* transistor selalu dicantumkan juga.

2.4.12. Pembanding *(Comparator)*

Komparator adalah rangkaian yang membandingkan sinyal input tegangan V1 dengan tegangan acuan V2. jika kondisi dimana tegangan tak membalik lebih

besar dari tegangan membalik komparator menghasilkan tegangan yang lebih tinggi, namun bila masukan tak membalik lebih kecil daripada masukan membalik maka nilai tegangan keluarannya rendah.



Gambar 2-19 Pembanding (Comparator)

(Sumber: www.labITB.com)

Dari gambar tersebut dapat dijelaskan keadaan sebagai berikut:

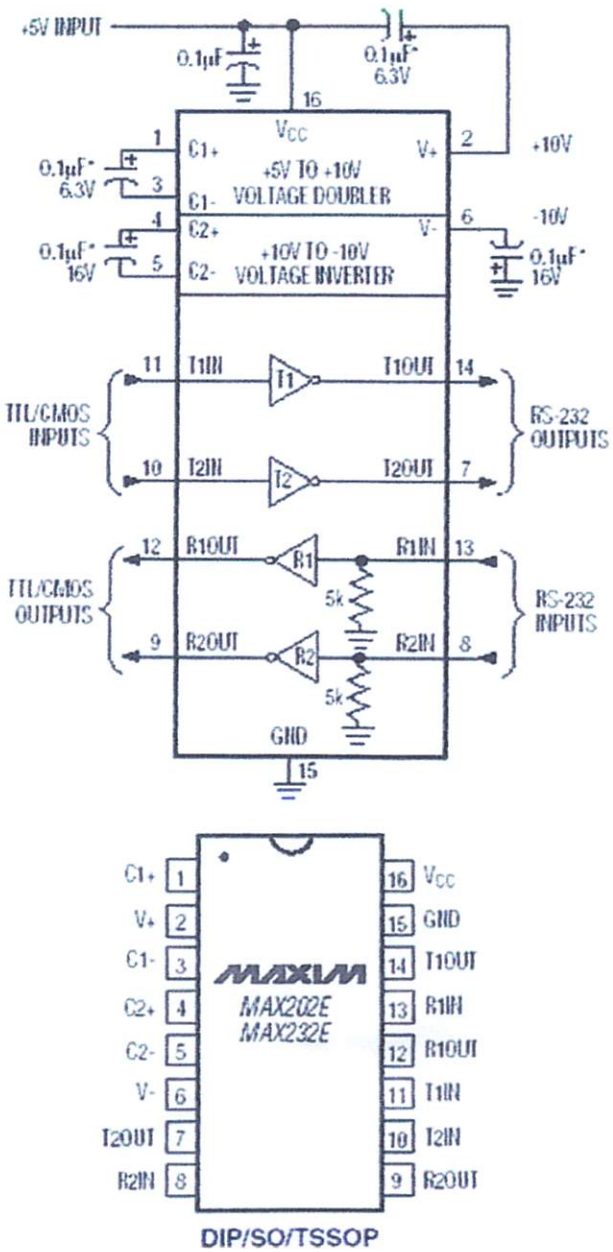
V1 > V2 : maka V0 = High

V1 < V2 : maka V0 = Low

2.5. RS 232

RS-232 (*Recommended Standart number 232*) adalah suatu standar antar muka yang dipublikasikan oleh EISA (*Electronic Industries Association*), untuk interfece antara peralatan terminal data dan peralatan komunikasi data, dengan menggunakan data biner serial sebagai data yang ditransmisikan. IC Max 232 merupakan converter tegangan dari level-level TTL Cmos ke level RS 232. Max 232 ini mempunyai empat buah bagian converter yaitu dua buah driver receiver dan dua buah driver transmitter.Antarmuka RS-232 digunakan dalam banyak aplikasi komunikasi data. RS 232 merupakan salah satu jenis antarmuka *(interface)* dalam proses transfer data antar komputer dalam bentuk serial transfer.

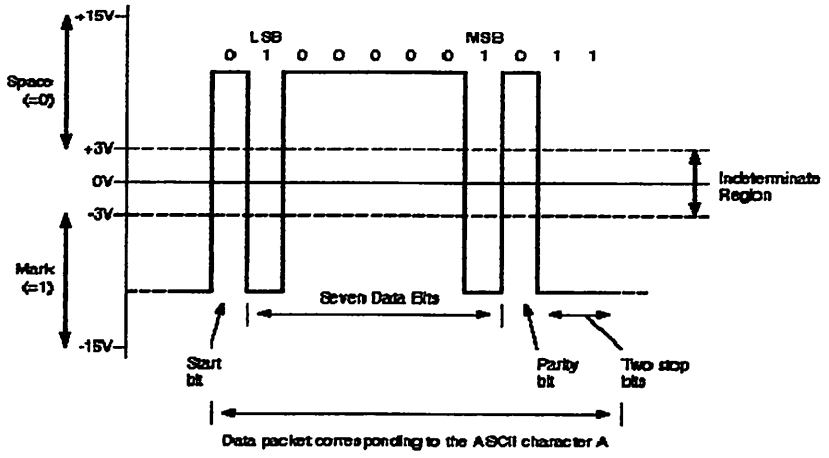
Hal-hal yang perlu diketahui dalam standar antarmuka RS232 antara lain: tegangan standar bagi data *biner* 0 dan 1, sinyal-sinyal yang dipergunakan, dan cara interkoneksi antar RS-232.



Gambar 2-20. Konfigurasi Pin dan blok diagram RS 232

2.5.1. Tegangan RS-232

Untuk menampilkan data biner dibutuhkan dua besaran tegangan. *Biner* 1 atau disebut *mark* dinyatakan dengan tegangan antara –3 V sampai dengan –15 V, sedang biner 0 atau yang disebut *space* dinyatakan tegangan antara +3 V sampai +15 V. tegangan antara –3 V sampai dengan +3 adalah tegangan *invalid.* Tegangan yang mewakili *biner* nol dan satu ditunjukkan dalam gambar 2-21 berikut ini:



Gambar 2-21
Tegangan Yang Mewakili Biner 0 dan 1.

2.5.2. Sinyal-sinyal RS-232

Standar antarmuka RS-232 mendefinisikan sinyal-sinyal yang dipakai baik untuk mentransmisikan/ menerima data maupun untuk proses jabat tangan (*handshaking*). Dalam aplikasi sinyal-sinyal RS-232 ini dihubungkan dengan konektor 25 pin (DB-25) atau 9 pin (DB-9). Pada tabel 2-4 berikut ini diberikan nama sinyal penting beserta hubungannya dengan konektor DB-9 maupun DB-25.

Tabel 2-6 Sinyal-Sinyal Untuk Konektor DB-9 Dan DB-25

Nomor Pin		Nama Sinyal
DB 9	DB 25	
1	8	DCD (*Data Carrier Detect*)
2	3	RD (*Receive Data*)
3	2	TD (*Transmit Data*)
4	20	DTR (*Data Teminal Ready*)
5	7	SG (*Signal Ground*)
6	6	DSR (*Data Set Ready*)
7	4	RTS (*Request To Send*)
8	5	CTS (*Clear To Send*)

2.5.3. Antarmuka Serial RS-232 Pada IBM PC

Pada IBM PC terdapat antarmuka serial yang mengikuti standar antarmuka RS-232. antarmuka ini menggunakan rangkaian terintegrasi UART (*Universal Asyncronous Reciever/ Transmitter*). IBM PC dapat mempunyai beberapa antarmuka serial, dua diantaranya yang paling penting adalah *primary asynchronous communication adaptor* yang disebut COM 1 dan *secondary asynchronous adaptor* yang disebut COM 2.

Tabel 2-7 Address Register RS-232

Nama Register	COM1	COM2
TX Buffer (Transmit Buffer)	03F8H	02F8H
RX Buffer (Receive Buffer)	03F8H	02F8H

Baud rate devisor latch LSB	03F8H	02F8H
Baud rate devisor latch MSB	03F9H	02F9H
Interrupt Enable Register	03F9H	02F9H
Interrupt identification Register	03FAH	02FAH
Line Control Register	03FBH	02FBH
Modem Control Register	03FCH	02FCH
Line Status Rgister	03FDH	02FDH
Modem Status Register	03FEH	02FEH

Fungsi dari beberapa register diantaranya adalah sebagai berikut:

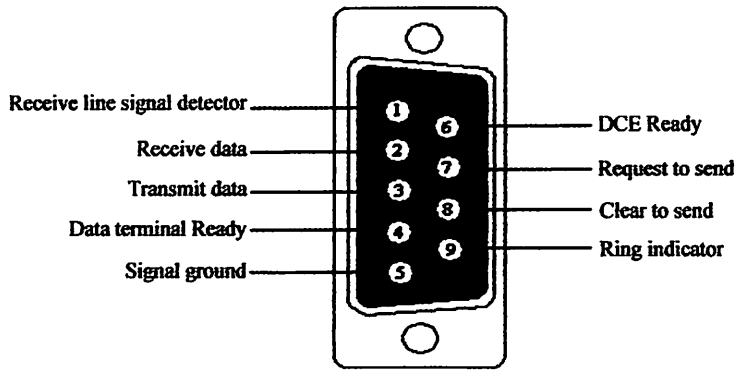
a) TX *buffer*: menampung dan menyimpan data yang akan dikirim keluar. Data ini dikirim oleh CPU ke TX *buffer* setelah mengecek kepastian tentang diperbolehkannya melakukan pengiriman.

b) RX *buffer* : menampung dan menyimpan data yang diterima dari luar. Data itu harus dibaca oleh CPU setelah mengecek kepastian tentang masukannya data.

c) *Baud rate devisor last significant bit*: menampung angka byte bobot rendah untuk pembagi clock yang akan dimasukkan agar didapat baud rate yang diplih. Angka pembagi dapat dipilih antara 01H hingga FFH.

d) *Baud rate devisor most significant bit*: menampung angka *byte* bobot tinggi untuk pembagi *clock* yang akan dimasukkan agar didapat *baud rate* yang dipilih. Angka pembagi dapat dipilih antara 00H hingga FFH.

e)



Gambar 2-22
Skematik DB 9.

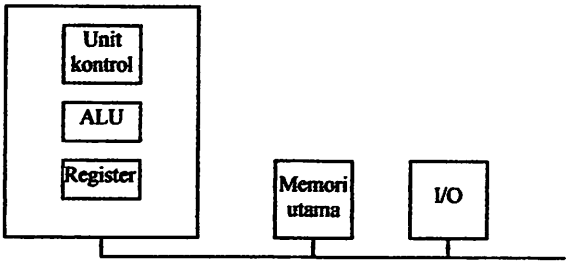
2.6. Personal Komputer

IBM PC mempunyai organisasi dan struktur yang sama dengan PC yang lainya. Diantaranya prosesor, memori utama, memori skunder dan input/ output.

2.6.1. Prosesor

Organisasi sebuah komputer sederhana yang berorientasi pada bus ditampilkan pada gambar 2-23. *Central Processing Unit* (CPU) adalah "otak" dari sebuah komputer. Fungsi CPU adalah menjalankan program-program yang disimpan dalam memori' utama dengan cara mengambil instruksi-instruksi, menguji instruksi tersebut, dan kemudian menjalankanya satu demi satu. Komponen-komponen itu dihubungkan oleh sebuah bus, yaitu sekumpulan kabel kabel paralel untuk mentransmisikan alamat (address), data dan sinyal-sinyal kontrol. Bus dapat berada diluar CPU, yang menghubungkan CPU dengan memori dan peralatan I/O (input/output). CPU terdiri dari beberapa bagian berbeda. Unit kontrol bertanggung jawab mengambil instruksi-instruksi dari memori utama dan menentukan jenis instruksi tersebut. Unit logika aritmatik (ALU) menjalankan operasi-operasi seperti penjumlahan dan Boolean AND.

CPU juga berhasil sebuah memori kecil berkecapatan tinggi yang digunakan untuk menyimpan hasil-hasil sementara dan informasi kontrol tertentu. Memori ini terdiri dari sejumlah register, yang masing-masing memiliki ukuran dan fungsi tersendiri. Biasanya, seluruh register itu memiliki ukuran sama. Setiap register dapat menyimpan satu bilangan, hingga mencapai jumlah maksimum tertentu tergantung pada ukuran register tersebut. Register-register dapat dibaca dan ditulis dengan kecepatan tinggi karena berada dalam CPU. Register yang paling penting adalah Program Counter (PC), yang menunjuk ke instruksi berikutnya yang harus diambil untuk dijalankan. Fungsi penting lainnya adalah *Instruction Register (IR),* yang menyimpan instruksi yang sedang dijalankan.



Gambar 2-23. Organisasi Komputer Sederhana Dengan CPU dan Peralatan

2.6.2. Memori Utama

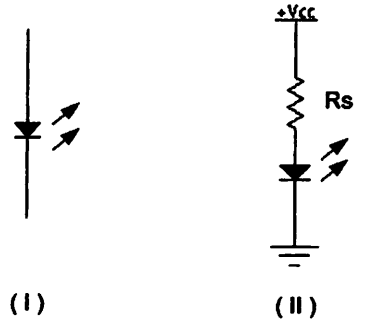
Memori adalah bagian dari komputer tempat program-program dan data data disimpan. Tanpa sebuah memori sebagai tempat untuk mendapatkan informasi guna dibaca dan ditulis oleh mikroprosesor, maka tidak akan ada komputer-komputer digital dengan sistem penyimpanan program.

2.6.3. Memori Skunder

Memori skunder terdiri dari disk magnetik, floppy disk, disk-disk IDE, disk-disk SCSI, RAID, CD-ROM, CD-RW, DVD dan banyak lagi yang lainnya.

2.7. LED (*Light Emitting Dioda*)

Dioda pancar cahaya (*light Emiting Diode* = LED) adalah dioda semikonduktor khusus yang dirancang untuk memancarkan cahaya apabila arus melaluinya. Apabila diberi bias maju, energi elektron yang mengalir diubah menjadi energi cahaya. LED harus dioperasikan di dalam ukuran kerja tegangan dan arus tertentu untuk mencegah kerusakan. Sebagian besar LED membutuhkan 1,2 V sampai 2,5 V dan arus diantara 20 mA sampai 50 mA. Keuntungan utama penggunaan LED sebagai sumber cahaya dibandingkan dengan bola lampu cahaya biasa adalah penggunaan daya yang jauh lebih rendah, jauh lebih lama umurnya, dan beroperasi dengan kecepatan tinggi. Dioda konvensional mengubah energi menjadi panas. Dioda *arsenide gallium* mengubah energi menjadi panas dan sinar inframerah.



Gambar 2-24

Lambang skematis dan contoh rangkaian dari LED

Sumber: Malvino, 1985: 97

Cara mengendalikan LED yaitu dengan memperhitungkan arus dan tegangan sumber. Besar arus LED yang diberikan dalam contoh rangkaian Gambar 2.19 adalah (Malvino, 1985:97):

$$I = \frac{Vcc - V_{IR}}{R_s} \quad \text{.....2-17}$$

$$R_s = \frac{Vcc - V_{IR}}{I} \quad \text{.....2-18}$$

2.7. WIRELESS-CAMERA

Kamera disini berfungsi untuk mengambil gambar dengan spesifikasi seperti berikut:

TV system : NTSC / EIA

Frekuwensi : 900 MHz, 1200 MHz, 2400 MHz

Minimal illumination : 2 LUX

Output power : 200 mW

Camera power : DC +8V 200 mA

Reciver power : DC +9V 500 mA

Image Pickup : 1/3, ¼

Scan Frequency : NTSC/EIA: 60Hz

BAB III

PERENCANAAN DAN PEMBUATAN ALAT

Dalam bab ini akan dibahas perancangan dan pembuatan alat. Pembahasan akan dilakukan pada setiap blok rangkaian, cara kerja masing-masing blok rangkaian, perhitungan dan fungsi masing-masing blok rangkaian tersebut. Secara garis besar terdapat dua bagian perangkat yang ada yaitu :

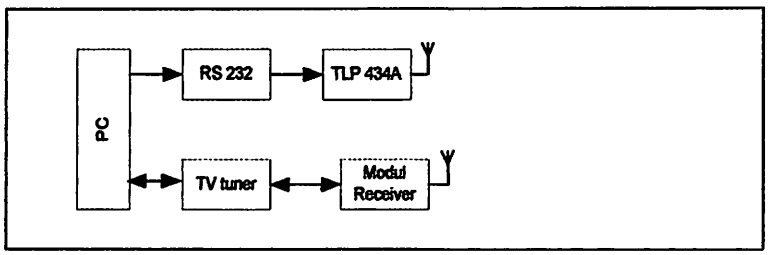
- Perancangan perangkat keras *(Hardware).*
- Perancangan perangkat lunak *(Software).*

Pada perancangan perangkat keras akan meliputi seluruh *peripheral* yang digunakan pada sistem ini. Pada perancangan perangkat lunak akan meliputi diagram alir dan *software* secara umum, akan tetapi kedua perangkat ini dalam kerjanya akan saling menunjang satu sama lain.

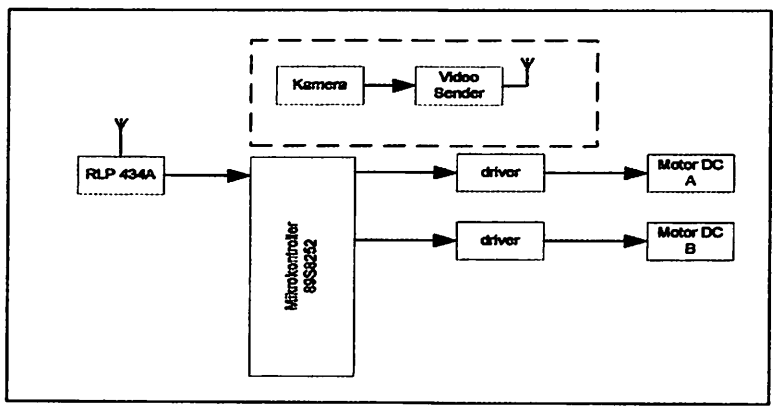
Peralatan yang digunakan adalah satu set PC (personal computer) ketika kita menekan keyboard untuk memasukkan data sebagai inputan, maka kemudian data akan dipancarkan oleh TLP434A sebagai transmitter dan selanjutnya data diterima oleh RLP434A sebagai receiver dan kemudian akan diteruskan ke mikrokontroller AT89S8252 untuk diproses sebagai inputan yang akan mengaktifkan motor dc.

3.1. Blok Diagram Keseluruhan Sistem

PERSONAL COMPUTER



HELICOPTER MODEL



Gambar 3-1 Diagram Blok Sistem

Keterangan Blok Diagram Sistem:

- *Keyboard PC*

  Alat pengendali gerak helicopter (kemudi helicopter) dari jauh.

- TLP434A

  Berfungsi sebagai pengirim data melalui frekuensi tertentu radio frekuensi (433,92 MHz). Digunakan *hardware* TLP434A dengan spesifikasi, 4 pin dengan fungsi yaitu 1 pin sebagai Vcc, 1 pin sebagai *ground*, 1 pin sebagai data digital *input* dan 1 pin sebagai RF *output*.

- RLP434A

  Berfungsi sebagai penerima data melalui frekuensi tertentu (433,92 MHz). Digunakan *hardware* RLP434A dengan spesifikasi, 8 pin dengan fungsi yaitu 2 pin sebagai Vcc, 3 pin sebagai *ground*, 1 pin sebagai data *digital output*, 1 pin *linear output*, dan 1 pin untuk antena.

- IC HT648L

  IC Decoder yang berfungsi memproses 1 inputan data menjadi beberapa data outputan dari *receiver*. Digunakan *Decoder* HT648L.

- MIKROKONTROLLER AT89S8252

  Berfungsi sebagai pengontrol semua system yang memiliki spesifikasi 8Kbytes PEROM 256bytes RAM internal,32 Programmable I/O lines,@Kbytes EEPROM dan 3 buah timer counter 16 bit.

- *Driver*

  Rangkaian untuk mengendalikan motor dc.

- Motor DC (A/PRIMER)

  Sebagai penggerak baling-baling atas helicopter dengan berbagai versi kecepatan untuk bergerak atas dan bawah.

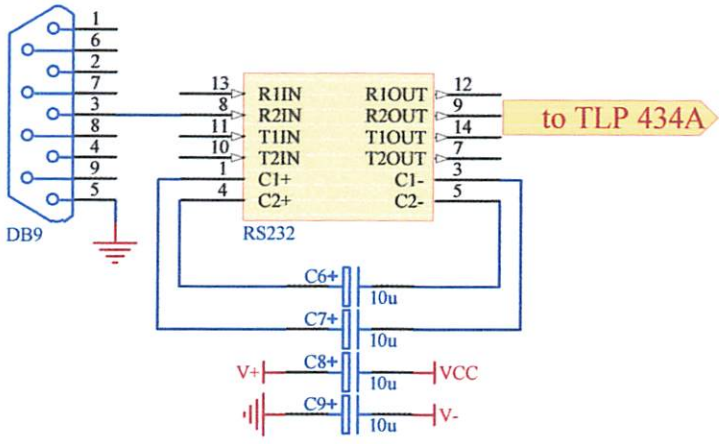
- Motor DC (B/SECONDARY)

  Sebagai penggerak baling-baling belakang yang hanya dapat berbelok ke kanan atau ke kiri dan sebagai penyeimbang.

3.2. Perancangan Perangkat Keras (Hardware)

3.2.1. Perancangan Komunikasi Antara Alat Dengan PC

Komunikasi antara alat dengan PC akan dilakukan secara serial menggunakan *port* serial (*Com*) dari PC. Komunikasi tersebut dirancang menggunakan *baud rate* 9600 bps tanpa paritas, 1 bit start, 8 bit data, dan satu bit stop. Dikarenakan standart komunikasi serial dari PC menggunakan *standart* komunikasi RS232 yang mimiliki *range* logika tinggi (H) adalah sebesar -15 Volt sampai –3 Volt, dan logika rendah (L) sebesar 3 Volt sampai 15 Volt. Sedangkan mikrokontroller menggunakan logika TTL yang memiliki logika tinggi (H) sebesar 3 volt sampai 5 volt, dan logika rendah (L) 0 volt sampai 0,45 volt, maka dibutuhkan rangkaian penyesuai kedua kondisi logika tersebut. Perangkat utama dari rangkaian penyesuai kondisi logika RS232 dengan kondisi logika TTL adalah IC MAX 232. Perancangan rangkaian selengkapnya ditetapkan pada *data sheet* seperti pada gambar di bawah ini:

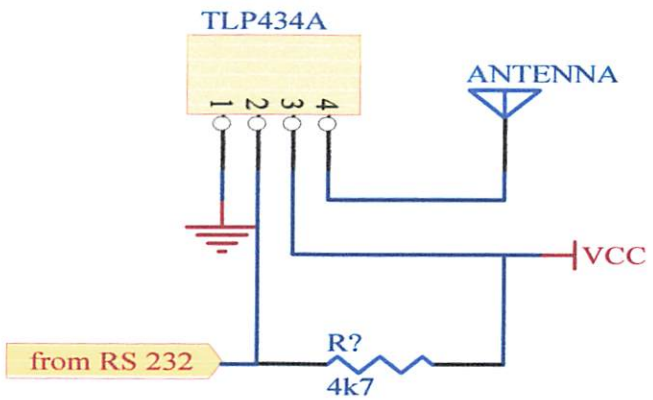


Gambar 3-2
Perancangan Rangkaian RS 232

3.2. 2. Perancangan TLP434A

Untuk pengiriman data digunakan TLP434A yang bekerja melalui gelombang yang mempunyai frekuensi 433.99MHz. Alat ini sangat mudah digunakan karena tidak banyak membutuhkan banyak komponen pendukung, serta membutuhkan tegangan yang relatif kecil yaitu 2-5 Volt. Konfigurasi pin – pin TLP434A dalam perancangan adalah :

1. Kaki 1 dihubungkan dengan *ground*
2. Kaki 2 dihubungkan dengan pin $R2_{OUT}$ dari RS232 sebagai data *input*
3. Kaki 3 dihubungkan dengan sumber tegangan
4. Kaki 4 dihubungkan dengan antenna

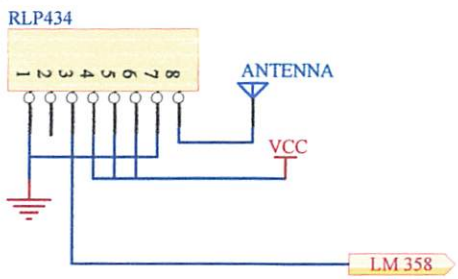


Gambar 3-3 Rangkaian TLP434A

3.2.3. Perancangan RLP434A

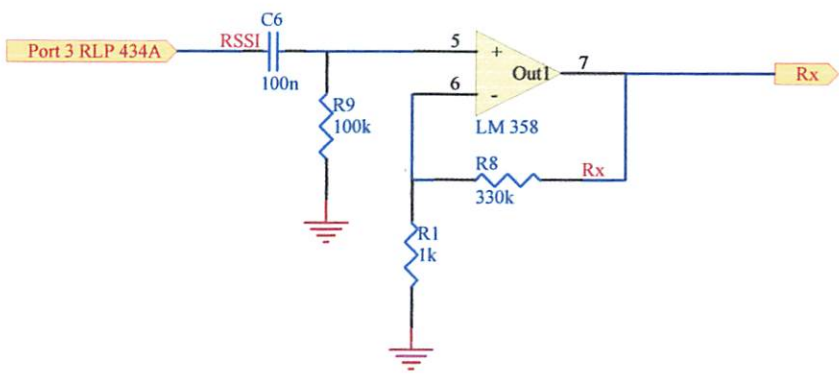
Konfigurasi perancangan pin-pin RLP434A adalah :

1. Kaki 1 dan 7 dihubungkan dengan *ground*
2. Kaki 2 sebagai *output* ke mikrokontroller
3. Kaki 3 yaitu pin RSSI dihubungkan dengan pin comparator
4. Kaki 4, 5, dan 6 dihubungkan dengan Vcc
5. Kaki 8 dihubungkan dengan antena.



Gambar 3-4 Rangkaian RLP 434A

3.2.4. Perancangan Penguatan



Gambar 3-5 Rangkaian Penguat Tak Membalik *(Non Inverting)*

Keluaran dari receiver RLP 434A diumpankan ke penguat non inverting. Fungsi dari penguatan ini adalah untuk menguatkan amplitudo sinyal keluaran receiver RLP 434A.

Dalam perancangan ini besar penguatan yang dibutuhkan adalah 331 kali, maka untuk mencari nilai RF jika nilai R1 sudah ditentukan sebesar 1KΩ adalah:

$$A = \frac{Vout}{Vin} = \left[\left(\frac{Rf}{R1}\right) + 1\right]$$

$$A = \left[\left(\frac{330k}{1k}\right) + 1\right]$$

$$= 331$$

$$Av = \frac{Rf}{R1} + 1$$

$$331 = \frac{Rf}{1K} + 1$$

$$331 = \frac{Rf}{1K} + \frac{1K}{1K}$$

$$331 = \frac{Rf + 1K}{1K}$$

$$331K = Rf + 1K$$

$$Rf = 331K - 1K$$

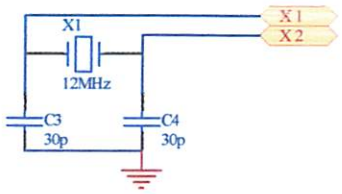
$$= 330\ \text{K}\Omega$$

3.2.5. Mikrokontroler AT89S8252

3.2.5.1. Perancangan Rangkaian Clock

Kecepatan proses yang dilakukan oleh mikrokontroller ditentukan oleh sumber *clock* yang mengendalikan mikrokontroller tersebut. Sistem yang dirancang ini menggunakan osilator internal yang telah tersedia dalam *chip* AT89S8252. Untuk menentukan frekuensi osilatornya cukup dengan menghubungkan kristal dalam pin 19 (X1) dan pin 18 (X2) serta dua buah kapasitor ke *ground*.

Besarnya kapasitansinya disesuaikan dengan spesifikasi dalam lembar data AT89S8252 yaitu 33pF. Kristal yang digunakan adalah 12 MHz. gambar 3-8 memperlihatkan rangkaian *clock* yang dirancang.

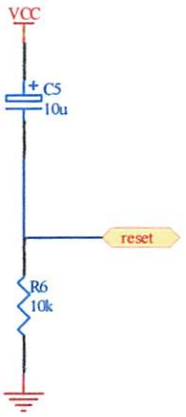


Gambar 3-6 Rangkaian Clock untuk MCU AT89S8252

Dalam sistem ini, rangkaian terdiri dari dua buah kapasitor sebesar 33 pF dan sebuah kristal sebesar 12 MHz.

3.2.5.2 Perancangan Rangkaian Reset

Pin reset pada mikrokontroller merupakan masukan aktif high (1). Pulsa transisi dari low (0) dan high (1) akan mereset mikrokontroller menuju alamat 0000H. pin reset dihubungkan dengan rangkaian power on reset yang diperlihatkan pada gambar 3-9 berikut ini:



Gambar 3-7 Rangkaian Reset untuk MCU AT89S8252

Karena kristal yang digunakan mempunyai frekuensi sebesar 11,0592 MHz maka satu periode membutuhkan waktu sebesar:

$$T = \frac{1}{f_{XTAL}}$$

$$= \frac{1}{11{,}0592MHz} S$$

$= 9{,}042 \times 10^{-8}$ S

Sehingga waktu minimal logika tinggi yag dibutuhkan untuk me-*reset* mikrokontroller adalah:

Reset (min) = T x periode yang dibutuhkan

$= 9{,}042 \times 10^{-8} \times 24$

$= 2{,}17$ µS

Mikrokontroller membutuhkan waktu minimal 2,17 µS untuk me-*reset*. Waktu minimal inilah yang dijadikan pedoman untuk menentukan nilai R dan C. dari persamaan diatas dengan menentukan nilai R = 10 KΩ dan C = 10 µF, maka:

$T = 0{,}357 * R * C$

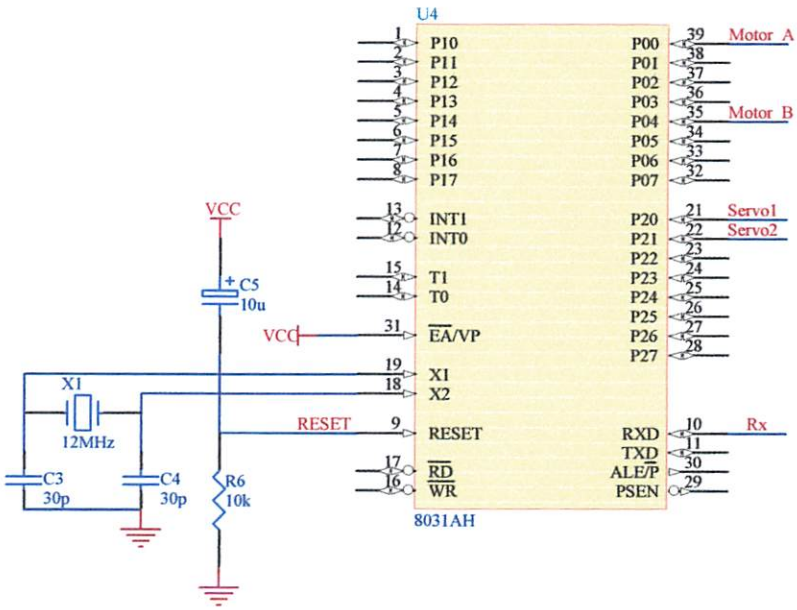
$= 0{,}357 * 10 \times 10^{6}\ \Omega * 10 \times 10^{-6}$

$= 35{,}7$ mS.

Jadi dengan nilai komponen R = 10 KΩ dan C = 10 µF dapat memenuhi syarat minimal untuk waktu yang dibutuhkan oleh mikrokontroller.

3.2.5.3 Perancangan Penggunaan Port pada Mikrokontroller AT89S8252

Dalam perancangan dan pembuatan rangkaian pada alat pengirim gambar dari helicopter model yang dikontrol oleh personal computer ini menggunakan mikrokontroller AT89S8252 bertujuan untuk mendapatkan rangkaian yang praktis dan ringkas. IC mikrokontroller AT89S8252 ini sendiri memiliki fasilitas EEPROM sebagai memory program, sehingga sistem alat menjadi lebih sederhana. Adapun aplikasi mikrokontroller AT89S8252 dalam perancangan ini adalah sebagai berikut:

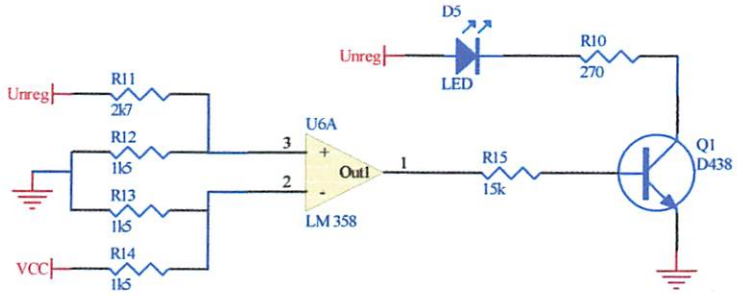


Gambar 3-8 Rangkaian MCU AT89S8252

Mikrokontroller AT89S8252 memiliki 4 port paralel yang masing-masing digunakan sebagai berikut:

- Port 0 (P0.0) digunakan sebagai port data untuk pengendali kecepatan motor A.
- Port 0 (P0.4) digunakan sebagai port data untuk pengendali kecepatan motor B.
- Port 2.0 digunakan sebagai port untuk pengendali gerak servo 1.
- Port 2.1 digunakan sebagai port untuk pengendali gerak servo 1.
- Port 3.0 (RXD) digunakan untuk receiver RLP 434A.

3.2.6. Perancangan Komparator Tegangan



Gambar 3-9 Rangkaian Komparator Tegangan

Comparator disini berfungsi untuk membandingkan masukan positif (+) dan negatif (-).

Jika positif lebih besar dari negatif (+ > -) maka keluaran akan terjadi saturasi positif (+),dan sebaliknya, jika positif lebih kecil dari negatif (+ < -) maka keluaran akan terjadi saturasi negatif (-)

$$Vb = \frac{R2}{R1 + R2} \bullet Va$$

$$Vi.1 = \frac{1{,}5 \bullet 10^3}{1{,}5 \bullet 10^3 + 1{,}5 \bullet 10} \bullet Vcc$$

$$= 330\ \text{K}\Omega$$

$$Vi.1 = \frac{1{,}5 \bullet 10^3}{1{,}5 \bullet 10^3 + 2{,}7 \bullet 10} \bullet Vbat$$

$$= 0{,}36\ \text{Vbat}$$

$$\therefore Vi.1 = Vi.2$$

$$2{,}5 = 0{,}36 \bullet Vbat$$

$$Vbat = \frac{2{,}5}{0{,}36}$$

$$= 6{,}94\ \text{V}$$

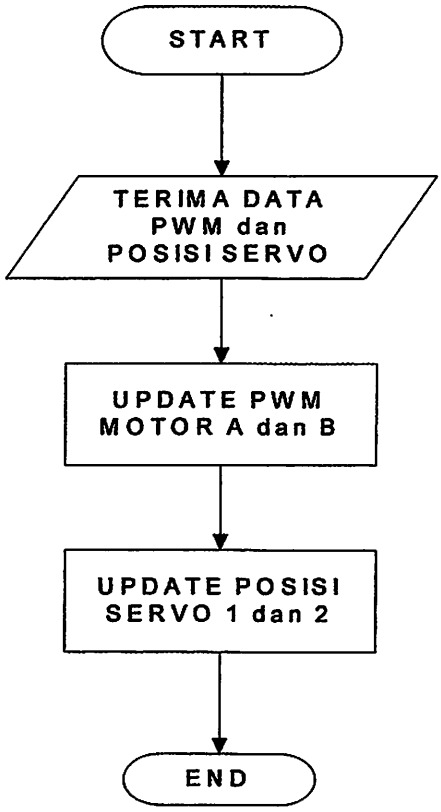
3.3. Perancangan Perangkat Lunak (Software)

Untuk mendukung agar perangkat keras berfungsi sesuai dengan perencanaan, maka diperlukan perangkat lunak sebagai penunjangnya. Perangkat lunak ini sendiri maksudnya adalah suatu program yang kita buat yang nantinya akan ditanam kedalam mikrokontroller AT89S8252. setelah mikrokontroller tersebut diprogram, maka akan diketahui apakah program yang telah kita buat bekerja sesuai dengan yang kita rencanakan ataukah masih memiliki kesalahan.

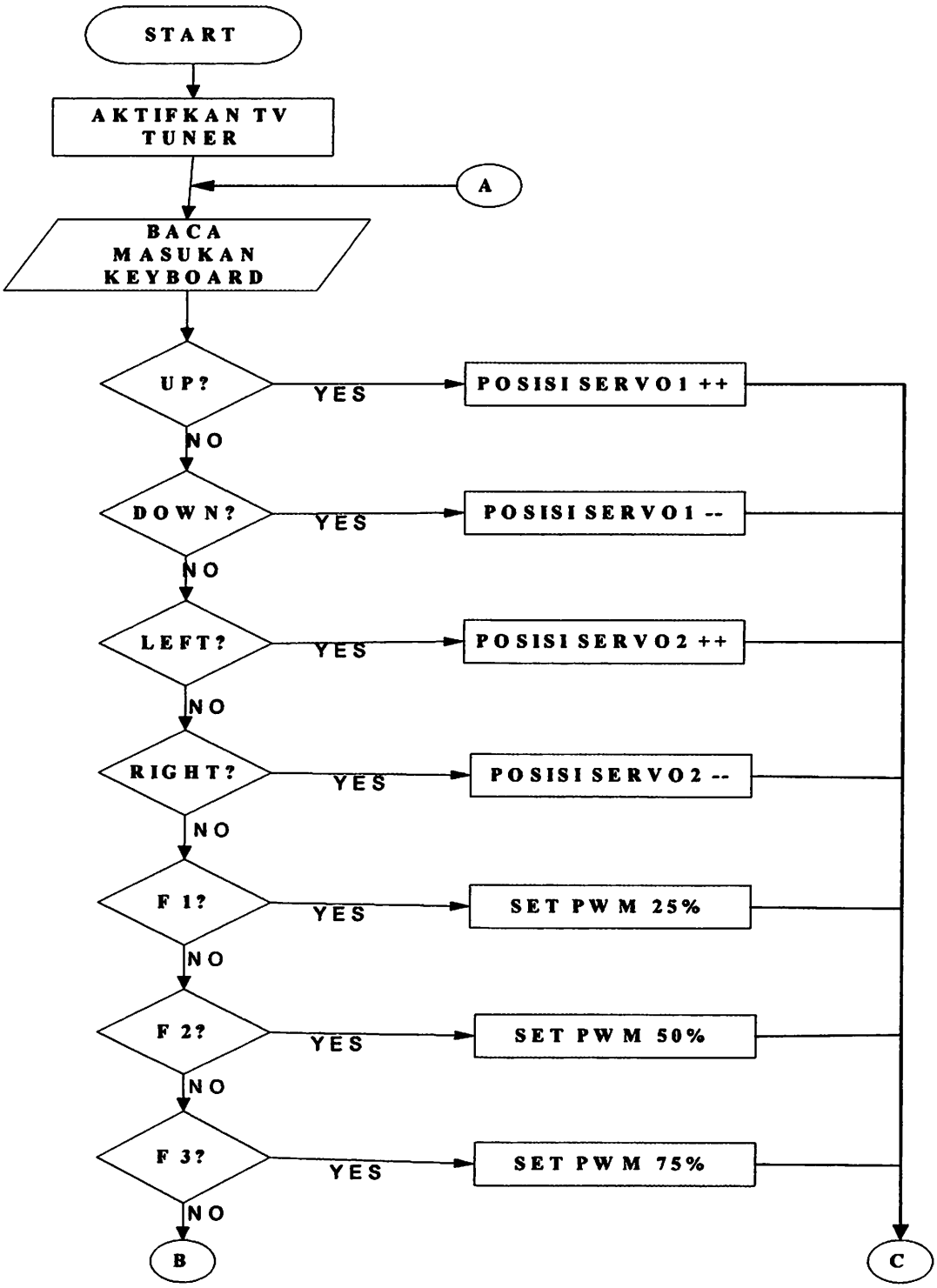
Sistem aplikasi mikrokontroller AT89S8252 ini dapat mengatur dan mengendalikan keseluruhan sistem apabila ada urutan instruksi yang mendefinisikan secara jelas urutan tugas yang harus dikerjakan. Urutan instruksi ini sangat penting untuk didefinisikan, karena mikrokontroller dapat bekerja secara pasti sesuai dengan instruksi yang telah dibuat.

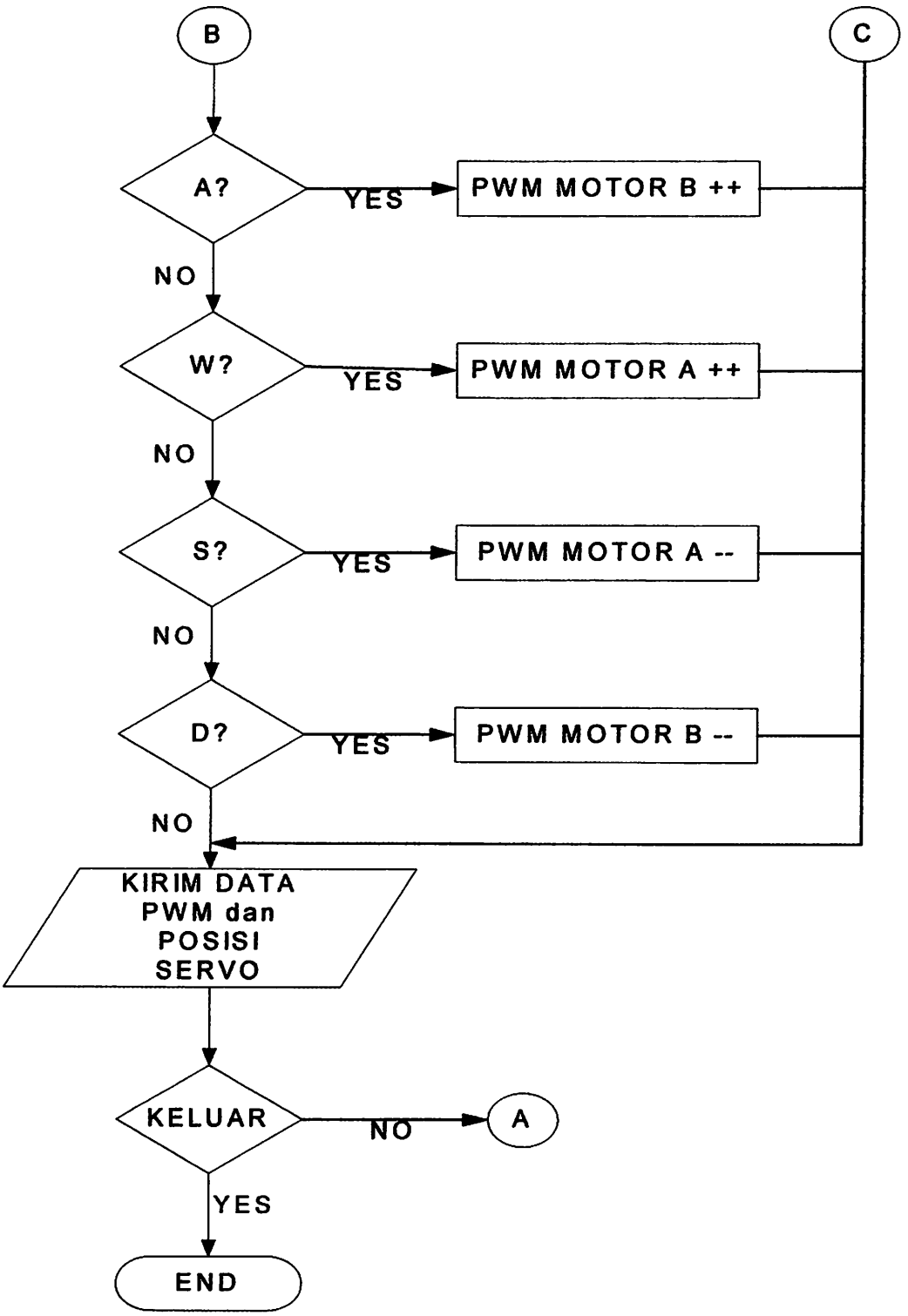
3.4. FLOWCHART

✓ Flowchart program microcontroller



✓ Flowchart pada PC




B
C
A?
YES
PWM MOTOR B ++
NO
W?
YES
PWM MOTOR A ++
NO
S?
YES
PWM MOTOR A --
NO
D?
YES
PWM MOTOR B --
NO
KIRIM DATA
PWM dan
POSISI
SERVO
KELUAR
NO
A
YES
END

BAB IV

ANALISA DAN PENGUJIAN ALAT

Pada bab ini membahas cara pengujian dan analisa dari alat yang dirancang, sehingga dapat diketahui apakah alat tersebut dapat bekerja sesuai dengan yang telah direncanakan. Dalam rangka pengujian alat tersebut diuraikan percobaan yang dilakukan untuk mengetahui respon dari keseluruhan alat yang telah dirancang.

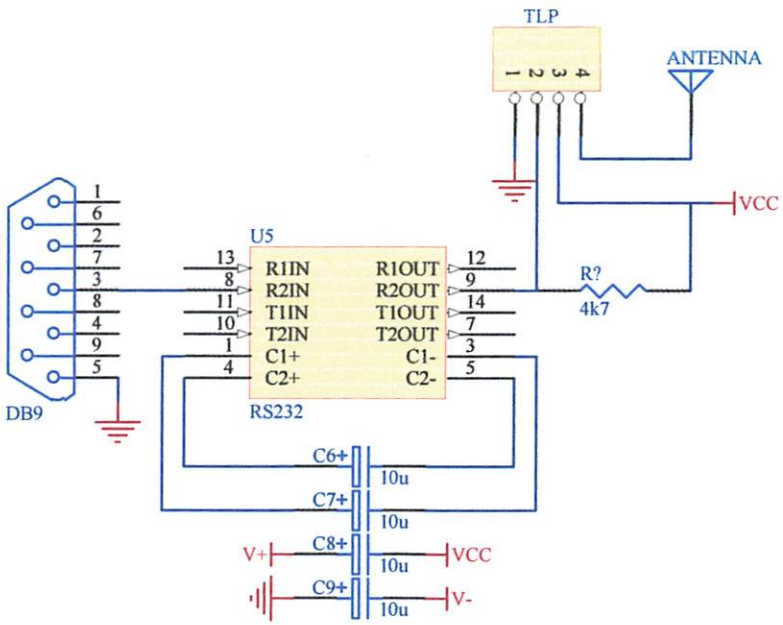
Untuk mengetahui kemampuan alat dan sistem kerja sesuai dengan program yang telah dibuat maka dilakukan pengujian pada alat dan sistem kerja alat dengan prosedur pengujian perblok.

4.1. Pengujian TLP434A dan RLP434A

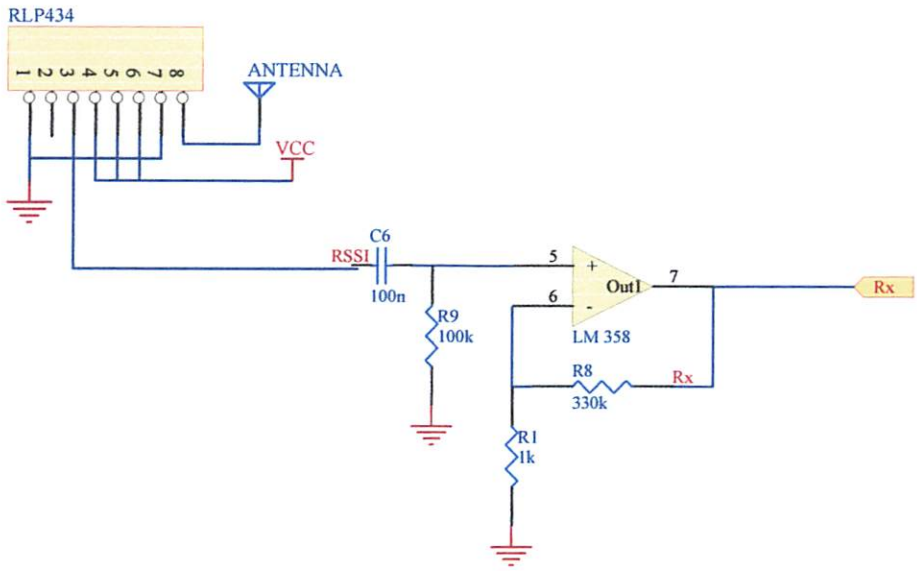
4.1.1. Tujuan

Tujuan pengujian TLP434A dan RLP434A adalah untuk melihat apakah kedua komponen ini memiliki keadaan yang baik untuk mentransfer data dan untuk mengetahui bentuk gelombang yang dihasilkan dengan menggunakan bantuan osiloscope. Pada pengujian ini dilakukan pengiriman data dan melihat outputannya. Dilakukan juga pengujian terhadap jarak yang mampu dicapai oleh TLP434A dan RLP434A dengan halangan maupun tanpa halangan.

4.1.2. Rangkaian Pengujian



Gambar 4-1 Rangkaian Blok Pengiriman



Gambar 4-2 Rangkaian Blok Penerimaan

4.1.3. Peralatan Yang Digunakan

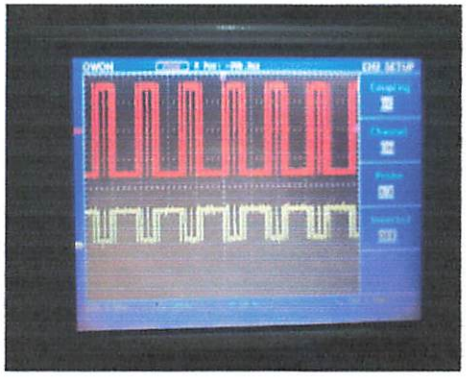
1. Sumber tegangan 5V
2. Rangkaian pengujian
3. Multimeter Digital
4. Oscilloscope

4.1.4. Langkah-Langkah Pengujian

1. Hubungkan pin Vcc dan *Ground* pada rangkaian pengujian dengan sumber tegangan 5 volt.
2. Memberikan inputan pada *keyboard* melihat outputannya pada pin data TLP 434A dann RLP434A.
3. Melihat bentuk gelombang keluaran pada oscilloscope.
4. Memisahkan blok pengirim dan blok penerima dengan jarak yang telah ditentukan di ruang terbuka (tanpa halangan).
5. Memisahkan blok pengirim dan blok penerima dengan jarak yang telah ditentukan di ruang tertutup (dengan halangan).
6. Mencatat hasil pengamatan pada tabel.

4.1.5. Hasil Pengujian

4.1.5.1. Hasil Pengujian *Output* Gelombang



Ket : Sinyal Merah = Output dari Pc/RS 232

Sinyal Kuning = input TLP434A

Gambar 4-3 Keluaran Sinyal TLP

4.1.5.2. Hasil Pengujian Pengiriman dan Penerimaan Tanpa Halangan

Tabel 4-1 Hasil Pengujian Jarak Pengiriman dan Penerimaan Data Dengan TLP434A dan RLP434A Tanpa Halangan

Tombol	Jarak (m)				
	1-10	11-20	21-30	31-40	41-50
W	sukses	sukses	sukses	sukses	gagal
S	sukses	sukses	sukses	sukses	gagal
D	sukses	sukses	sukses	sukses	gagal
A	sukses	sukses	sukses	sukses	gagal
F1	sukses	sukses	sukses	sukses	gagal
F2	sukses	sukses	sukses	sukses	gagal
F3	sukses	sukses	sukses	sukses	gagal
C	sukses	sukses	sukses	sukses	gagal
C&W	sukses	sukses	sukses	sukses	gagal
C&S	sukses	sukses	sukses	sukses	gagal

C&D	sukses	sukses	sukses	sukses	gagal
C&A	sukses	sukses	sukses	sukses	gagal
ↄ&W	sukses	sukses	sukses	sukses	gagal
ↄ&S	sukses	sukses	sukses	sukses	gagal
ↄ&D	sukses	sukses	sukses	sukses	gagal
ↄ&A	sukses	sukses	sukses	sukses	gagal

Keterangan:

Sukses : data yang dikirim berhasil atau terdeteksi oleh penerima dan *driver* bergerak sesuai dengan yang direncanakan.

Gagal : data yang dikirim tidak terdeteksi oleh penerima.

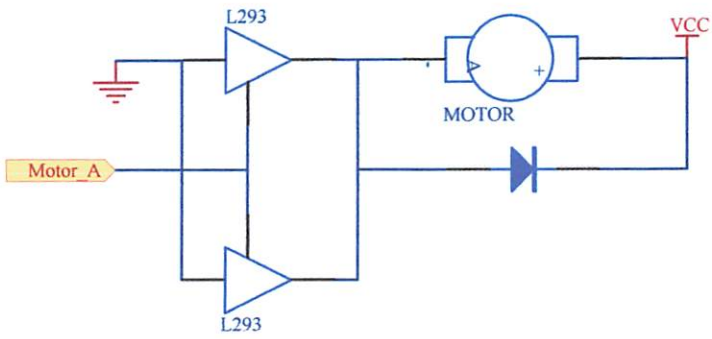
Dari hasil pengujian jarak pengiriman dan penerimaan tanpa halangan dapat di simpulkan bahwa jarak maksimal yang dicapai adalah ±40 meter dengan kondisi baterai terisi penuh dan antena (tembaga) panjang ±18cm dan diameter ±0,5mm.

4.1.5.3. Hasil Pengujian Pengiriman dan Penerimaan Dengan Halangan

Tabel 4-2 Hasil Pengujian Jarak Pengiriman dan Penerimaan Data Dengan TLP434A dan RLP434A Dengan Halangan

Tombol	Jarak (m)		
	1-10	11-20	21-22
W	sukses	sukses	gagal
S	sukses	sukses	gagal
D	sukses	sukses	gagal
A	sukses	sukses	gagal
F1	sukses	sukses	gagal
F2	sukses	sukses	gagal
F3	sukses	sukses	gagal

4.2.1.2. Rangkaian Pengujian



Gambar 4-4 Rangkaian Pengujian Motor DC

4.2.1.3. Peralatan yang digunakan

Peralatan yang digunakan adalah:

- *Power supply* 5 volt DC dan 12 volt DC
- *Digital* multimeter
- Rangkaian *driver* motor DC (A)

4.2.1.4. Langkah Pengujian

Langkah-langkah pengujian antara lain:

- Rangkaian diberi catu daya 5 volt DC
- Motor DC diberi catu daya 12 volt DC
- Dilakukan pengukuran tegangan pada rangkaian pengujian *driver* motor DC menggunakan multimeter digital.

4.2.1.5. Hasil Pengujian

Besar tegangan *output* pada mikrokontroller dengan keluaran *low* dan *high* dapat dilihat pada gambar dibawah ini:

(a) (b)

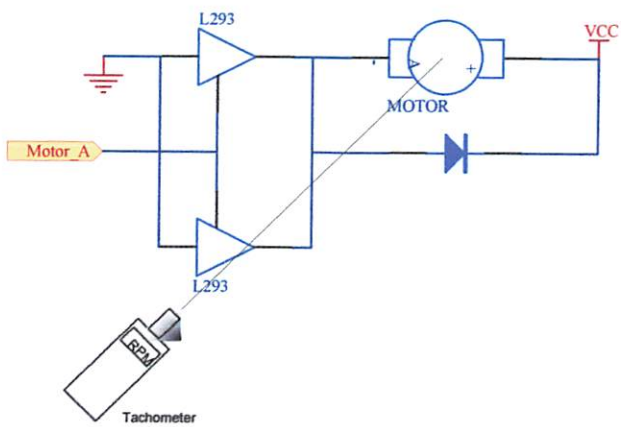
Gambar 4-5 Kondisi Keluaran Port Mikrokontroller
(a) *High*, (b) *Low*

4.3. Pengujian Putaran Motor DC

4.3.1. Tujuan

Pengujian ini bertujuan untuk mengetahui putaran pada motor dc dan mengetahui bentuk gelombang dari PWM.

4.3.2. Rangkaian Percobaan



Gambar 4-6 Rangkaian Pengujian Putaran Motor DC

⊂&D	sukses	sukses	sukses	sukses	gagal
⊂&A	sukses	sukses	sukses	sukses	gagal
⊃&W	sukses	sukses	sukses	sukses	gagal
⊃&S	sukses	sukses	sukses	sukses	gagal
⊃&D	sukses	sukses	sukses	sukses	gagal
⊃&A	sukses	sukses	sukses	sukses	gagal

Keterangan:

Sukses : data yang dikirim berhasil atau terdeteksi oleh penerima dan *driver* bergerak sesuai dengan yang direncanakan.

Gagal : data yang dikirim tidak terdeteksi oleh penerima.

Dari hasil pengujian jarak pengiriman dan penerimaan tanpa halangan dapat di simpulkan bahwa jarak maksimal yang dicapai adalah ±40 meter dengan kondisi baterai terisi penuh dan antena (tembaga) panjang ±18cm dan diameter ±0,5mm.

4.1.5.3. Hasil Pengujian Pengiriman dan Penerimaan Dengan Halangan

Tabel 4-2 Hasil Pengujian Jarak Pengiriman dan Penerimaan Data Dengan TLP434A dan RLP434A Dengan Halangan

Tombol	Jarak (m)		
	1-10	11-20	21-22
W	sukses	sukses	gagal
S	sukses	sukses	gagal
D	sukses	sukses	gagal
A	sukses	sukses	gagal
F1	sukses	sukses	gagal
F2	sukses	sukses	gagal
F3	sukses	sukses	gagal

C&W	sukses	sukses	gagal
C&S	sukses	sukses	gagal
C&D	sukses	sukses	gagal
C&A	sukses	sukses	gagal
Ↄ&W	sukses	sukses	gagal
Ↄ&S	sukses	sukses	gagal
Ↄ&D	sukses	sukses	gagal
Ↄ&A	sukses	sukses	gagal

Keterangan:

Sukses : data yang dikirim berhasil atau terdeteksi oleh penerima dan *driver* bergerak sesuai dengan yang direncanakan.

Gagal : data yang dikirim tidak terdeteksi oleh penerima.

Dari hasil pengujian jarak pengiriman dan penerimaan dengan halangan dapat di simpulkan bahwa jarak maksimal yang dicapai adalah ± 20 meter dengan kondisi baterai terisi penuh.

4.2. Pengujian Rangkaian *Driver* Motor DC

4.2.1. Pengujian Rangkaian *Driver* Motor DC

4.2.1.1. Tujuan

Mengetahui apakah *driver* Motor DC dapat bekerja dengan baik sesuai yang diharapkan dan mengetahui nilai tegangan pada rangkaian *driver* motor DC.

4.3.3. Peralatan yang digunakan

Peralatan yang digunakan adalah:

- *Power supply* 5 volt DC dan 8,4 volt DC
- Rangkaian *driver* motor dc
- Digital Hand Optical Tachometer
- Oscilloscope

4.3.4. Langkah Pengujian

Langkah-langkah pengujian antara lain:

- Rangkaian diberi catu daya 5 volt DC
- Motor DC diberi catu daya 8,4 Volt DC
- Melakukan pengukuran putaran motor dc dengan Tachometer
- Melihat bentuk keluaran gelombang dengan oscilloscope

4.3.5. Hasil Pengujian

4.3.5.1. Hasil Pengujian Pada Oscilloscope

Hasil pengujian putaran Motor DC pada oscilloscope adalah sebagai berikut:

Tabel 4-3 Hasil Perhitungan Perancangan *Duty Cycle* Motor DC

Tombol	Duty Cycle (%)	Sinyal Analog (V)
F1	25%	6
F2	50%	4,1
F3	75%	2,42

Sinyal analog diatas merupakan tegangan yang dihasilkan dari besarnya *duty cycle* dalam 1 periode, besarnya tegangan tersebut dapat dicari dengan menggunakan perhitungan sebagai berikut:

$$V = \frac{t_1 \bullet V_1 + t_2 \bullet V_2}{t_1 + t_2}$$

Diketahui Vdd adalah 8,4 Volt, maka;

- Untuk tombol "F❶", $V = \frac{(25\% \bullet 0) + (75\% \bullet 8{,}4)}{25\% + 75\%}$

  $= 6{,}3$ volt

- Untuk tombol "F❷", $V = \frac{(50\% \bullet 0) + (50\% \bullet 8{,}4)}{50\% + 50\%}$

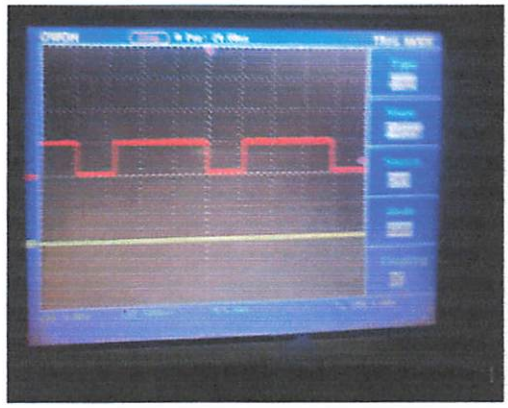
  $= 4{,}2$ volt

- Untuk tombol "F❸", $V = \frac{(75\% \bullet 0) + (25\% \bullet 8{,}4)}{75\% + 25\%}$

  $= 2{,}1$ volt

Gambar 4-7 PWM Dengan *Duty Cycle* 25%

Gambar 4-8 PWM Dengan *Duty Cycle* 50%



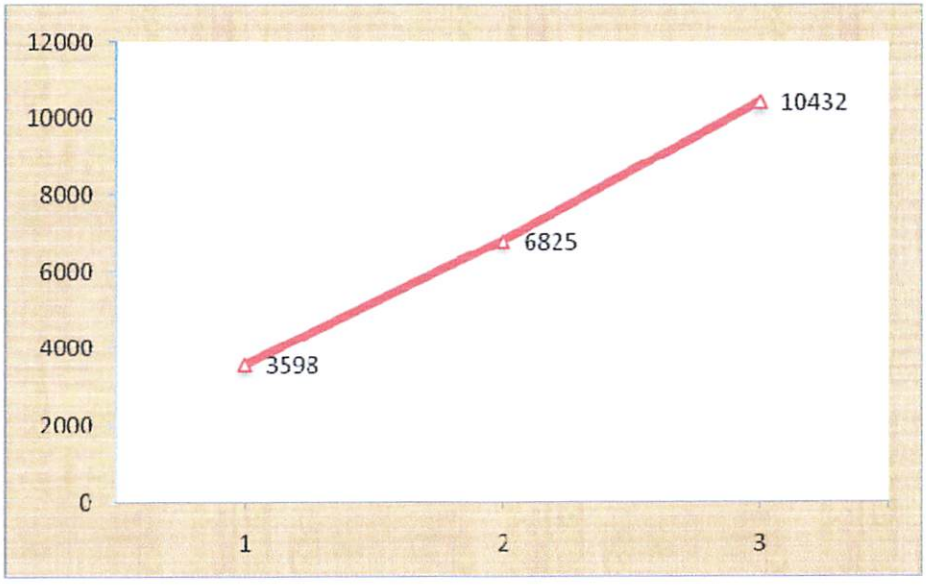
Gambar 4-9 PWM Dengan *Duty Cycle* 75%

4.3.5.2. Hasil Pengujian Menggunakan Tachometer

Pada pengujian putaran motor dc menggunakan Digital Stroboskop Tachometer di dapat hasil sebagai berikut:

Tabel 4-4 Hasil Pengukuran Putaran Motor

Tombol	PWM (%)	Percobaan Pengukuran Putaran Motor DC (rpm)					Rata-rata (rpm)
		Pertama	Kedua	Ketiga	Keempat	Kelima	
F1	25	3598	3597	3598	3599	3598	3598
F2	50	6825	6826	6826	6825	6825	6825
F3	75	10432	10422	10435	10432	10432	10432



Grafik 4-1 Pengukuran PWM Terhadap Putaran Motor DC

4.4. Perancangan Putaran Motor DC

Motor DC mempunyai putaran (rpm) yang bervariasi. Dalam perancangan ini, terdapat tiga kondisi putaran motor dc yang sesuai dengan penekanan tombol masing-masing. Sebelumnya, harus mengetahui putaran maksimal motor dc yang ditentukan dengan melakukan percobaan dengan peralatan sebagai berikut:

- Catu daya (Baterai 8,4V)
- Rangkaian *Driver* Motor DC
- Motor DC
- Digital Stroboskop Tachometer *range* 20-20.000rpm

Tabel 4-5 Putaran Motor DC

Percobaan	Putaran motor DC (rpm)
1	14056
2	13755
3	14064
4	13856
5	14125
Jumlah	69856

Jadi rata rata putaran maksimal motor dc (B) adalah sebagai berikut:

$$rata-rata = \frac{jumlah\ hasil\ percobaan}{jumlah\ percobaan}$$

$$= \frac{69856}{5}$$

$$= 13971\ rpm$$

Jika kecepatan putaran maksimum motor dc adalah 13971 rpm, maka putaran motor dc masing-masing tombol dapat dicari dengan rumus:

$$D = \frac{rpm}{rpm\ (\max)} \text{ X } 100\%$$

rpm = D * rpm (max)

- Tombol "F❶" *Duty Cycle* 25%, rpm yang dihasilkan adalah:

  Rpm25% = 0,25 X 13971

  = 3492 rpm

- Tombol "F❷" *Duty Cycle* 50%, rpm yang dihasilkan adalah:

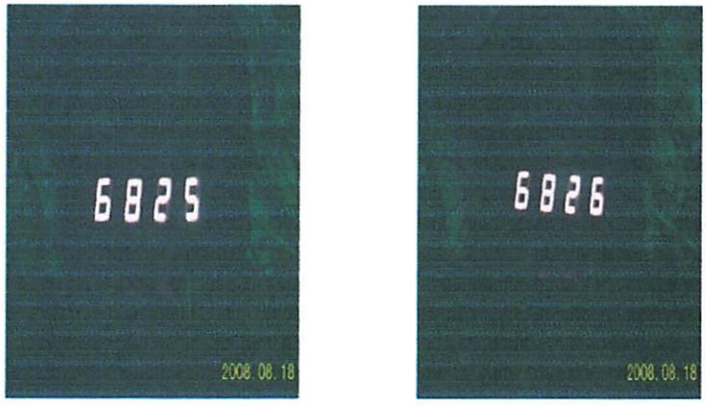
  rpm50% = 0,5 X 13971

  = 6985 rpm

- Tombol "F❸" *Duty Cycle* 75%, rpm yang dihasilkan adalah:

  Rpm75% = 0,75 X 13971

  = 10478 rpm



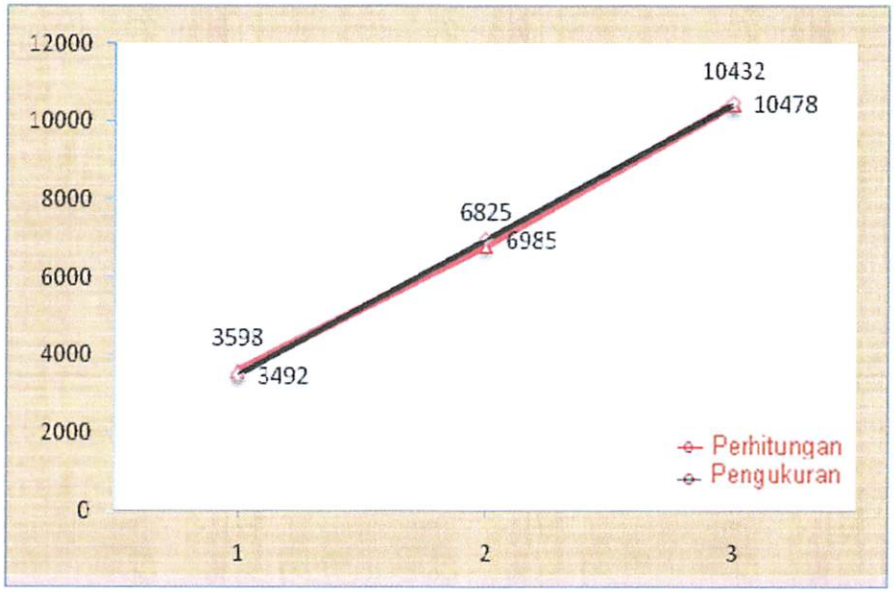
(a) (b)

Gambar 4-10 Pengujian Putaran Motor DC
(a) Pengujian Pertama (b) Pengujian Kedua

Hasil pengukuran putaran motor dc tidak sesuai dengan hasil perhitungan. Ini dikarenakan kondisi baterai sudah lemah yang mengakibatkan konsumsi daya motor dc kurang, Akurasi tachometer, dan kesalahan pengujian.

Tabel 4-6 Perbandingan Perhitungan Dan Pengukuran Putaran Motor DC

Tombol	Putaran Motor DC (rpm)		Error (%)
	Perhitungan	Pengukuran	
F1	3492	3598	3
F2	6985	6825	2,2
F3	10478	10432	0,4
Total	20955	21855	4,2



Grafik 4-2 Perbandingan Pengukuran Dan Perhitungan PWM Terhadap Putaran Motor DC

Dari percobaan dapat di ambil kesimpulan, semakin besar PWM yang diberikan maka semakin besar pula putaran motor dc yang dihasilkan.

Penyimpangan dari pengambilan data diperoleh dari besarnya selisih antara hasil penunjukan alat ukur yang diuji dengan hasil perhitungan. Jadi besarnya penyimpangan adalah :

$$\mathbf{E = Fu - Fs}$$

Dimana : E = Hasil Penyimpangan (Error)

Fu = Hasil Perhitungan.

Fs = Hasil Pengukuran.

Sedangkan penyimpangan (error) total terhadap perhitungan dalam bentuk persen diperoleh dari :

$$\sum \%Penyimpangan = \frac{\sum Fu - \sum Fs}{\sum Fu} \text{ X } 100\%$$

$$= \frac{20955 - 21855}{20955} \text{ X } 100\%$$

$$= 4{,}2\ \%$$

Sedangkan untuk persentase ketelitian dalam persen diperoleh perhitungan sebagai berikut :

$$\sum \%Ketelitian = 100\% - \sum \%Penyimpangan$$

$$= 100\% - 4{,}2\ \%$$

$$= 95{,}8\ \%$$

4.5. Pengujian review camera cctv

Gambar 4-11 Review camera cctv

BAB V

PENUTUP

5.1. Kesimpulan

Setelah dilakukan pengujian, pengamatan, dan penganalisaan alat berdasarkan *literature*, maka penulis mengambil beberapa kesimpulan:

- Secara keseluruhan sistem pengendalian motor dc pada helicopter model *remote control (R/C)* menggunakan mikrokontroller AT89s8252 telah bekerja dengan baik.
- Jarak maksimal pengiriman dan penerimaan dengan halangan *(indoor)* adalah ± 15m, untuk kamera adalah ± 20 m.
- Gerakan yang dapat dilakukan adalah naik ke atas, turun, maju, mundur, belok kanan dan kiri.
- Semakin besar PWM yang diberikan maka semakin besar pula putaran motor dc yang dihasilkan.
- Hasil pengukuran putaran motor dc tidak sesuai dengan hasil perhitungan. Ini dikarenakan kondisi baterai sudah lemah yang mengakibatkan konsumsi daya motor dc kurang, Akurasi tachometer, dan kesalahan pengujian.
- Penyimpangan (error) total terhadap perhitungan dalam bentuk persen diperoleh dari :

$$\sum \%Penyimpangan = \frac{\sum Fu - \sum Fs}{\sum Fu} \text{ X } 100\%$$

$$= \frac{20955 - 21855}{20955} \text{ X } 100\%$$

$$= 4{,}2\ \%$$

Sedangkan untuk persentase ketelitian dalam persen diperoleh perhitungan sebagai berikut :

$$\sum \%Ketelitian = 100\% - \sum \%Penyimpangan$$

$$= 100\% - 4{,}2\ \%$$

$$= 95{,}8\ \%$$

5.2. Saran

Beberapa tambahan yang diperlukan dalam meningkatkan kemampuan alat ini adalah:

- Untuk menambah jarak jangkauan dapat diganti dengan pemancar dan antena yang lebih baik.
- Untuk mendapatkan gambar yang lebih baik dapat di ganti dengan kamera yang lebih baik kualitasnya.

DAFTAR PUSTAKA

Budiharto, Widodo. 2006. *Belajar Sendiri Membuat Robot Cerdas*. Jakarta: Elex Media Koputindo

Budiharto, Widodo. 2005. *Interfacing Komputer dan Mikrokontroler*. Jakarta: Elex Media Komputindo

Data Sheet, *TLP434A & RLP434A*. Lapiac Teknologi.Inc

www.electroniclab.com, rubrik elka dasar, aswan hamonangan

www.futurlec.com

www.CoolCircuit.com

www.geocities.com

www.holteksemiconductor.com.tw

www.lapiac.com

www.digiware.com

www.datasheet.com

www.delta-electronic.com

www.robotik.com

LAMPIRAN

PERSEMBAHAN
DARI HATI UNTUK HATI

Syukur yang tak terhingga ku ucap kepada pemilik hidup hamba, ALLAH SWT, dan junjungan hamba, Nabi besar MUHAMMAD SAW..Di dalam lindunganSijin Nya hamba mampu menyelesaikan satu babak dalam perjalanan hidup hamba, dan dalam genggamanNya hamba berserah dalam ikhtiar untuk masa depan hamba.

Tunduk hormat kepada Ayah Dedy Soeharno & Bunda Tri Punjungwati untuk kesabaran, cinta, kasih sayang & kepercayaan yang diberikan kepada hamba."Semoga anakmu ini mampu selalu membanggakan ayah&bunda"

De' Yudha,terima kasih untuk cinta,kasih sayang dan game2 yang ente install di computer, bisa ngilangin stress...(kuliahmu cepet selesain yo le...)

Gemblong, makasih untuk keceriaannya dan sms kangennya slalu bikin semangat pengen cepat2 selesain skripsi biar bias cepet pulang.

"Semoga mase bisa jadi ka2k yang baik buat kalian"

Ovan & Hepi makasih untuk keceriaannya.

Makasih untuk seluruh keluarga yang selalu ada dibelakangku,mensupport tanpa lelah.

My deepest love, My wife (Mama Aminila Sari), My Lil' Sons (Kakak Moshi-moshi, kakak Muhammad Faisal Raisya Waldani & Ade' Zahratus Shita), kalian adalah lentera jiwaku cahaya hidupku, kalian adalah keluarga kecil papa, cinta dalam hidup papa.

For all my friends,agus danny,ongky,pipie,ponchos,mupid,bigshow,dan semuanya pokoe wes arek elka 4 (suwun atas pertemanan & supportnya...lets make some noise guys), anak2 D5,B6 dan seluruh temen di villa sengkaling,makasih banyak bro untuk semuanya... ,anak2 SENGKALING GYM (makasih bikin ane sehat..hehehe...),tedjo, grandong, pak budi ndut, andro (enak yo mumet skripsi?hehe..), Kenyob, Poncol, Jack, Dona, Sari..(Ayo SEMANGAAAAT...), Mas Ruri suwun yo mas tak repoti terus...

Para dosen2 yang ku banggakan, terimakasih so much untuk ilmu yang ada di otakku.

Dan semua temen2 yang ga bisa terucap, ane terima kasih banyak untuk pertemanannya.

With Love...

ANGGA WIDYA OKTAVIAN

INSTITUT TEKNOLOGI NASIONAL
Jl. Raya Karanglo Km 2
MALANG

FORMULIR BIMBINGAN SKRIPSI

Nama : ANGGA WIDYA OKTAVIAN
NIM : 02.17.140
Masa Bimbingan : 15 Desember 2007 s/d 15 Juni 2008
Judul Skripsi : Perancangan dan Pembuatan Pengirim Gambar Dari Helikopter Model Yang Dikontrol Oleh Personal Komputer (PC)

NO	TANGGAL	URAIAN	PARAF PEMBIMBING
1	05/07	proposal Skripsi	
2	25/08	Konsultasi Bab I s/d III Revisi	
3	29/08	Acc Bab I s/d III di Lanjutkan ke Bab berikutnya!	
4	1/09 08	Konsultasi Bab IV Hasil Kesimpulan dan Saran	
5		Hasil perbaikan Bab IV	
6	3/09	Konsultasi Bab V & V Lanjutkan ke makalah seminar hasil!	
7	23/09	Acc ujian SEMHASIL	
8			
9			
10			

Malang,
Dosen Pembimbing

(I Komang Somawirata, ST, MT)
NIP.P. 1030100361

Form S-4

INSTITUT TEKNOLOGI NASIONAL
Jl. Raya Karanglo Km 2
MALANG

FORMULIR BIMBINGAN SKRIPSI

Nama : ANGGA WIDYA OKTAVIAN
NIM : 02.17.140
Masa Bimbingan : 15 Desember 2007 s/d 15 Juni 2008
Judul Skripsi : Perancangan dan Pembuatan Pengirim Gambar Dari Helikopter Model Yang Dikontrol Oleh Personal Komputer (PC)

No	Tanggal	Uraian	Paraf Pembimbing
1	15-12-2007	Proposal Skripsi (acc)	
2	17-07-2008	Bab I dan bab II (revisi)	
3	21-07-2008	Bab II (acc) , Bab III (revisi)	
4	4-08-2008	Bab III (acc) , Bab IV (revisi)	
5	21-08-2008	Bab IV (acc) , Bab V (acc)	
6	13-09-2008	Makalah Seminar Hasil (revisi)	
7	15-09-2008	Makalah Seminar Hasil (acc)	
8	20-09-2008	Ujian skripsi	
9			
10			

Malang,
Dosen Pembimbing

(Ir. Eko Nurcahyo)
NIP.Y. 1028700172

Form S-4

INSTITUT TEKNOLOGI NASIONAL
Jl. Bendungan Sigura-gura No. 2
MALANG

PERMOHONAN PERSETUJUAN SKRIPSI

Yang betanda tangan dibawah ini :

Nama	: ANEEA . WIDYA . OKTAVIAN
NIM	: 02.17.140
Semester	: 10
Fakultas	: Teknologi Industri
Jurusan	: Teknik Elektro S-1
Konsentrasi	: Teknik Elektronika / Teknik Energi Listrik
Alamat	: VILLA SENEKALING D-5

Dengan ini kami mengajukan permohonan untuk mendapatkan persetujuan untuk membuat ***SKRIPSI Tingkat Sarjana***. Untuk melengkapi permohonan tersebut, bersama kami lampirkan persyaratan-persyaratan yang harus dipenuhi.
Adapun persyaratan-persyaratan pengambilan ***SKRIPSI*** adalah sebagai berikut :

1. Telah melaksanakan semua praktikum sesuai dengan konsentrasinya (.....)
2. Telah lulus dan menyerahkan Laporan Praktek Kerja (.....)
3. Telah lulus seluruh mata kuliah keahlian (MKB) sesuai konsentrasinya (.....)
4. Telah menempuh mata kuliah ≥ 134 sks dengan IPK ≥ 2 dan tidak ada nilai E (.....)
5. Telah mengikuti secara aktif kegiatan seminar skripsi yang diadakan Jurusan (.....)
6. Memenuhi persyaratan administrasi (.....)

Demikian permohonan ini untuk mendapatkan penyelesaian lebih lanjut dan atas perhatiannya kami ucapkan terima kasih.

Telah diteliti kebenaran data tersebut diatas
Recording Teknik Elektro

(Ir. Handayani)

Malang, 25 MEI 2007
Pemohon

(ANEEA . WIDYA . OKTAVIAN)

Disetujui
Ketua Jurusan Teknik Elektro

Ir. F. Yudi Limpraptono, MT
NIP. P. 1039500274

Mengetahui
Dosen Wali

(Ir. Yusuf Ismail Nakhoda, MT)

Catatan :
Bagi mahasiswa yang telah memenuhi persyaratan mengambil SKRIPSI agar membuat proposal dan mendapat persetujuan dari Ketua Jurusan/Sekretaris Jurusan T. Elektro S-1

1 IPK = 356.81 / 135 = 2.64
2 Mata kuliah baru : SISTEM INSTRUMENTASI ELEKTRONIKA
3 Praktikum : D.S.K ; SISKOM ; KENDALI INDUSTRI

Form. S-1a

Institut Teknologi Nasional Malang
Fakultas Teknologi Industri
Jurusan Teknik Elektro S-1

BERITA ACARA SEMINAR PROPOSAL SKRIPSI
JURUSAN TEKNIK ELEKTRO S-1

Konsentrasi : ~~Teknik Energi Listrik~~/Teknik Elektronika*)

1.	Nama Mahasiswa: ANGGA WIDYA CETAVIAN			Nim: 02 17 140
2.	Keterangan Pelaksanaan	Tanggal	Waktu	Tempat Ruang:
3.	Spesifikasi Judul (berilah tanda silang)**) a. Sistem Tenaga Elektrik b. Energi & Konversi Energi c. Tegangan Tinggi & Pengukuran d. Sistem Kendali Industri (e) Elektronika & Komponen f. Elektronika Digital & Komputer g. Elektronika Komunikasi h. lainnya			
4.	Judul Proposal yang diseminarkan Mahasiswa	PERANCANGAN DAN PEMBUATAN PENGIRIM GAMBAR DARI HELICOPTER MODEL YANG DIKONTROL OLEH PERSONAL COMPUTER (PC)		
5.	Perubahan Judul yang diusulkan oleh Kelompok Dosen Keahlian	.....		
6.	Catatan: 1). Kurang menguasai konsep pengendalian dr helikopter			
7.	Catatan: Persetujuan Judul Skripsi Disetujui, Dosen Keahlian I	Disetujui, Dosen Keahlian II		
	Mengetahui, Ketua Jurusan. Ir. F. Yudi Limpraptono,MT NIP. P. 1039500274	Disetujui, Calon Dosen Pembimbing ybs		

Perhatian:
1. Keterangan: *) Coret yang tidak perlu
   **) dilingkari a, b, c, ...atau g sesuai bidang keahlian

Form S-3c

LABORATORIUM KONVERSI ENERGI ELEKTRIK
INSTITUT TEKNOLOGI NASIONAL MALANG
KAMPUS II : Jl. Raya Karanglo KM 2 Tlp. (0341)417636 EXT 604

SURAT PERMOHONAN

Dengan Hormat,

Yang bertandatangan sebagai pemohon, saya mahasiswa dengan Identitas sebagai berikut :

Nama	: Angga Widya Oktavian
Nim	: 02.17.140
Jurusan	: Teknik Elektronika S-1
Fakultas	: Fakultas Teknologi Industri
Keperluan	: Pengukuran Rpm Putaran motor,Untuk Pengujian Alat Tugas Akhir
Waktu	: Tgl 16 - 20 Agt 2008

Mengajukan permohonan peminjaman dan penggunaan alat untuk keperluan sebagaimana yang tercantum diatas.

Daftar Peralatan Yang Dipinjam

No	Kode Alat	Nama Alat	Trade Mark
1	DL 2239A	DIGITAL STROBOSKOPE TACHOMETER	DE LORENZO INSTRUMENTS
2			
3			
4			
5			
6			
7			
8			
9			
10			

Catatan : 1. Sanggup mentaati tata tertib Laboratorium Konversi Energi Elektrik
2. Apabila terjadi kerusakan pada alat yang digunakan saya sanggup mengganti.

Demikian surat permohonan ini saya buat dengan sebenarnya dan dapat dipertanggung jawabkan, Terima Kasih.

Asisten Pendamping

Andik Sandia
03.12.078

Kaur Lab. Konversi Energi Elektrik

Ir. M. Abdul Hamid, MT
Nip . 101 880 00188

Pemohon

Angga Widya O
02.17.140

```
//=============================================================================
/*  Helicopter + Camera
    ITM - MALANG 2008
    Oka
    0217140

    Hardware :
    - AT89S8252
    - L293
    - Komunikasi Data via TLP RLP 434                                        */
//=============================================================================
// Port-map of the MCU :                                                    =
// P0_0 -> EnaMotUt                                                         =
// P0_1 -> MotUt                                                            =
// P0_2 -> unused                                                           =
// P0_3 -> unused                                                           =
// P0_4 -> EnaMotEk                                                         =
// P0_5 -> MotEk                                                            =
// P0_6 -> unused                                                           =
// P0_7 -> unused                                                           =
// P2_0 -> Signal Servo 0                                                   =
// P2_1 -> Signal Servo 1                                                   =
// P3_0 -> [RXD]                                                            =
//=============================================================================
```

```
//#include <at89s8252.h>   // Definitions of registers, SFRs and Bits
#include <8052.h>   // Definitions of registers, SFRs and Bits
#include <string.h>

#define Servo1   P2_0   // Motor Servo 1
#define Servo2   P2_1   // Motor Servo 2
#define EnaMotUt  P0_0   // Enable Motor Utama
#define MotUt    P0_1   // Motor Utama 2
#define EnaMotEk  P0_4   // Enable Motor Ekor
#define MotEk    P0_5   // Motor Ekor

void Serial_ISR(void) interrupt 4 using 1;
void init_MCU(void);

unsigned char last,arser[10],Add,itrMotUt,itrMotEk;
bit fSerial;
unsigned int hitServo1,hitServo2,hitMotUt,hitMotEk,hitdelay;

void Serial_ISR(void) interrupt 4 using 1
{
 if(RI){ last = SBUF;
       if (Add==0){if(last & 0x80){arser[Add]=last;Add++;}}
       else {if (Add>3){Add=0;fSerial=1;}
         else {arser[Add]=last;Add++;}};
//       if (Add>0){{if (Add>3){ES = 0;Add=0;fSerial=1;}
```

```
//                          else {arser[Add]=last;Add++;}}}
//          else {if (last & 0x80) {arser[0]=last;Add=1;}};
            RI =0;}
  return;
}

void init_MCU(void)
{
EA = ET0 = ET1 = ES = TR0 = TR1 = 0;
P1 = P3 = 0xFF;
TL0  = 0x00;
TH0  = 0x70;  // 20 ms Tick / Reload PWM
TL1  = TH1 = 0xF4; // 2400 bps @ 11.0592 MHz
//TL1  = TH1 = 0xE8; // 1200 bps @ 11.0594 MHz
TMOD = 0x21;
SM1  = 1;
SM0  = 0;
REN  = 1;
RI = TI = 0;
TR0 = 0;
```

AT89S8252 In-System Programming

oduction

application note illustrates the in-em programmability of the Atmel 9SXXXX (S-series) microcontrol-. A method is shown by which an 9S8252 in an application may be rammed remotely over a standard hone line.

software for this application note be obtained by downloading from: el BBS (408) 436-4309 or site: http://www.atmel.com

Example Application

application shown in Figure 1 is a le implementation of a moving dis-. This application was selected for its licity and ability to show graphically esults of in-system programming. text to be displayed is programmed the AT89S8252 microcontroller as of its firmware, and can be changed programming the device.

displayed text is presented in one of modes, selected by a switch. In the mode, one character at a time rs the display from the right and es quickly to the left through each ent of the display to its final position e assembled message. In the sec-mode, the message moves through isplay, from right to left, with the dis-acting as a window onto the mes-e. This mode is familiar as the od often used in displays of stock s.

text is displayed on four DL1414T, element, 17-segment alphanumeric ays with integral decoders and driv-. This yields 16 total display ele-s, each capable of displaying digits the upper case alphabet, and punc-on characters. The displayable char-codes are ASCII 20-5F (hexadeci-

A power-on reset circuit and a 6-MHz crystal complete the application. Neither external program memory nor external data memory is used.

Modifications to the Application to Support In-System Programming

The AT89S8252 microcontroller features an SPI port, through which on-chip Flash memory and EEPROM may be programmed. To program the microcontroller, RST is held high while commands, addresses and data are applied to the SPI port. For command format and timing requirements, refer to the Atmel AT89S8252 Microcontroller data sheet.

Figure 2 shows the example application modified for in-system programming. The microcontroller reset circuit has been eliminated and RST is controlled by the programmer. The absence of a reset circuit requires that the programmer reset the microcontroller when power is first applied to the application. An optional connection ($\overline{\text{SHUTDN}}$) to an AT89S8252 interrupt input has been provided to allow the programmer to signal the microcontroller prior to programming. The resident firmware responds to the interrupt by displaying a message ("PROGRAMMING") indicating that programming is in progress.

A simple latch, composed of four OR gates, has been added between the outputs of the microcontroller and the display control inputs. The latch holds the display control signals inactive when RST is asserted, eliminating erratic operation of the displays during programming. No isolation of the display address or data inputs is required, since these inputs are ignored by the displays when the control signals are inactive. After programming, when RST is deasserted, the microcontroller I/O ports are high as

Microcontroller

Application Note

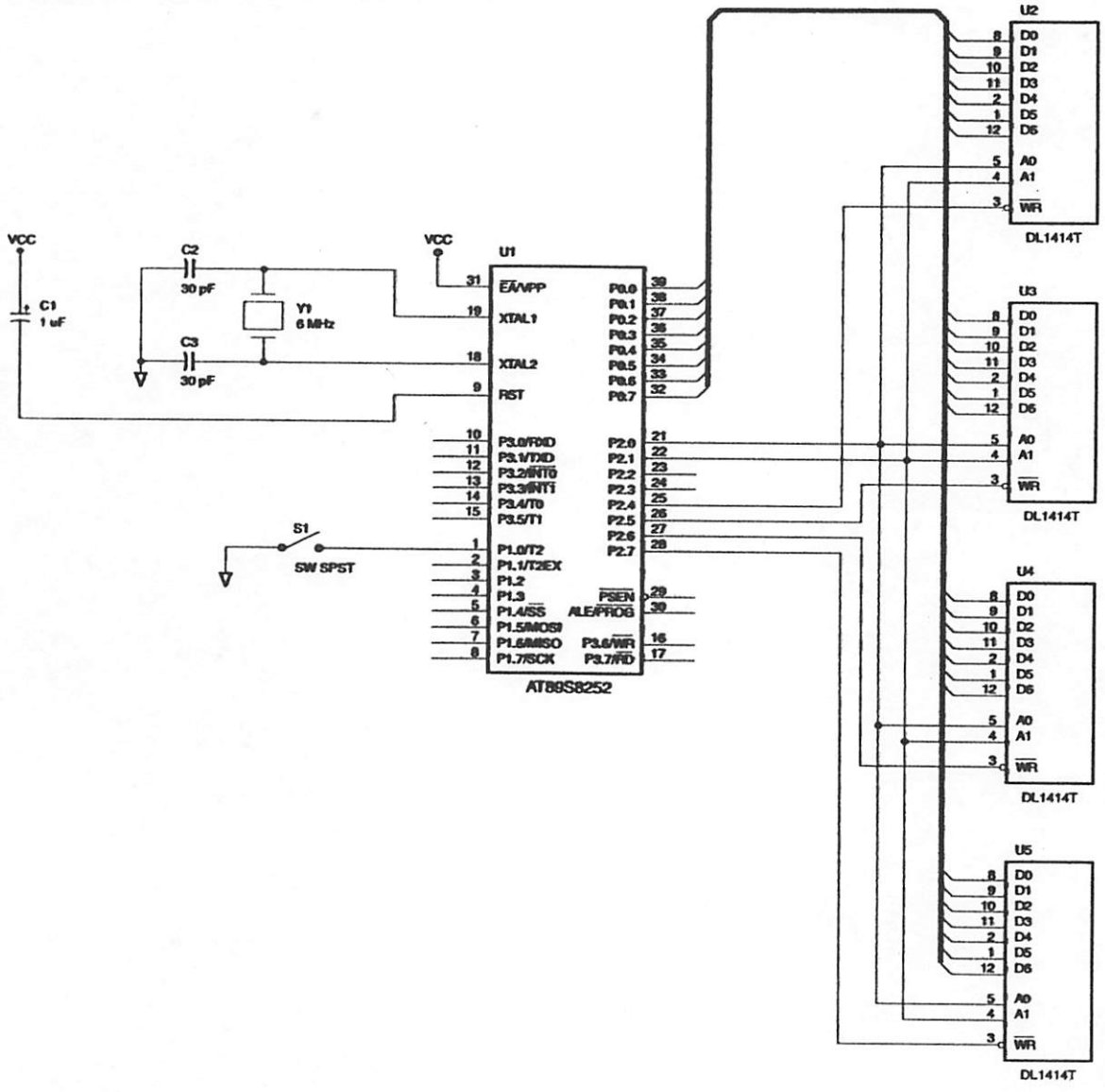
0898A-A–12/97

e latch becomes transparent. Since the display control puts are inactive high, the display contents are not dis-rbed until the new firmware writes the displays. Although ot essential in this application, it might be imperative in me applications that the state of the peripheral circuitry t be disturbed during programming.

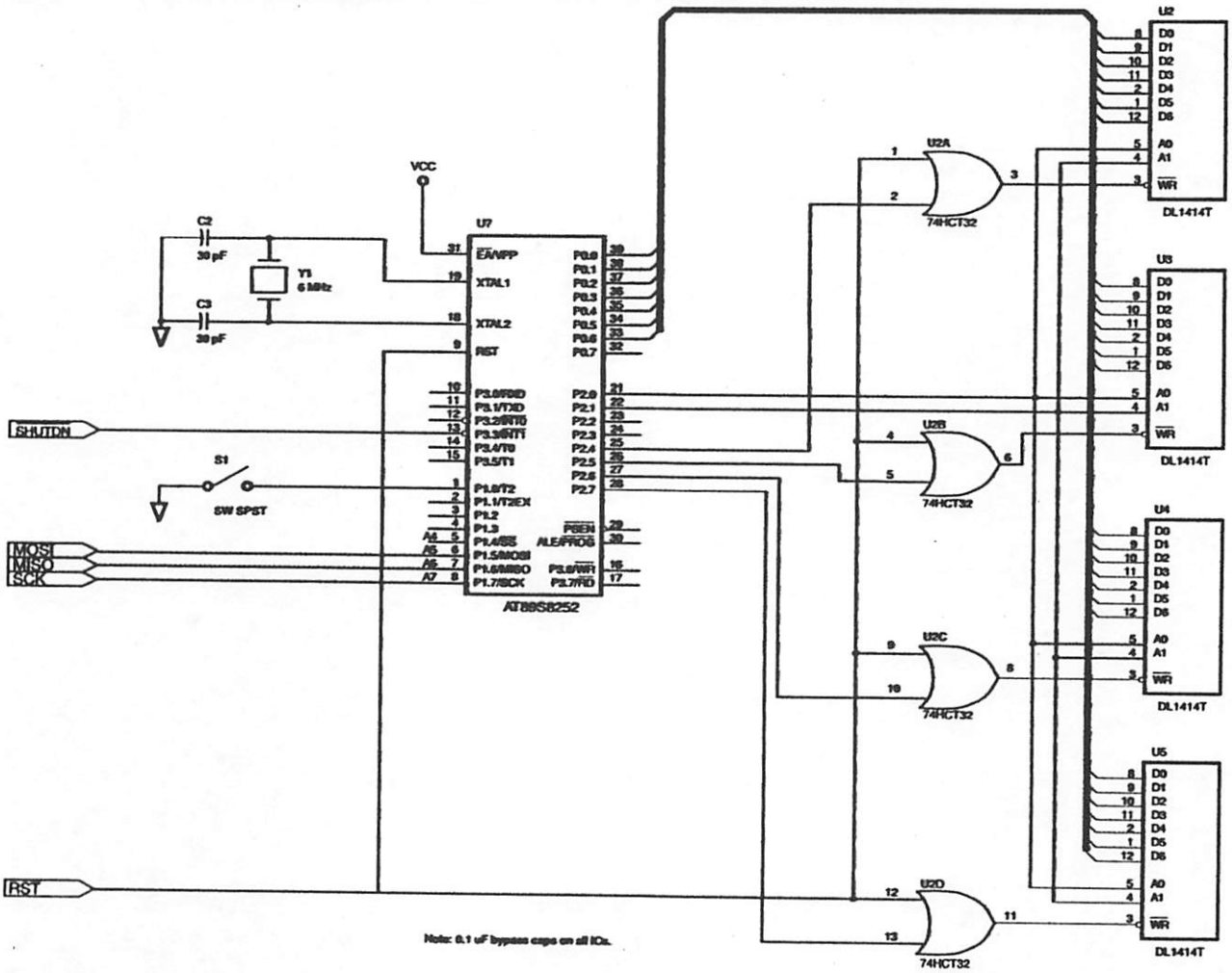
nally, programmer access has been provided to three T89S8252 SPI port pins: P1.5/MOSI, P1.6/MISO and 1.7/SCK. SPI port pin P1.4/$\overline{SS}$ is not used during pro-amming. In the example application, the SPI port pins are available for use in programming the microcontroller. Applications which utilize the SPI port pins must be modified by the addition of circuitry which will isolate the SPI port when RST is asserted, freeing the pins for use in programming the microcontroller. Circuitry which is added to support programming must appear transparent to the application during normal operation.

The code for the modified display application is shown in Appendix 2.

gure 1. AT89S8252 Moving Display Application Example



gure 2. AT89S8252 Moving Display Application Modified for In-System Programming



e Programmer

e programmer shown in Figure 3 interfaces with a dem, from which it receives packetized data. After dis-ting the data packets, the programmer generates the nals required to program the data into the AT89S8252 rocontroller in the modified application. Code for the grammer is shown in Appendix 3.

e programmer circuitry consists of little more than an nel 20-pin AT89C2051 microcontroller and a Maxim X232 line driver/receiver. The microcontroller runs at 0592 MHz, which allows the serial port to operate at a nber of standard baud rates. The line driver/receiver duces RS-232 levels at the modem interface while uiring only a 5-V power supply. The AT89C2051 micro-troller does not support external program or data mem-, which requires that program code be kept small ugh to fit into on-chip memory.

The serial interface, through which the programmer connects to the modem, supports two handshaking signals, DTR and DSR. On power up, the programmer asserts DTR, to which the modem responds by asserting DSR. If the modem should fail to respond to any command, including the command to hang up, the programmer deasserts DTR, which forces the modem to hang up.

The programmer controls the modem by sending ASCII command strings over the serial interface, to which the modem responds with Hayes-style ASCII numeric codes. The programmer code is optimized for use with the U.S. Robotics Sportster 14,400 baud external modem used in the test configuration and may require modifications if used with other modems.

Since a reset circuit is absent from the modified application, the programmer provides the power-on reset function to the AT89S8252 microcontroller. The programmer powers up with RST asserted, resetting the microcontroller. Some

ng mode, preventing it from executing the incomplete or valid firmware which it contains.

is important to note that invalid packets are NEVER programmed into the application microcontroller. To do so ight over-write valid program data which could not be covered.

oon receipt of an end-of-file packet, the programmer turns its control outputs to the inactive, power-on state, owing the application microcontroller to begin execution its new firmware. The programmer then resumes polling r a valid packet header, subject to the 30-second reset lay. If a valid packet is received prior to the expiration of e 30-second delay, another programming cycle begins, hich can only be terminated by the reception of a valid d-of-file packet.

the reset delay expires prior to the reception of a valid d-of-file packet, the modem will hang up and the programmer will return to sleep, waiting for the next call. In this se, the application microcontroller is left in programming ode, preventing it from executing its firmware. To return e application to normal operation, another call must be ceived, and a valid program file downloaded, terminated an end-of-file packet.

etting Up the Hardware

cal Station

stall the selected modem into the IBM PC AT-compatible mputer and connect it to a standard analog telephone e. The modem must support a data rate of at least 9600 ud.

mote Station

nnect the programmer and modified display application the U.S. Robotics Sportster 14,400 baud external odem. Connect the modem to a standard analog telene line and set the modem switches as indicated below.

dem switch settings:

1	UP	DTR normal
2	DOWN	Numeric result codes
3	DOWN	Display result codes
4	DOWN	Suppress command echo
5	UP	Auto answer
6	UP	CD normal
7	UP	Load NVRAM defaults
8	DOWN	Smart mode

rn the modem on and apply power to the programmer d display application. The microcontroller in the applicaı will begin executing its firmware, if it contains any. The ogrammer will initialize the modem, as indicated by the ivity on the modem status indicators. If it should become necessary to reinitialize the modem, briefly interrupt power to the programmer.

Installing and Configuring Procomm Plus for Windows, Version 3.0

Install Procomm Plus as instructed in the User Manual. When prompted to specify the modem in use, select the installed modem from the list.

Put the provided ASPECT script (ATX.WAX) into the Procomm Plus ASPECT directory. If the default directories were utilized during installation, the correct directory is: \PROWIN3\ASPECT.ATX.WAX is the executable ASPECT script which results from compiling the source file ATX.WAS, shown in Appendix 4. Source files may be edited from within Procomm Plus using the ASPECT Editor, available in the Tools menu. The ASPECT Editor provides the option to compile a source file in the Editor Tools menu.

Launch Procomm Plus and create a Connection Directory entry for the remote station. Under Port Settings, set the baud rate to 9600, parity to EVEN, number of data bits to 7, number of stop bits to 1, plex to FULL.

Creating a Hex File

The example source code for the modified display application (Appendix 2) contains a string at location "usr_msg" which is written repeatedly to the alphanumeric displays. The user may substitute a different message, as long as it is enclosed in single quotes and is null-terminated. Long messages may require that the value in the subsequent ORG directive be increased to prevent the message from being over-written by code. The message may contain only characters with ASCII codes from 20-5F (hexadecimal). The modified source code may then be assembled, linked and an Intel hex file produced.

During the development of this application note, code was assembled and hex files generated utilizing the tools in a vintage copy of the Intel MCS-51 Software Development Package for the IBM PC. The source code may require cosmetic changes for compatibility with other assemblers and software tools. It is especially important to note that variations exist in Intel hex file format. This application requires that record data fields be limited to 16 or fewer entries and that address fields contain 4 hex digits. The user must verify that the hex files produced by the selected tools conform to the format documented in Appendix 1.

Uploading a Hex File

Launch Procomm Plus and select the correct entry from the list box in the toolbar to dial the remote site. If the line is busy and remains busy for more than 30 seconds, the programmer must be reset.

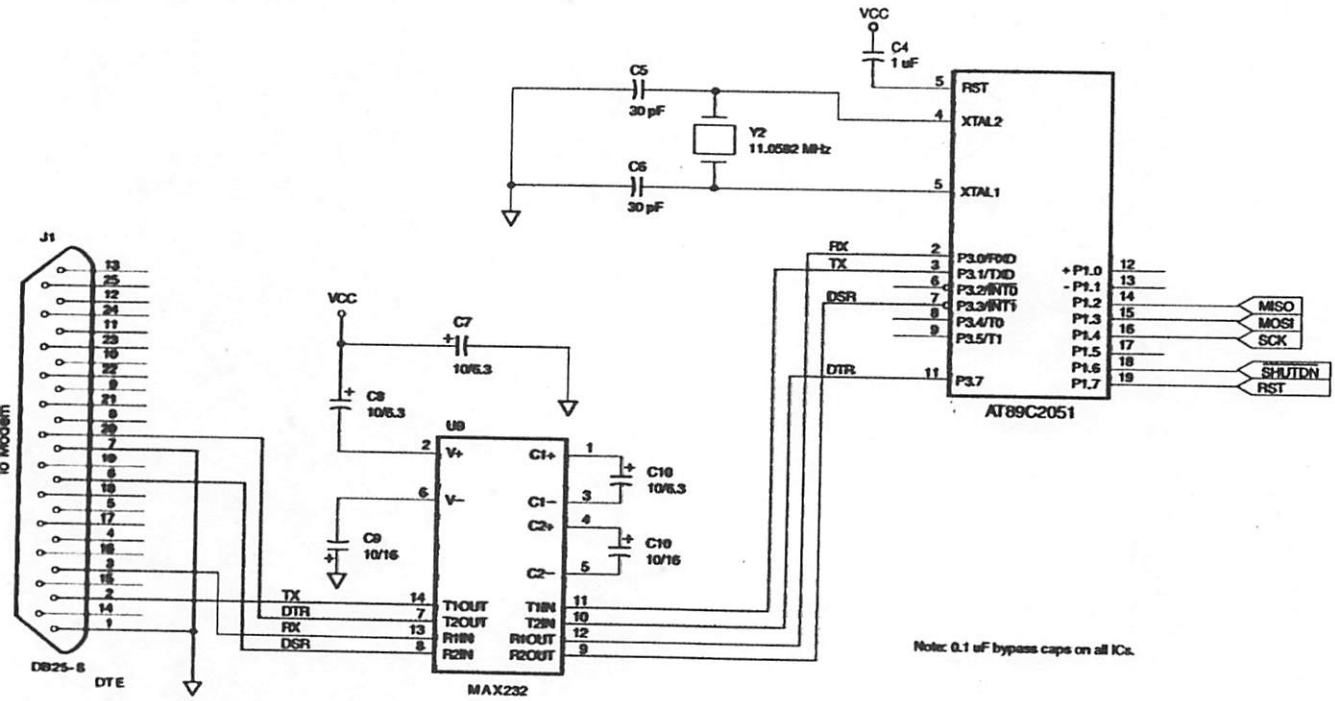
After a connection with the remote site has been established, run the ATX ASPECT script by selecting it from the list box in the toolbar. When prompted by the script, enter

ne path and file name (including extension) of the hex file
o upload to the programmer at the remote site. The pro-
rammer must receive the first record from the file within 30
econds of the time the connection was established or it
ill hang up and the user will be required to redial.

uring the data transfer, data and status information is dis-
layed in the Procomm Plus Terminal Window. If the trans-
r completes successfully, the message "End of File" will
ppear in the Terminal Window. The user has 30 seconds
om the appearance of messages "End of File" or
EXCESSIVE RETRIES: UPLOAD ABORTED" to rerun the
cript and upload another file, if desired, before the pro-
rammer hangs up. If the message "UPLOAD ABORTED
Y REMOTE" appears, the programmer has hung up and
e user must redial before uploading another file.

me later, RST is deasserted under firmware control, allowng the application microcontroller to run normally. When programming is required, the programmer again asserts RST.

igure 3. AT89S8252 Programmer



uring programming, the programmer outputs serial data the MOSI pin, synchronized to a software-generated ock output on the SCK pin. Serial data is input on the SO pin, also synchronized to SCK. The maximum freency of SCK must be less than 1/40th the crystal freency of the AT89S8252 microcontroller being proammed, as specified in the AT89S8252 data sheet. The cumented code produces a maximum SCK frequency of proximately 90 KHz, permitting a minimum AT89S8252 rstal frequency of approximately 3.6 MHz.

emote Programming Over a Standard lephone Line

e programmer and modified application described previsly are connected to a phone line through a modem at a note site. Using a personal computer with a modem, a er can upload code containing a new message, which is grammed into the AT89S8252 microcontroller in the olication. When programming is complete, the microconler executes the new firmware, which displays the new ssage.

cal Station

e local station in the test configuration consists of an IBM AT-compatible personal computer with a Cardinal PV34ILC 33,600 baud internal modem. Any modem y be used, as long as it is compatible with the data communications software and matches the data rate and error correction protocols of the modem at the remote site.

Procomm Plus for Windows, version 3.0, a commercial data communications package, is used to configure the modem, set up communications parameters, and establish a link with the remote modem. Procomm Plus includes a macro language called ASPECT, which allows the user to write and compile scripts which implement custom file transfer protocols. A simple ASPECT script was written to read the contents of a code file and upload it to the remote programmer. The ASPECT script is shown in Appendix 4.

The file transfer protocol (FTP) implemented is a simple send-and-wait, packet-oriented protocol. The FTP transmit and receive modes are diagrammed in the flowcharts in figures 4 and 5, respectively. The computer sends each packet without flow control and waits for a response. The programmer may acknowledge the packet by sending an ACK or may negatively acknowledge the packet by sending a NAK. Upon receipt of an ACK, the computer sends the next packet. If the clone receives a NAK, it resends the same packet. Transmission proceeds in this manner until the entire file has been transferred.

The programmer might respond to a packet by sending a CAN, which indicates that a non-recoverable error has occurred and that the computer should immediately abort the file transfer. If the programmer fails to respond to a

acket within a limited period of time, the computer will send the same packet. The computer will continue to send the same packet until a valid response is received until the allowed number of attempts is exceeded, at hich time the file transfer is aborted.

he send-and-wait nature of the FTP allows the time equired for the programmer to program the packet data to the application microcontroller to be easily absorbed. rogramming verification requires no explicit command or sult codes, or additional data transfers. The programer's response to a packet reflects the result of the proamming verification operation performed by the programer: ACK indicates success, CAN indicates failure.

exadecimal object file format (Intel hex) was chosen as e format of the files to be uploaded to the programmer. ne records in a hex file serve, unchanged, as the packets the FTP described above; no service fields need to be lded. The fields in Intel hex file records are shown in ppendix 1. The colon which begins each record serves as e packet signature field. The load address field serves as e packet sequence number. A checksum is provided as e last field in each record. Since 7-bit ASCII coding is uied, the eighth bit of each byte is available to be used for rity checking.

nce the AT89C2051 microcontroller in the programmer es not utilize external data memory, necessary packet ffering must be done using internal RAM. Limited memy precludes the use of conventional FTPs which utilize ckets of 128 bytes or more. The hex packet format used this application limits packet data fields to 16 or fewer tries, requiring little memory for buffering.

disadvantage of the hex packet format is the use of SCII, which requires each program data byte to be pressed as two hex characters. This demands that arly twice as many bytes be transferred as might otherse be required. This is not a severe limitation, however, ce typical file transfer times are on the order of a few conds.

mote Station

e remote station in the test configuration consists of the ogrammer and modified application, previously scribed, connected to a U.S. Robotics Sportster 14,400ud external modem.

er power is applied, the programmer resets the 89S8252 microcontroller in the application, and then s its control outputs inactive, allowing the application to normally. The programmer configures the modem to swer incoming calls and puts itself to sleep. While the grammer sleeps, the modem monitors the phone line, iting for an incoming call. When a call is detected, the dem answers and attempts to establish communication h the caller. If a connection is established, the modem nds a connect code to the programmer, waking it up. The programmer verifies the connect code and begins polling for a valid packet header. Invalid connect codes are ignored.

Incoming packets must arrive fewer than 30 seconds apart, or the modem hangs up and the programmer returns to sleep, waiting for the next call. If the caller hangs up, the 30-second period must expire before another call will be answered. Calls incoming during the reset delay period are ignored.

If a valid packet header is received prior to the expiration of the reset delay period, the programmer will attempt to read and validate the incoming packet. At any time during packet reception, an invalid character, parity error or timeout during character reception will cause the partial packet to be declared invalid and discarded.

Two packet types are defined: data and end-of-file. A data packet contains five fields in addition to the packet header, one of which is a variable length data field. The data field contains program data to be written into the application microcontroller. The load address field contains the address at which the data is to be written. The end-of-file packet contains the same fields as the data packet, except that the data field is empty. This packet type has special meaning to the programmer, as explained below.

Any packet which contains an invalid record type, record length or checksum is invalid. Program data accumulated during the processing of an invalid packet is discarded. The programmer sends a NAK to the computer to signal reception of an invalid packet and resumes polling for a valid packet header.

Receipt of the first valid data packet causes the programmer to interrupt the application microcontroller. The microcontroller responds to the interrupt by abandoning its usual routine and displaying a message ("PROGRAMMING") indicating that programming is taking place. If this is the first valid data packet since power was applied or an end-of-file packet was received, the programmer asserts the control signals necessary to place the microcontroller into programming mode.

The first and subsequent valid data packets are dissected as they are received and the data which they contain is programmed into the application microcontroller at the address indicated in the packet load address field. After programming, the data is read back from the microcontroller and verified against the received packet data. If programming was successful, the programmer sends ACK to the computer. The programmer then resumes polling for a valid packet header, subject to the thirty second reset delay.

If programming fails, the programmer sends CAN to signal the computer to abort the file transfer. The modem hangs up and the programmer returns to sleep, waiting for the next call. The application microcontroller is left in program-

Appendix 1: Intel Hex File Definition

Each record in hexadecimal object file format (Intel hex) contains the following fields:

<:> <rec length> <load address> <rec type> <data> <checksum>

The colon is the record header.

The record length field consists of two hex digits, and represents the number of entries in the data field.

The load address field consists of four hex digits, and indicates the absolute address at which the data in the data field is to be loaded.

The record type field consists of two hex digits, which are always zero in data records.

The data field contains from one to 16 pairs of hex digits.

The last two hex digits are a checksum on the record length, load address, record type, and data fields. The sum of the binary equivalents of these fields and the checksum itself is zero.

Each record in the file is terminated by a carriage return (0D hex) and line feed (0A hex).

A type one record marks the end of the file. The record always contains ":00000001FF".

Appendix 2: Code for Modified Display Application

```
NAME LEDShow1
; Displays predefined text strings on the LED display in one of two modes.
; The display mode can be changed at run time with the switch.

; The program may be interrupted by External Interrupt 1. This will cause the
; processor to display a string and enter a wait loop with interrupts disabled.
; Only reset will restore normal operation. This facility is provided so that
; the programmer can trigger an orderly shutdown before reprogramming the part.

; The LED display consists of four devices of four elements each,
; for a total display capacity of 16 characters.
; The display devices are numbered 0 to 3, from the right.
; The display elements are numbered from 0 to 3, from the right.
; Character positions are numbered 1 to 16, from the right.

NDEVS   EQU  4          ; number of devices
NELMS   EQU  4          ; number of elements in each device
SPACE   EQU  20h        ; blank
        DSEG AT 60h     ; stack origin
stack:  DS 20h          ; stack depth
SWITCH  BIT p1.0        ; display mode select input
        CSEG
        ORG 0000h       ; power on/reset vector
        jmp init
        ORG 0003h       ; external interrupt 0 vector
        reti            ; undefined
        ORG 000bh       ; timer 0 overflow vector
        reti            ; undefined
        ORG 0013h       ; external interrupt 1 vector
```

```
  jmp shutdown
  ORG 001bh              ; timer 1 overflow vector
  reti                   ; undefined
  ORG 0023h              ; serial I/O interrupt vector
  reti                   ; undefined
  ORG 30h                ; begin constant data space

_msg: DB '  PROGRAMMING', 0
msg: DB ' ATMEL AT89S8252 CMOS MICROCONTROLLER'
  DB ' WITH FLASH MEMORY AND SPI PORT', 0
  ORG 0100h              ; begin code space
  USING 0                ; Register bank 0 (RB0)

  mov sp, #(stack-1)     ; initialize stack pointer
  setb IT1               ; ext 1 interrupt edge triggered
  mov IE, #10000100b     ; enable ext 1 and global interrupts
0:
  jb   SWITCH, m1        ; check position of switch
  call rotate_msg        ; display message
  jmp m0                 ; again
1:
  call shift_msg         ; display message
  mov a, #3              ; pause 3 sec between displays
  call delay_sec         ;
  jmp m0                 ; again
tdown:
                         ; Respond to interrupt generated by serial programmer.
  clr  ea                ; prevent interrupts
  mov dptr, #pgm_msg     ; point to message
  call show_string       ; display message
  jmp $                  ; wait for reset

w_string:
Display null-terminated string pointed to by DPTR. The string is
eft-justified in the display. If the length of the string exceeds
he number of display positions the excess characters are ignored.
  call clear_display; begin by blanking display
  mov b, #(NDEVS*NELMS)  ; total display positions

s1:
  clr  a                 ; get char
  mov ca, @a+dptr;
  jz   gs2               ; done if string terminator
  call put_char          ; display char at position in B
```

```
    inc  dptr                   ; point to next char
    djnz b, gs1                 ; done when last position is filled
gs2:
    ret
ear_display:
                                ; Fill display with blanks.
                                ; All registers preserved.
    push acc
    push b
    mov b, #(NDEVS*NELMS)       ; total display positions
c1:
    mov a, #SPACE
    call put_char               ; write space char
    djnz b, c1                  ; do all positions
    pop b
    pop acc
    ret

ft_msg:
Display null-terminated string. Each character in the string,
in turn, enters the display from the right and is moved quickly
through each element of the display to its final position.
The string may contain any number of characters, including none.
If the length of the string exceeds the number of display
positions, the excess characters are ignored.

    call    clear_display       ; begin by blanking display
    mov     r5, #(NDEVS*NELMS)  ; total display positions
    mov     dptr, #usr_msg      ; point to message
s1:
    mov     b, #1               ; first display position
s2:
    clr     a
    movc    a, @a+dptr          ; get char
    jz      ps4                 ; done if string terminator
    call    put_char            ; display char at position in B
    mov     a, #25              ; 25 ms
    call    delay_ms            ; delay so char can be seen
    mov     a, b                ; set up for compare
    clr     c                   ; ready for subtraction
    subb    a, r5               ; compare next position to final
    jnc     ps3                 ; jump if char is in final position
    mov     a, #SPACE
    call    put_char            ; blank out char
```

```
    inc     b                   ; next position
    jmp     ps2
ps3:
    inc     dptr                ; point to next char
    djnz    r5, ps1             ; final position for next char
ps4:
        ret

otate_msg:
 ; Display null-terminated string. The string moves through the
 ; display, from right to left, with the display acting as a window
 ; onto the string. The string may contain any number of characters,
 ; including none.

    mov     dptr, #usr_msg      ; point to string
    clr     a                   ; get first char
    movc    a, @a+dptr;
    jz      dd11                ; blank display and exit if null string
    call    clear_display       ; begin by blanking display

    ; Phase I. Shift the string into the display from the
    ; right until the first character is in the left-most
    ; display element. If the string has fewer characters than
    ; the display has elements, fill the balance with blanks.

    mov     r7, #0              ; loop counter, one pass per element
dd1:
    mov     dptr, #usr_msg      ; point to string
    mov     b, r7               ; character position
    inc     b                   ; adjust
dd2:
    clr     a                   ; get next char
    movc    a, @a+dptr;
    jz      dd3                 ; jump if string terminator
    call    put_char            ; display char at position in B
    inc     dptr                ; point to next char
    djnz    b, dd2              ; loop until all positions written
    jmp     dd5                 ; next pass
dd3:                            ; encountered end of string
    mov     a, #SPACE           ; pad balance of display with blanks
    call    put_char            ; display char at position in B
    djnz    b, dd3              ; next position
dd5:
    mov     a, #150             ; 150 ms
```

```
   call    delay_ms                ; delay so string can be seen
   inc     r7                      ; next pass
   cjne    r7, #(NDEVS*NELMS), dd1 ; loop until all elements done
   ; Phase II. Shift the string THROUGH the display from
   ; the right until the last character is in the left-most
   ; display element. If the string has fewer characters than
   ; the display has elements, pad the balance with blanks.

   mov     dptr, #usr_msg          ; point to string
   inc     dptr                    ; start with the second char
d6:
   clr     a                       ; get char
   movc    a, @a+dptr;
   jz      dd11                    ; blank display and exit if string end
   push    dpl                     ; save string pointer
   push    dph
   mov     b, #(NDEVS*NELMS)       ; total char positions
d7:
   clr     a                       ; get next char
   movc    a, @a+dptr              ;
   jz      dd8                     ; jump if string terminator
   call    put_char                ; display char at position in B
   inc     dptr                    ; point to next char
   djnz    b, dd7                  ; loop until all positions written
   jmp     dd10                    ; next pass
d8:                                ; encountered end of string
   mov     a, #SPACE               ; pad balance of display with blanks
   call    put_char                ; display char at position in B
   djnz    b, dd8                  ; next position
d10:
   pop     dph                     ; restore string pointer
   pop     dpl                     ;
   inc     dptr                    ; point to next char
   mov     a, #150                 ; 150 ms
   call    delay_ms                ; delay so string can be seen
   jmp     dd6                     ; process next char
d11:
   call    clear_display           ; blank display
   mov     a, #150                 ; 150 ms
   call    delay_ms                ; delay
   ret

char:
isplay character in A at position indicated in B.
```

```
; All registers preserved.

    push    acc
    push    b
    mov     p0, a                   ; move character to output port

    ; Calculate device and element from display position.

    mov     a, b                    ; position 1..n
    dec     a                       ; convert to 0..n-1
    mov     b, #NELMS               ; elements per device
    div     ab                      ; A= device, B= element
    mov     p2, #0ffh               ; clear display control port
s0:
    cjne    a, #0, s1               ; check device number
    mov     a, #00010000b           ; device 0 select
    jmp     s5
s1:
    cjne    a, #1, s2
    mov     a, #00100000b           ; device 1 select
    jmp     s5
s2:
    cjne    a, #2, s3
    mov     a, #01000000b           ; device 2 select
    jmp     s5
s3:
    cjne    a, #3, s4
    mov     a, #10000000b           ; device 3 select
    jmp     s5
s4:
    jmp     init                    ; undefined device, restart
s5:
    orl     a, b                    ; add element selector
    xrl     a, #11110000b           ; invert device selector
    mov     p2, a                   ; write strobe low
    orl     a, #11110000b           ; reset device selector
    mov     p2, a                   ; write strobe high (latch data)
    pop     b
    pop     acc
    ret

dly_ms:
; Delay for 1 ms times the value in the accumulator.
```

```
        push    acc
        push    b
        mov     b, #0
dd:
        djnz    b, $            ; 500 us @ 12 MHz
        djnz    b, $            ; 500 us @ 12 MHz
        djnz    acc, dd
        pop     b
        pop     acc
        ret

elay_sec:
; Delay for 1 second times the value in the accumulator.

        push    acc
        push    b
        mov     b, a
ddd:
        mov     a, #250
        call    delay_ms        ; 250 ms
        call    delay_ms        ; 500 ms
        call    delay_ms        ; 750 ms
        call    delay_ms        ; 1000 ms
        djnz    b, ddd
        pop     b
        pop     acc
        ret

        END
```

ppendix 3: Code for AT89S8252 Programmer

```
ME AT89S8252_Programmer
he programmer powers up with the control signals to the target AT89S8252
active, allowing the program in the target to run normally. Upon receipt
 the first valid data record, the programmer puts the target into write
ode. The first and subsequent valid records are dissected as they are
ceived and their data is written into the target. Receipt of a valid
d-of-file record terminates programming and resets the target control
gnals, allowing the new program in the target to run.

ch record received is checked for validity. If it is invalid,
 receiver sends a NAK to the remote system and discards the record.
d records are not programmed into the target AT89S8252. Valid records
```

e programmed into the target AT89S8252 and verified. If verification
cceeds, an ACK is sent to the remote system. If verification fails,
e receiver sends CAN to abort the upload. Failure to verify is a fatal
ror. The target AT89S8252 will be left in program mode (held reset) so
at the incomplete or invalid code which it contains cannot be executed.

coming records must appear less than 30 seconds apart, or the line
dropped in preparation for the next call. If the remote system drops
e line, the programmer will wait 30 seconds before resetting. Calls
coming during this time are ignored.

ne programmer manages five lines ($\overline{\text{SHUTDN}}$, RST, SCK, MOSI, MISO)
nich control the target AT89S8252 and 4 lines which handle the modem
terface. The AT89S8252 control lines occupy bits of port 1 and the
odem interface lines bits of port 3, as defined in the EQUates.

ocedures SHOUT (SHift OUT) and SHIN (SHift IN) manage the serial transfer
data between the programmer and the target AT89S8252. The serial clock
generated and timed by software. The code meets timing requirements
nen executed by an AT89Cx051 microcontroller with a 12-MHz clock.
ode modifications may be required if a faster clock is substituted.

wo long period timers are implemented utilizing Timer Zero and members of
gister bank one. Timer Zero is configured in 16-bit mode and is loaded
th an initial count of zero, which yields the maximum delay of 65.5 ms
12 MHz). The timer is allowed to free-run, generating an interrupt
ch time the count rolls over from FFFF to 0000. At each interrupt, the
unts in each of the long period timers are decremented if their respective
erflow flags are not set. If the new count in either long timer is zero,
e corresponding overflow flag is set. It is not necessary to stop Timer
r to disable interrupts to reload the long timers, because they will
t be disturbed by the Timer 0 interrupt service routine whenever their
erflow flags are set. Because Timer 0 free-runs, it is not possible to
ow where in a period timing of an event begins. Therefore, one additional
unt should be added to the calculated long timer count to guarantee that
e timed interval is not short.

ng timer 0 is 16 bits, allowing a maximum timed interval of
er one hour. Long timer 1 is 8 bits, allowing a maximum timed
erval of 16 seconds.

e programmer software is compatible with the U.S. Robotics Sportster
,400-baud external modem and may require modifications if used with other
odems. The switches on the modem are set as follows:

1 UP DTR normal
2 DOWNNumeric result codes
3 DOWNDisplay result codes
4 DOWNSuppress command echo
5 UP Auto answer
6 UP CD normal
7 UP Load NVRAM defaults
8 DOWNSmart mode

lodem switch 7 specifies that the power on and reset configuration be
aded from NVRAM profile zero, which must contain the factory default
ardware flow control template. Other switch settings then override the
aded configuration. If NVRAM profile zero does not contain the hardware
ow control template, it may be restored with the following command
equence:

AT&F1&W0<ENTER>

ome of the switch functions can be controlled by software, but making
se of the switches simplifies the code required to initialize the modem.
he only additional commands which must be issued to the modem are:

&R1Ignore RTS,
&A0Disable ARQ result codes.

&R1" causes the modem to forward incoming data to the programmer regardless
the state of RTS. "&A0" suppresses the extended protocol result codes.
ote that suppression of the codes does not affect the connection. If it is
sired to disable Error Control, issue the command "&M0".

```
          EQU   0dh       ; carriage return
          EQU   0ah       ; line feed
K         EQU   6h        ; responses to remote system
K         EQU   15h       ;
N         EQU   18h       ;
UD_1200   EQU   0e8h      ; 1200 baud timer reload values
UD_2400   EQU   0f4h      ; 2400 baud
UD_9600   EQU   0fdh      ; 9600 baud
          EQU   '0'       ; modem status codes
GING      EQU   '2'       ;
NNECT_1200 EQU  '5'       ;
NNECT_2400 EQU  '10'      ;
NNECT_9600 EQU  '13'      ;
```

```
RIES    EQU   5          ; max attempts to access modem
ASE_1   EQU   0ach       ; erase chip function, first byte
ASE_2   EQU   04h        ; second byte
ABLE_1  EQU   0ach       ; enable write function, first byte
ABLE_2  EQU   53h        ; second byte
MMY     EQU   55h        ; function third byte
RITE_CODE EQU 02h        ; write code memory function (Flash)
AD_CODE EQU   01h        ; read code memory function
RITE_DATA EQU 06h        ; write data memory function (EEPROM)
AD_DATA EQU   05h        ; read data memory function
_lo     EQU   r2         ; long timer one low byte
_hi     EQU   r3         ; long timer one high byte
        EQU   r4         ; long timer two only byte
ex      EQU   r0         ; general purpose index register
sum     EQU   r5         ; running checksum on record
p       EQU   r6         ; temporary storage
nt      EQU   r7         ; loop counter
R_      BIT   p3.3       ; modem control signals
R_      BIT   p3.7       ;
T       BIT   p1.7       ; target control signals
UTDN_   BIT   p1.6       ;
K       BIT   p1.4       ;  serial clock
SI      BIT   p1.3       ;  serial data out
O       BIT   p1.2       ;  serial data in

        DSEG AT 20h

s       DATA  20h        ; misc flags
F       BIT   flags.0    ; long timer 0 overflow flag
F       BIT   flags.1    ; long timer 1 overflow flag

        ORG30h
type:   DS    1          ; record type
r_lo:   DS    1          ; record load address, low byte
r_hi:   DS    1          ; record load address, high byte
_len:   DS    1          ; record data byte count
_buf:   DS    32         ; storage for record data field
        ORG   60h        ; stack origin
k:      DS    20h        ; stack depth
N       DATA  87h        ; address of Power Control register
                         ;  (added to enlighten the assembler)
        CSEG
        ORG   0000h      ; power on/reset vector
        jmp   init
```

```
                ORG     0003h           ; external interrupt 0 vector
                reti                    ; undefined
                ORG     000Bh           ; timer 0 overflow vector
                jmp     timer_int
                ORG     0013h           ; external interrupt 1 vector
                reti                    ; undefined
                ORG     001Bh           ; timer 1 overflow vector
                reti                    ; undefined
                ORG     0023h           ; serial I/O interrupt vector
                jmp     serial_int
                ORG     40h             ; begin constant data space
tn_cmd:         DB      '+++', 0        ; modem return to command mode
set_cmd:        DB      'ATZ', CR, 0    ; modem reset string
                                        ;  must be last command on line and
                                        ;  modem returns code before executing
it_cmd:         DB      'AT&R1&A0', CR, 0    ; modem init string
ngup_cmd:       DB      'ATH', CR, 0    ; modem on-hook string
                ORG     0080h           ; begin code space
                USING 0                 ; register bank 0
t:
                mov     sp, #(stack-1)  ; initialize stack pointer
                call    initialize      ; initialize controller registers
                setb    LT0F            ; disable long timer 0
                setb    LT1F            ; disable long timer 1
                                        ; Initialize the modem.
                setb    TI              ; set transmit interrupt flag
                                        ;  (kludge for first use)
                setb    ET0             ; enable timer 0 interrupt
                call    modem_init      ; initialize modem
                clr     ET0             ; disable timer 0 interrupt
                jnc     m1              ; jump if modem init passes
                clr     EA              ; global interrupt disable
                orl     PCON, #1        ; idle the controller, reset exits
n1:
    ; Clear pending interrupts before enabling serial interrupts.
                jnb     TI, $           ; wait for transmitter to clear
                clr     TI              ; clear transmit interrupt flag
                clr     RI              ; clear receive interrupt flag
                setb    ES              ; enable serial ints to wake controller
                clr     F0              ; clear connect flag / PSW.5 bit
lle:
                orl     PCON, #1        ; idle the controller, serial int exits
                jnb     F0, idle        ; return to idle if not connected
                ; Connection has been established.
```

```
                ; Begin polling for valid record header.
                clr     ES              ; disable serial interrupts
                setb    TI              ; set transmit interrupt flag
                                        ;   (kludge for first use)
                clr     F0              ; clear program mode flag
                setb    ET0             ; enable timer 0 interrupt
n2:
                call    init_longtimer0 ; start 30-second timer
n3:
                call    get_char        ; get char, 1-second timeout
                jc      m8              ; try again if parity error or timeout
                cjne    a, #':', m8     ; try again if not record header
                                        ; Found header, process hex record.
                call    get_record      ; load and dissect record
                jnc     m4              ; jump if record is good
                mov     a, #NAK         ; tell sender record is bad
                call    send_char;
                jmp     m2              ; next record
n4:
                cjne    a, #0, m6       ; jump if record is not type zero
                ; Process record type zero (data).
                jb      F0, m5          ; jump if target is in write mode
                call    shutdown        ; notify target of impending doom
                call    erase_chip      ; erase target
                call    set_pgm         ; place target in write mode
                setb    F0              ; flag target in write mode
n5:
                call    write_record    ; program data into target
                call    verify_record   ; verify program data
                jnc     m7              ; jump if verify OK
                mov     a, #CAN         ; tell sender to abort
                call    send_char;
                jmp     m9              ; hang up and reset for next call
6:
  ; Process record type one (end-of-file).
                call    clear_pgm       ; take target out of write mode
                clr     F0              ; flag target not in write mode
7:
                mov     a, #ACK         ; tell sender record OK
                call    send_char       ;
                jmp     m2              ; next record
8:
                jnb     LT0F, m3        ; poll until timer times out
9:                                      ; timer timed out or upload cancelled
```

```
        call    hang_up         ; break the connection
        clr     ET0             ; disable timer 0 interrupt
        jmp     m1              ; return controller to idle
rial_int:
Process serial interrupt. Interrupts due to transmit done are
cleared and ignored. If interrupt is due to receive data ready,
check for a modem connect code, and set the connect flag.
The procedure includes code for identifying both single- and
double-character connect codes, but both may not be active
simultaneously. The code for identifying double-character
connect codes is dependent on the receive baud rate.
Serial interrupts are enabled elsewhere.

        clr     F0              ; clear connect flag
        clr     TI              ; clear transmit interrupt flag
        jnb     RI, si2         ; exit if not receive data ready
        mov     a, SBUF         ; get character into accumulator
        mov     c, p            ; carry set for odd parity (error)
        jc      si1             ; ignore char if parity error
        ; Test for single-character 1200-baud connect code.
        anl     a, #7fh         ; strip off parity (eighth) bit
        cjne    a, #CONNECT_1200, si1                   ; ignore char if wrong code
        ; Test for double-character 9600-baud connect code.
        anl     a, #7fh         ; strip off parity (eighth) bit
        cjne    a, #(HIGH CONNECT_9600), si1; ignore wrong char
        clr     RI              ; reset receive flag
        mov     a, #2           ; expect next char in about 1 ms
        call    delay_ms        ; wait for next char
        jnb     RI, si2         ; exit if not receive data ready
        mov     a, SBUF         ; get character into accumulator
        mov     c, p            ; carry set for odd parity (error)
        jc      si1             ; ignore char if parity error
        anl     a, #7fh         ; strip off parity (eighth) bit
        cjne    a, #(LOW CONNECT_9600), si1; ignore wrong char
        setb    F0              ; set connect flag
1:
        clr     RI              ; reset receive flag
2:
        reti

er_int:
Process Timer Zero interrupt, which occurs about every 65.5 ms.
Each long timer count is decremented if its overflow flag is clear.
When a long timer count reaches zero, its overflow flag is set.
```

Counts are reloaded and overflow flags are reset elsewhere.

```
        push    psw             ; save flags
        setb    RS0             ; select register bank one
        jb      LT0F, ti2       ; skip if long timer 0 overflow set
        cjne    lt0_lo, #0, ti1 ; test low byte
        dec     lt0_hi          ; low byte is zero, borrow from high
1:
        djnz    lt0_lo, ti2     ; dec low byte, skip if not zero
        cjne    lt0_hi, #0, ti2 ; low byte is zero, test high byte
                                ; both bytes equal zero
        setb    LT0F            ; set overflow flag
2:
        jb      LT1F, ti3       ; skip if long timer 1 overflow set
        djnz    lt1, TI3        ; decrement count and skip if not zero
        setb    LT1F            ; count is zero, set overflow flag
3:
        pop     psw             ; restore flags and reg bank zero
        reti
```

alize:

Initialize controller registers and I/O lines.

```
        mov     PCON, #0            ; initialize power control register
        mov     IE, #0              ; deactivate all interrupts
        mov     SCON, #01000000b; serial port mode 1
        mov     TMOD, #00100001b; timer 1 8-bit auto-reload,
                                    ;  timer 0 16-bit
        mov     TH1, #BAUD_1200 ; timer 1 reload value
        mov     TH1, #BAUD_9600 ; timer 1 reload value
        mov     TCON, #01000000b; start timer 1
        mov     TL0, #0             ; set timer 0 to max count
        mov     TH0, #0             ;
        setb    TR0                 ; start timer 0
        setb    REN                 ; enable serial reception
        setb    EA                  ; global interrupt enable
        ; Initialize I/O lines.
        setb    DTR_
        setb    SHUTDN_
        setb    MISO
        setb    MOSI
        clr     SCK
        clr     RST                 ; remove reset from target
        ret
```

```
odem_init:
; Reset and initialize the modem.
; Return with carry set if modem fails to respond as expected.
                clr     DTR_                    ; assert DTR to talk to modem

    ; First must ensure that the modem is in command mode.
                mov     a, #1                   ; wait 1 second
                call    delay_sec               ;
                mov     dptr, #attn_cmd         ; point to attention string
                call    send_string             ; transmit string
                mov     a, #1                   ; wait 1 second
                call    delay_sec               ;
                ; Reset modem, causing the switches to be read.
                mov     dptr, #reset_cmd        ; point to reset string
                call    modem_cmd               ; transmit string
                jc      nn1                     ; jump on fail
                mov     a, #1                   ; wait 1 second before next command
                call    delay_sec               ;
    ; Modem is powered up and on-line.
    ; Send required software parameters.
                mov     dptr, #init_cmd         ; point to init string
                call    modem_cmd               ; transmit string
                jnc     nn2                     ; jump on pass
nn1:
                ; Modem is misbehaving, so deactivate it.
                ; The controller must be reset to exit this state.
                setb    DTR_                    ; deassert DTR to deactivate modem
nn2:
                ret
ang_up:
        ; Force the modem to drop the line.
                ; First must return the modem to command mode.
                mov     a, #1                   ; wait 1 second
                call    delay_sec               ;
                mov     dptr, #attn_cmd         ; point to attention string
                call    send_string             ; transmit string
                mov     a, #1                   ; wait 1 second
                call    delay_sec               ;
                ; Issue command to hang up.
                mov     dptr, #hangup_cmd       ; point to hang up string
                call    modem_cmd               ; transmit string
                jnc     hh                      ; jump on pass
                ; The polite way didn't work, so drop DTR.
```

```
                ; The controller must be reset to exit this state.
                setb    DTR_            ; force modem to drop the line
hh:
                ret

odem_cmd:
; Transmit command string to modem and validate the response.
; Return with carry set if modem fails to respond as expected,
; or if excessive parity errors or receive timeouts occur.
; Valid responses consist of a byte code followed by a carriage
; return. Parity errors and timeouts cause the command to be
; resent. Expected delays for command responses are absorbed
; by GET_CHAR. On entry, DPTR must point to a null-terminated
; command string.

                push    b
                mov     b, #MTRIES      ; number of attempts
mm1:
                call    send_string     ; transmit command string
                clr     RI              ; discard any waiting character
mm2:
                call    get_char        ; receive result code
                jc      mm3             ; jump on parity error or timeout
                cjne    a, #OK, mm2     ; loop if response is not valid
                call    get_char        ; receive carriage return
                jc      mm3             ; jump on parity error or timeout
                cjne    a, #CR, mm2     ; loop if response is not valid
                                        ; valid response complete
                clr     c               ; clear error flag
                jmp     mm4             ; return
nm3:
                djnzb,  mm1             ; resend command
                setb    c               ; out of retries, set error flag
nm4:
                pop     b
                ret

d_string:
Transmit string pointed to by DPTR.
String may be of any length, but must be null-terminated.

                push    acc
                push    dpl
                push    dph
```

```
ss1:
        clr     a
        movc    a, @a+dptr      ; get character
        jz      ss2             ; check for terminator
        call    send_char       ; send character
        inc     dptr            ; point to next character
        jmp     ss1
ss2:
        pop     dph
        pop     dpl
        pop     acc
        ret
```

```
nd_char:
 Wait for transmitter to clear, add even parity bit to character
 in accumulator and transmit it. Does not wait for transmitter
 to clear before returning.

        jnb     TI, $           ; wait here for transmitter to clear
        clr     TI              ; clear transmit flag
        push    acc             ; save char
        movc,   p               ; get parity bit
        mov     acc.7, c        ; add parity bit to data
        mov     SBUF, a         ; load character into transmitter
        pop     acc             ; restore char
        ret
```

```
_char:
Read a character from the serial port and check for even parity.
Return the character in the accumulator with parity stripped off.
The routine will wait for approximately 1 second before timing
out. Return with carry set on parity error or timeout.

        jb      RI, gc2         ; jump if char is waiting
        call    init_longtimer1 ; start 1-second timer
c1:
        jb      RI, gc2         ; exit loop when char received
        jnb     LT1F, gc1       ; loop until timer times out
        setbc                   ; set error flag
        jmp     gc3             ; return
c2:
        mov     a, SBUF         ; get character into accumulator
        mov     c, p            ; carry set for odd parity (error)
        anl     a, #7fh         ; strip off parity (eighth) bit
```

```
        clr     RI              ; reset receive flag
gc3:
        ret

t_byte:
; Read two hexadecimal ASCII characters from the serial port
; and return their binary equivalent in the accumulator.
; Return with carry set if either character was invalid or
; contained a parity error.

        call    get_char        ; get first char from serial port
        jc      gb              ; exit on parity error
        call    ascii2bin       ; convert hex to binary
        jc      gb              ; exit on invalid char
        swap    a               ; first hex digit times 16
        mov     b, a            ; save value
        call    get_char        ; get second char from serial port
        jc      gb              ; exit on parity error
        call    ascii2bin       ; convert hex to binary
        jc      gb              ; exit on invalid char
        orl     a, b            ; combined binary equivalent
b:
        ret

ii2bin:
Convert hexadecimal digit in the accumulator to its binary
equivalent and return it in the accumulator. Valid hex digits
are 0..9 and A..F (upper case only). Return with carry set
if the character received is not a valid hex digit.

        mov     temp, a         ; save char
        clr     c               ; prepare for subtraction
        subb    a, #('9'+1)     ; compare to '9'
        jnc     a1              ; jump if char above '9'
        mov     a, temp         ; get original char
        clr     c               ; prepare for subtraction
        subb    a, #'0'         ; compare to '0'
        jmp     a4              ; return error if char below '0'
                                ;   else binary value in accumulator
1:
        mov     a, temp         ; get original char
        subb    a, #('F'+1)     ; compare to 'F'
        cpl     c               ; invert error flag
        jc      a4              ; return error if char is above 'F'
```

```
mov     a, temp             ; get original char
subb    a, #'A'             ; compare to 'A'
jc      a4                  ; return error if char is below 'A'

add     a, #10              ; adjust binary value

ret
```

cord:
d and dissect record. Two record types are accepted: data and
-of-file. If the record type is data, the appropriate values
extracted and stored. If the record type and checksum are
d, the carry bit is cleared and the record type is returned
ne accumulator. Return with carry set to signal an invalid
ord type, checksum error, or other problem. Errors returned
outine GET_BYTE (invalid char or parity) cause an immediate
m with carry set.

```
mov     chksum, #0          ; clear running checksum
call    get_byte            ; get record data length field
jc      rr4                 ; jump on error
mov     data_len, a         ; save data length
clr     c                   ; prepare for subtraction
subb    a, #(16+1)          ; data length limited to 16 bytes
jnc     rr4                 ; jump if max size exceeded
call    get_byte            ; get high byte of load address field
jc      rr4                 ; jump on error
mov     laddr_hi, a         ; save it
call    get_byte            ; get low byte of load address field
jc      rr4                 ; jump on error
mov     laddr_lo, a         ; save it
call    get_byte            ; get record type field
jc      rr4                 ; jump on error
mov     rec_type, a         ; save type
cjne    a, #0, rr2          ; jump if not type zero (data)
; Process data in data type record.
mov     index, #data_buf    ; pointer to data buffer
mov     kount, data_len     ; byte counter

call    get_byte            ; get data from serial port
jc      rr4                 ; jump on error
mov     @index, a           ; save data in buffer
add     a, chksum           ; update checksum
```

```
        mov     chksum, a           ;
        inc     index               ; point to next location
        djnz    kount, rr1          ; decrement byte count and loop
        jmp     rr3                 ; done with data, do checksum
r2:
        mov     a, rec_type         ; get record type
        cjne    a, #1, rr4          ; jump if not type one (end-of-file)
r3:
        ; Process checksum.
        call    get_byte            ; get record checksum
        jc      rr4                 ; jump on error
        add     a, chksum           ; update running checksum
        add     a, data_len         ;
        add     a, laddr_lo         ;
        add     a, laddr_hi         ;
        add     a, rec_type         ;
        jnz     rr4                 ; jump if record checksum is not zero
                                    ; Discard CR/LF which terminates record.
        call    get_byte
        jc      rr4                 ; jump on error
        call    get_byte
        jc      rr4                 ; jump on error
        mov     a, rec_type         ; return record type in accumulator
        clr     c                   ; no errors
        jmp     rr5                 ; return
4:
                                    ; Error: data field too large, invalid type or bad checksum.
        setb    c                   ; set error flag
5:
        ret

e_record:
  ; Write the data extracted from the most recently received record
  ; into the target AT89S8252. Timing delays are enforced by software.
  ; This routine assumes that the target has already been prepared
  ; for programming. Returns nothing.

        mov     r2, laddr_lo        ; save low byte of load address
        mov     r3, laddr_hi        ; save high byte of load address
        mov     a, r3               ; get high byte of load address
        anl     a, #00011111b       ; isolate 5 bits
        rl      a                   ; move 5 bits to top
        rl      a                   ;
        rl      a                   ;
```

```
        orl     a, #WRITE_CODE  ; specify code write function
        mov     temp, a         ; save adjusted high byte
        mov     index, #data_buf ; pointer to data buffer
        mov     kount, data_len ; byte counter
p1:
        mov     a, temp         ; send adjusted high byte of address
        call    shout           ;
        mov     a, r2           ; send low byte of address
        call    shout           ;
        mov     a, @index       ; send data from buffer
        call    shout           ;
        mov     a, #3           ; wait 3 ms
        call    delay_ms
        ; Next address.
        mov     a, r2           ; get low byte of address
        add     a, #1           ; increment low byte
        movr2, a                ; save incremented value
        jnc     pp2             ; jump if no carry out of low byte
        ; carry out of low byte
        mov     a, r3           ; get high byte of address
        add     a, #1           ; increment high byte
        mov     r3, a           ; save incremented value
        anl     a, #00011111b   ; isolate 5 bits
        rl      a               ; move 5 bits to top
        rl      a               ;
        rl      a               ;
        orl     a, #WRITE_CODE  ; specify code write function
        mov     temp, a         ; save adjusted high byte
2:
        ; Next data.
        inc     index           ; point to next buffer location
        djnz    kount, pp1      ; decrement byte count and loop
        ret

y_record:
erify the data extracted from the latest record against that
ritten into the target AT89S8252. Timing delays are enforced by
oftware. This routine assumes that the target has already been
repared for programming. Return with carry set if verify fails.

        mov     r2, laddr_lo    ; save low byte of load address
        mov     r3, laddr_hi    ; save high byte of load address
        mov     a, r3           ; get high byte of load address
        anl     a, #00011111b   ; isolate 5 bits
```

```
        rl      a               ; move 5 bits to top
        rl      a               ;
        rl      a               ;
        orl     a, #READ_CODE   ; specify code read function
        mov     temp, a         ; save adjusted high byte
        mov     index, #data_buf ; pointer to data buffer
        mov     kount, data_len ; byte counter
vv1:
        mov     a, temp         ; send adjusted high byte of address
        call    shout           ;
        mov     a, r2           ; send low byte of address
        call    shout           ;
        ; Read data and verify.
        call    shin            ; read data
        mov     b, @index       ; get record data
        cjne    a, b, vv2       ; jump on verify fail
        jmp     vv3             ; verify OK, do next address
vv2:
        setb    c               ; set error flag
        jmp     vv5             ; return
vv3:
        ; Next address.
        mov     a, r2           ; get low byte of address
        add     a, #1           ; increment low byte
        mov     r2, a           ; save incremented value
        jnc     vv4             ; jump if no carry out of low byte
        ; carry out of low byte
        mov     a, r3           ; get high byte of address
        add     a, #1           ; increment high byte
        mov     r3, a           ; save incremented value
        anl     a, #00011111b   ; isolate 5 bits
        rl      a               ; move 5 bits to top
        rl      a               ;
        rl      a               ;
        orl     a, #READ_CODE   ; specify code write function
        mov     temp, a         ; save adjusted high byte
v4:
        ; Next data.
        inc     index           ; point to next buffer location
        djnz    kount, vv1      ; decrement byte count and loop
        clr     c               ; clear error flag
v5:
        ret
```

```
nout:
; Shift out a byte, most significant bit first.
; SCK expected low on entry. Return with SCK low.
; Called with data to send in A.

        push    b
        mov     b, #8           ; bit counter
x42:
        rlc     a               ; move bit into CY
        mov     MOSI, c         ; output bit
        nop                     ; enforce data setup
        nop                     ;
        setb    SCK             ; raise clock
        nop                     ; enforce SCK high
        nop                     ;
        nop                     ;
        nop                     ;
        clr     SCK             ; drop clock
        djnz    b, x42          ; next bit
        pop     b
        ret

in:
 Shift in a byte, most significant bit first.
 SCK expected low on entry. Return with SCK low.
 Returns received data byte in A.

        push    b
        mov     b, #8           ; bit counter
:43:
        setb    SCK             ; raise clock
        mov     c, MISO         ; input bit
        rlc     a               ; move bit into byte
        nop                     ; enforce SCK high
        nop                     ;
        clr     SCK             ; drop clock
        nop                     ; enforce SCK low
        nop                     ;
        djnz    b, x43          ; next bit
        pop     b
        ret

se_chip:
Erase target AT89S8252.
```

```
        setb    RST             ; force target into reset
        mov     a, #ERASE_1     ; send first byte of erase function
        call    shout           ;
        mov     a, #ERASE_2     ; send second byte
        call    shout           ;
        mov     a, #DUMMY       ; send third byte
        call    shout           ;
        mov     a, #10          ; wait 10 milliseconds
        call    delay_ms        ;
        clr     RST             ; remove reset from target
        ret

utdown:
; Force target to abandon execution of its internal program.

        clr     SHUTDN_         ; notify target of impending reset
        mov     a, #5           ; give target 5 ms to shut down
        call    delay_ms        ;
        setb    SHUTDN_         ; deassert interrupt
        ret

t_pgm:
; Prepare the target AT89S8252 for programming.

        setb    RST             ; force target into reset
        mov     a, #1           ; wait 1 ms (arbitrary)
        call    delay_ms        ;
        ; Enable writes to code and data memory.
        mov     a, #ENABLE_1    ; send first byte of enable code
        call    shout           ;
        mov     a, #ENABLE_2    ; send second byte
        call    shout           ;
        mov     a, #DUMMY       ; send third byte
        call    shout           ;
        ret
ar_pgm:
        ; Allow target AT89S8252 to resume execution of its own program.

        clr     RST             ; remove reset from target
        ret

_longtimer0:
Load and start long timer 0.
```

; System Timer 0 count loaded and interrupt enabled elsewhere.

```
        setb    LT0F            ; disable counter
        setb    RS0             ; select register bank one
        mov     lt0_lo, #0c8h   ; load 30-second count
        mov     lt0_hi, #1      ;
        clr     RS0             ; back to bank zero
        clr     LT0F            ; enable counter
        ret
```

t_longtimer1:
; Load and start long timer 1.
; System Timer Zero count loaded and interrupt enabled elsewhere.

```
        setb    LT1F            ; disable counter
        setb    RS0             ; select register bank one
        mov     lt1, #17        ; load 1-second count
        clr     RS0             ; back to bank zero
        clr     LT1F            ; enable counter
        ret
```

lay_ms:
; Delay for 1 ms times the value in the accumulator.

```
        push    acc
        push    b
        mov     b, #0
dd:
        djnz    b, $            ; 500 us @ 12 MHz
        djnz    b, $            ; 500 us @ 12 MHz
        djnz    acc, dd
        pop     b
        pop     acc
        ret
```

lay_sec:
Delay for 1 s times the value in the accumulator.

```
        push    acc
        push    b
        mov     b, a
ddd:
        mov     a, #250
        call    delay_ms; 250 ms
        call    delay_ms; 500 ms
```

```
call    delay_ms; 750 ms
call    delay_ms; 1000 ms
djnz    b, ddd
pop     b
pop     acc
ret

END
```

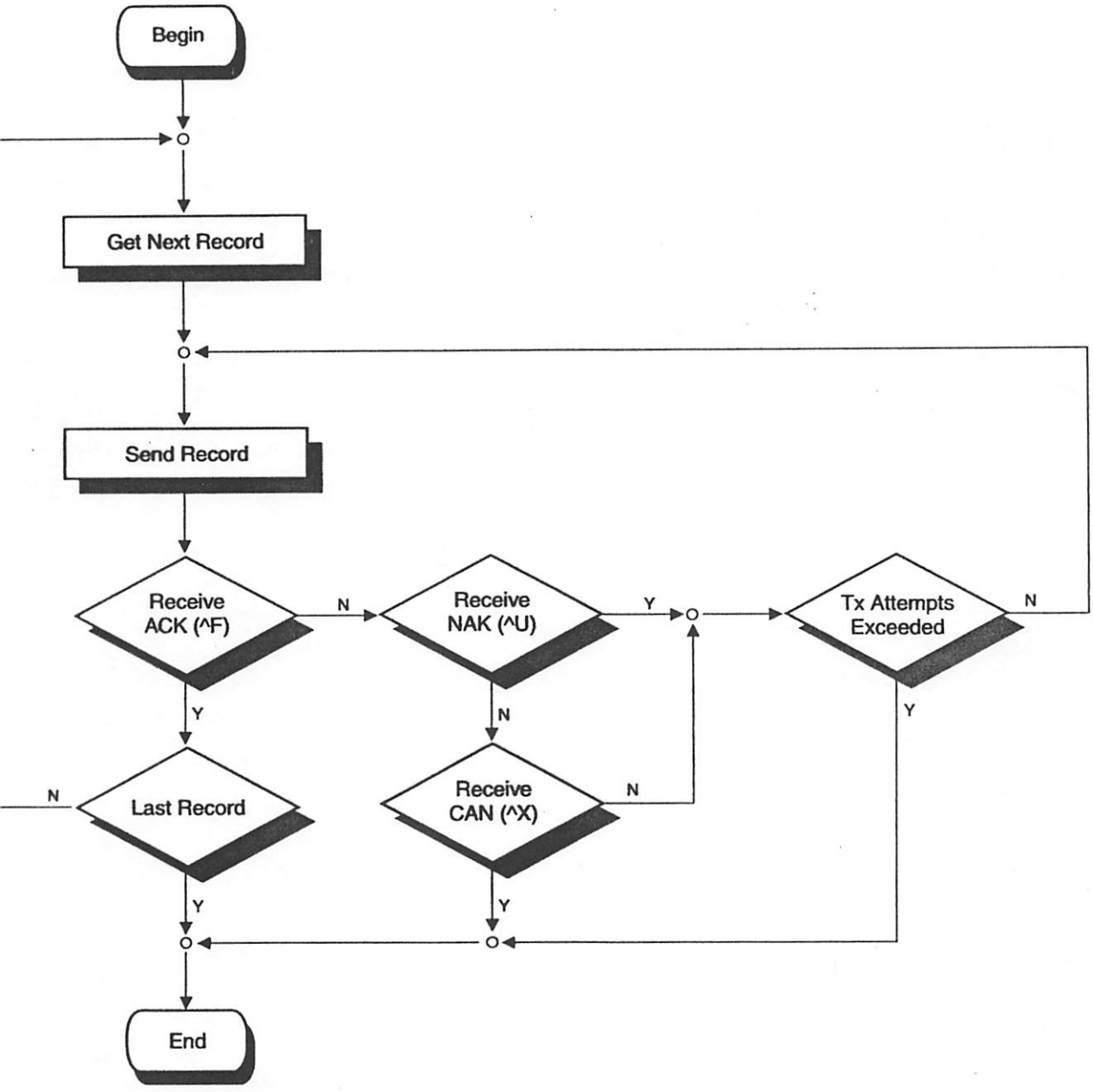
ppendix 4: ASPECT Script for Procomm Plus

```
PROCOMM ASPECT script to read and transmit an Intel hex file.
The script does not set up communications parameters, initialize the
nodem, dial out or establish a connection with the receiver; this is
lone manually via the PROCOMM Connection Directory.
Each record in the hex file is terminated by a CR/LF. The receiver is
xpected to respond with an ACK after each record is validated and
rogrammed into the target processor. If the receiver cannot validate
ne record, it responds with a NAK. If the receiver cannot verify the
ecord data after programming the target processor, it responds with
 CAN, which tells the transmitter to abort the upload. The transmitter
aits 2 seconds between records for a response. If a response is not
eceived in the allowed interval, or if the response is other than an
CK or a CAN, the record is retransmitted.
#define ACK             6           ; ^F
#define NAK             21          ; ^U
#define CAN             24          ; ^X
#define MAXRETRIES  4
roc main
tring filename, record
nteger retry, rxcode
dlgfopen "Select HEX File" "*.hex" single filename; get file name
 failure          ; get file name failed
   exit
ndif
 filename   ; validate path and file name
   if fopen 0 filename read; open file for read
      fgets 0 record; read record
   else
      errormsg "FILE OPEN FAILED"
      exit
   endif
lse
   errormsg "FILE DOES NOT EXIST"
   exit
ndif
et aspect rxdata on                    ; script processes receive data
hile not feof 0                        ; check for EOF
   termwrites record                   ; show record
   rxflush                             ; purge pending receive data
   transmit record raw                 ; send record including CR/LF
   comgetc rxcode 2                    ; wait max 2 seconds for answer
   call show_rxcode with rxcode        ; show received code
   retry = 0                           ; initialize counter
```

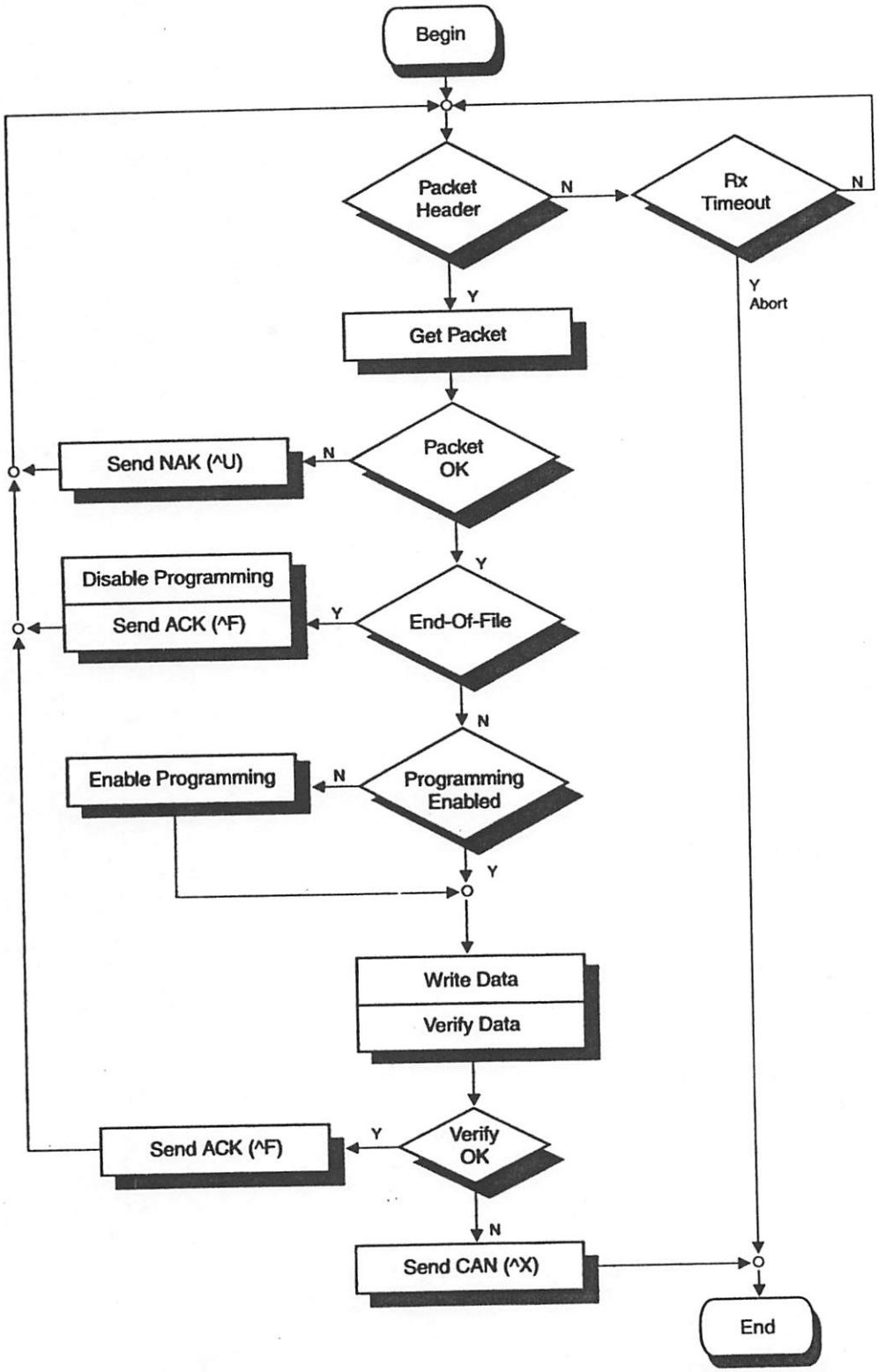
```
  while (rxcode != ACK) && (retry < MAXRETRIES)
      if (rxcode == CAN); abort ordered by remote
          errormsg "UPLOAD ABORTED BY REMOTE"
          fclose 0; close file
          set aspect rxdata off
          exit
      endif
      termwrites "Resending record^M^J"
      termwrites record; show record
      rxflush                          ; purge pending receive data
      transmit record raw              ; send record
      comgetc rxcode 2                 ; get response
      call show_rxcode with rxcode; show received code
      ++retry                          ; advance counter and try again
  endwhile
  if (rxcode != ACK)
      errormsg "EXCESSIVE RETRIES: UPLOAD ABORTED"
      fclose 0 ; close file
      set aspect rxdata off
      exit
  endif
  fgets 0 record; read next record
dwhile
rmwrites "End of file^M^J"
ose 0          ; close file
t aspect rxdata off
it
roc
 show_rxcode
ram integer rxcode
rmmsg "%#X`r`n", rxcode
itch rxcode
 case -1
     termwrites "Timed out^M^J"
 endcase
 case ACK
     termwrites "Received ACK^M^J"
 endcase
 case NAK
     termwrites "Received NAK^M^J"
 endcase
 case CAN
     termwrites "Received CAN^M^J"
 endcase
```

```
    default
        termwrites "Received garbage^M^J"
    endcase
ndswitch
lproc
```

re 4. FTP Transmit Mode



5. FTP Receive Mode



Prosedur Pengujian modul TLP / RLP

AN-03

Oleh: Tim Digiware

Untuk mengetahui bahwa modul RF tersebut bisa menerima atau mengirimkan data dengan baik, terlebih dahulu harus dilakukan suatu pengujian. Cara menguji modul RF biasanya hanya memberikan logika 1 atau 0 pada modul transmitter, kemudian sinyal tersebut diterima oleh modul penerima dan outputnya juga 1 atau 0 sesuai dengan logika yang dikirimkan. Pada modul – modul tertentu hal tersebut tidak dapat dilakukan, karena adanya batas kecepatan minimal di dalam pengiriman data.

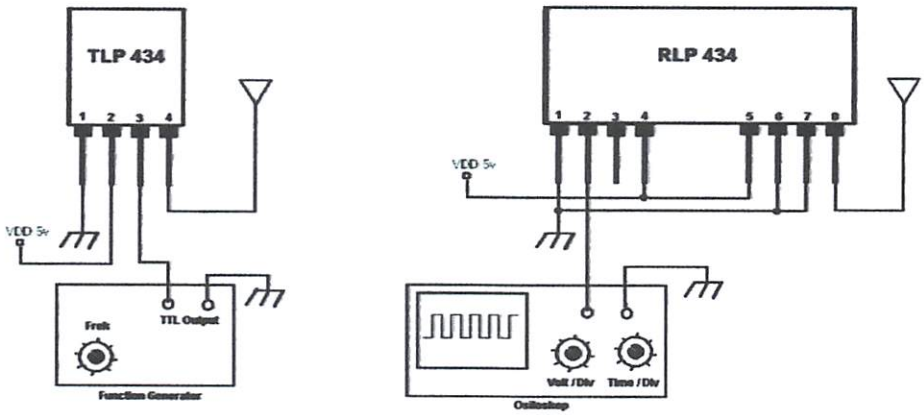
❖ Macam – macam pengujian

Modul yang akan dipakai sebagai contoh adalah TLP/RLP 434 dan 916 buatan Laipac. Modul ini tidak dapat diuji dengan cara memberikan logika 1 atau 0 saja, tetapi harus diberikan pulsa. Sumber pulsa bermacam – macam antara lain Function generator, Timer, dll. Di dalam artikel ini akan dilakukan pengujian dengan 2 cara yaitu menggunakan IC HT12E dan HT12D atau Osiloskop dengan Function generator. Apabila belum memliki osiloskop dan function generator disarankan memakai IC HT12E, HT12D serta beberapa komponen tambahan. Kedua IC ini dibuat oleh Holtek dan sering dipasangkan pada modul RF tipe TLP/RLP 434 dan 916, sebagai contoh aplikasi pengiriman / penerimaan data.

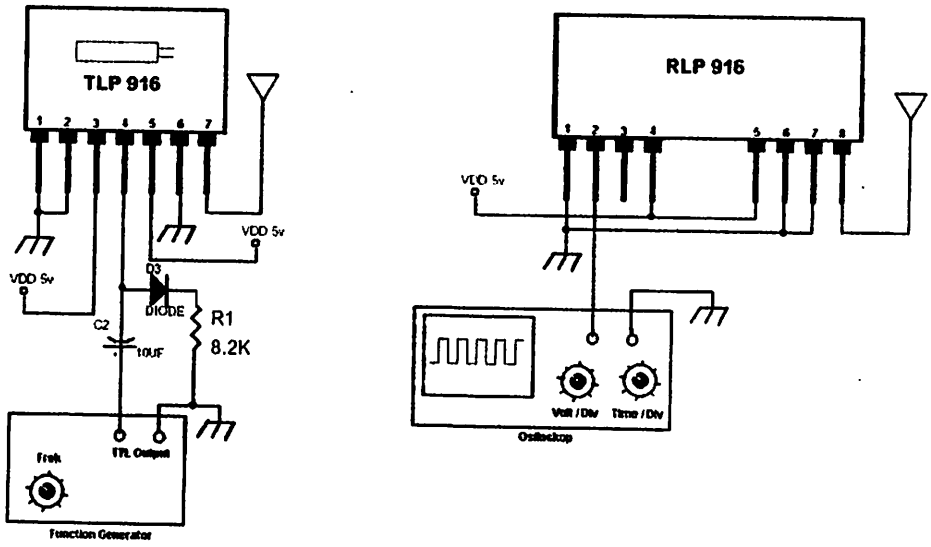
❖ Prosedur Pengujian menggunakan Osiloskop

Untuk melakukan pengujian terhadap modul ini menggunakan osiloskop dan Function generator, membutuhkan beberapa peralatan antara lain : Function generator, Osiloskop dan Power supply serta beberapa komponen tambahan khusus untuk tipe TLP916. Sedangkan tahap pengujian adalah sebagai berikut:

- Rangkai modul – modul tersebut seperti gambar di bawah



Gambar 1 . Rangkaian untuk menguji TLP/RLP 434.



Gambar 2. Rangkaian untuk menguji TLP/RLP 916.

TLP 434	RLP 434
1. GND	1. GND
2. VCC	2. Digital Output
3. RF Output	3. Linier Output
4. Digital Data Input	4. VCC
	5. VCC
	6. GND
	7. GND
	8. Antenna (17cm)

TLP 916	RLP 916
1. DCL	1. GND
2. CHP	2. Digital Output
3. E	3. Linier Output
4. DTH	4. VCC
5. VCC	5. VCC
6. GND	6. GND
7. Antenna	7. GND
	8. Antenna (17 cm)

Tabel 1. Konfigurasi Pin TLP/RLP.

- Konfigurasikan Function Generator anda dengan output TTL. **(PERINGATAN: JANGAN MEMASUKKAN INPUT UNTUK MODUL TLP DENGAN SINYAL ANALOG KARENA MODUL DAPAT RUSAK).**

- Atur frekuensi pada function generator dan lihat bentuk sinyal output pada RLP. Apabila bentuk sinyal output yang keluar pada RLP ikut berubah maka kedua modul tersebut masih baik.

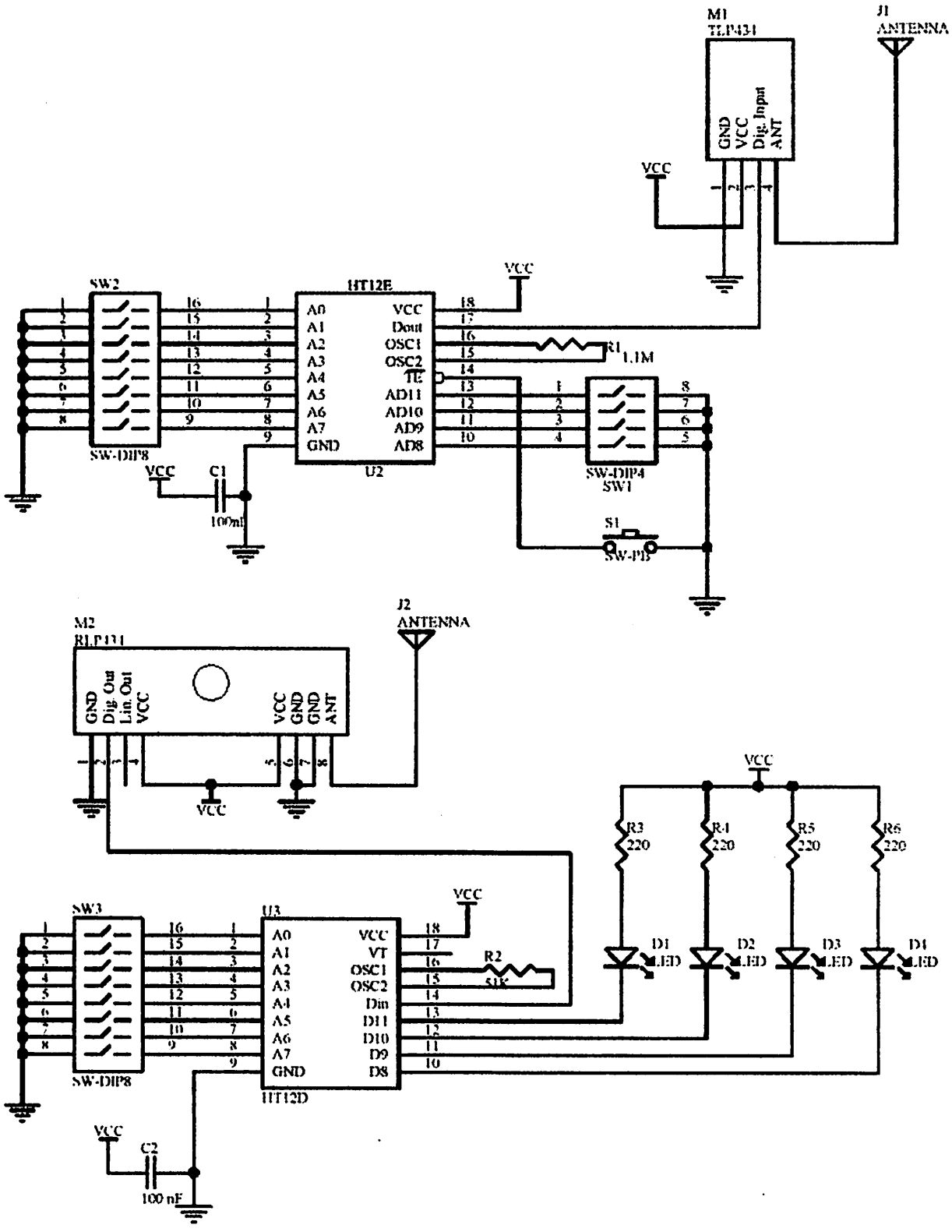
❖ **Prosedur Pengujian menggunakan IC HT12E dan HT12D**

Pengujian dilakukan dengan cara merangkai IC HT12E dan HT12D seperti pada gambar 3. Cara untuk melakukan pengujian adalah sebagai berikut:

1. Masukkan Alamat yang sama pada HT12E dan HT12D dengan cara menggeser kedua DIP Switch kearah ground. Contoh: apabila alamat HT12E adalah 0 maka HT12D juga 0.

2. Kirimkan data dari HT12E dengan cara menggeser SW1 menuju ground.

3. Lalu tekan push button S1 pada HT12E. Jika modul RLP dan TLP bekerja maka D1 – D4 akan menyala sesuai dengan penggeseran SW1 yang terdapat pada HT12E.



Gambar 3. Rangkaian HT12x pada modul RF.

Prosedur Pengujian modul TLP / RLP

AN-03

Oleh: Tim Digiware

Untuk mengetahui bahwa modul RF tersebut bisa menerima atau mengirimkan data dengan baik, terlebih dahulu harus dilakukan suatu pengujian. Cara menguji modul RF biasanya hanya memberikan logika 1 atau 0 pada modul transmitter, kemudian sinyal tersebut diterima oleh modul penerima dan outputnya juga 1 atau 0 sesuai dengan logika yang dikirimkan. Pada modul – modul tertentu hal tersebut tidak dapat dilakukan, karena adanya batas kecepatan minimal di dalam pengiriman data.

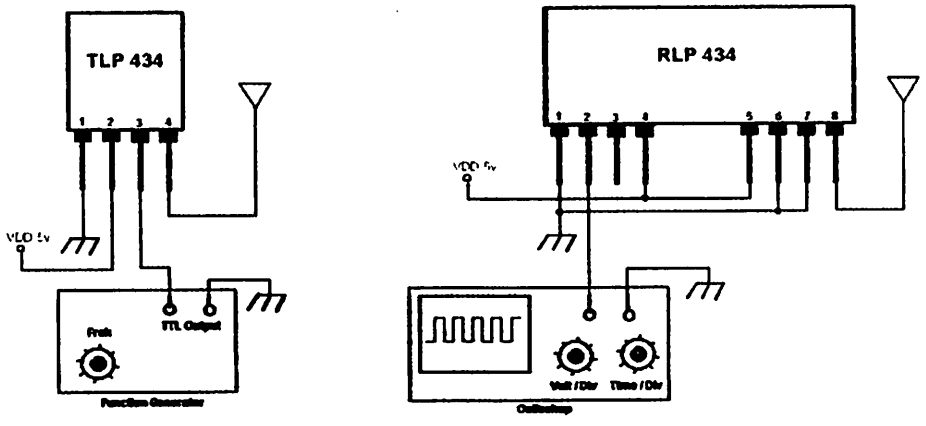
❖ Macam – macam pengujian

Modul yang akan dipakai sebagai contoh adalah TLP/RLP 434 dan 916 buatan Laipac. Modul ini tidak dapat diuji dengan cara memberikan logika 1 atau 0 saja, tetapi harus diberikan pulsa. Sumber pulsa bermacam – macam antara lain Function generator, Timer dll. Pada modul ini dilakukan pengujian dengan 2 cara yaitu menggunakan IC HT12E dan HT12D dan Osiloskop sebagai monitor IC HT12E, HT12D serta beberapa komponen tambahan. Kedua IC ini dibuat oleh Holtek sebagai pengiriman / penerimaan data.

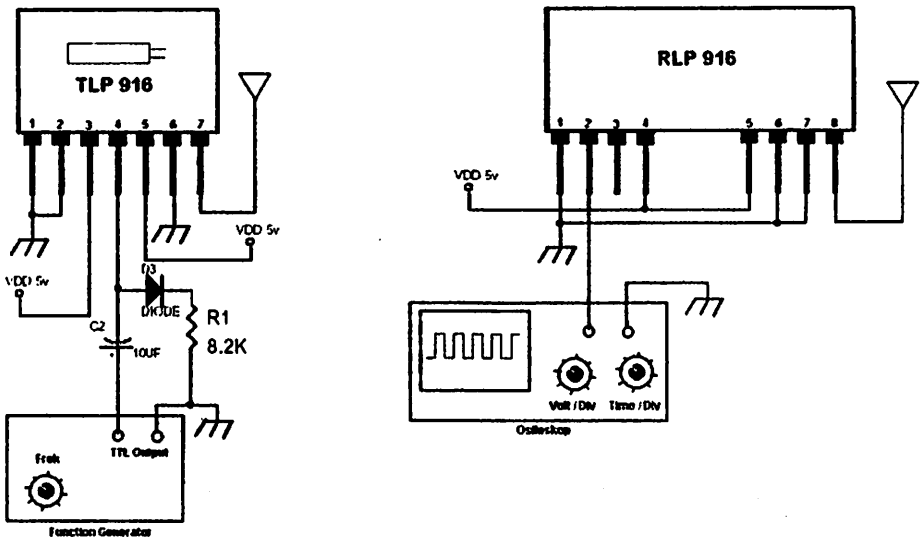
❖ Prosedur Pengujian menggunakan Osiloskop

Untuk melakukan pengujian terhadap modul ini menggunakan osiloskop dan Function generator, membutuhkan beberapa peralatan antara lain : Function generator, Osiloskop dan Power supply serta beberapa komponen tambahan. Untuk TLP 434 Gelombang input pengujian adalah sebagai berikut:

- Rangkai modul – modul tersebut seperti gambar di bawah



Gambar 1 . Rangkaian untuk menguji TLP/RLP 434.



Gambar 2. Rangkaian untuk menguji TLP/RLP 916.

TLP 434	RLP 434
1. GND	1. GND
2. VCC	2. Digital Output
3. RF Output	3. Linier Output
4. Digital Data Input	4. VCC
	5. VCC
	6. GND
	7. GND
	8. Antenna (17cm)

TLP 916	RLP 916
1. DCL	1. GND
2. CHP	2. Digital Output
3. E	3. Linier Output
4. DTH	4. VCC
5. VCC	5. VCC
6. GND	6. GND
7. Antenna	7. GND
	8. Antenna (17 cm)

Tabel 1. Konfigurasi Pin TLP/RLP.

- Konfigurasikan Function Generator anda dengan output TTL. **(PERINGATAN: JANGAN MEMASUKKAN INPUT UNTUK MODUL TLP DENGAN SINYAL ANALOG KARENA MODUL DAPAT RUSAK).**

- Atur frekuensi pada function generator dan lihat bentuk sinyal output pada RLP. Apabila bentuk sinyal output yang keluar pada RLP ikut berubah maka kedua modul tersebut masih baik.

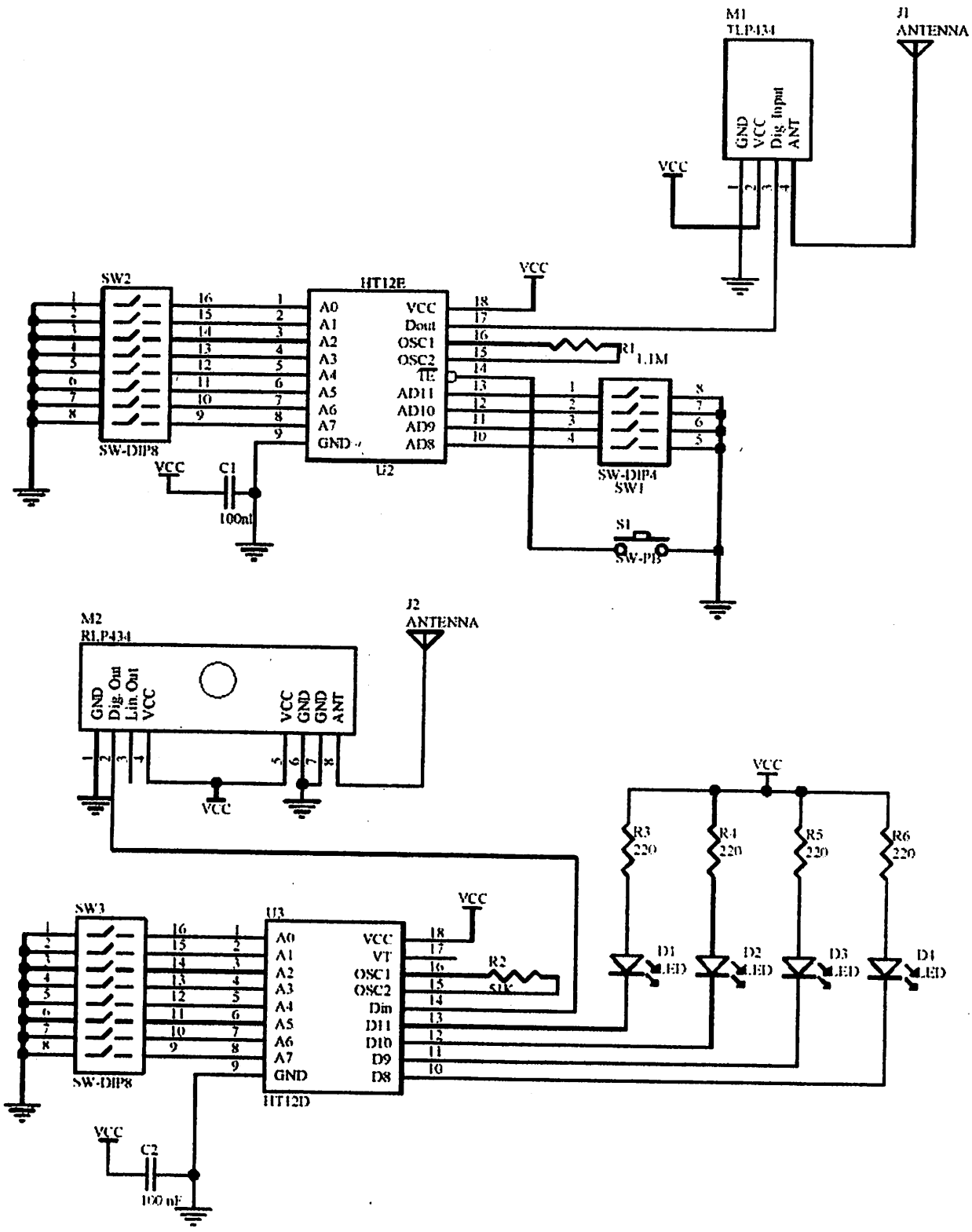
❖ Prosedur Pengujian menggunakan IC HT12E dan HT12D

Pengujian dilakukan dengan cara merangkai IC HT12E dan HT12D seperti pada gambar 3. Cara untuk melakukan pengujian adalah sebagai berikut:

1. Masukkan Alamat yang sama pada HT12E dan HT12D dengan cara menggeser kedua DIP Switch kearah ground. Contoh: apabila alamat HT12E adalah 0 maka HT12D juga 0.

2. Kirimkan data dari HT12E dengan cara menggeser SW1 menuju ground.

3. Lalu tekan push button S1 pada HT12E. Jika modul RLP dan TLP bekerja maka D1 – D4 akan menyala sesuai dengan penggeseran SW1 yang terdapat pada HT12E.



Gambar 3. Rangkaian HT12x pada modul RF.

Wireless Serial With RF Module

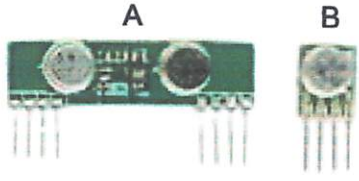
AN-04

Oleh: Tim Digiware dan Redi Yunianto K
(Universitas Katolik Widya Mandala)

Komunikasi data secara wireless (tanpa kabel) seringkali kita jumpai sehari – hari, terutama pada hal – hal yang berhubungan dengan komputer. Misalnya untuk wireless modem, mouse, keyboard, dll. Di dalam artikel ini akan dibahas mengenai cara penggunaan sebuah modul RF untuk komunikasi data secara wireless antara komputer dengan DT-51 MinSys. Modul RF yang digunakan adalah TLP-434A (Pemancar) dan RLP-434A (Penerima). Digunakannya TLP dan RLP 434A sebagai modul RF (Radio Frekuensi) pada artikel ini, selain kemampuannya di dalam pengiriman dan penerimaan data yang cukup baik, harganya tergolong relatif murah.

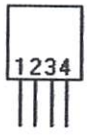
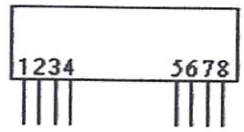
❖ Pengenalan TLP-434A dan RLP-434A

Modul RF buatan LAIPAC ini sering sekali digunakan sebagai alat untuk komunikasi data secara wireless menggunakan media udara. Biasanya kedua modul ini dihubungkan dengan mikrokontroler atau peralatan digital yang lainnya. Input masukan data adalah serial dengan level TTL (Transistor – Transistor Logic). Jarak pancar maksimum dari modul RF ini adalah 100 meter tanpa halangan dan 30 meter di dalam gedung. Ukuran ini dapat dipengaruhi oleh faktor antena, kebisingan, dan tegangan kerja dari pemancar. Panjang antena yang digunakan adalah 17 cm, dan terbuat dari kawat besi.



Bentuk fisik modul tampak depan.
RLP-434A (gb A) dan TLP-434 (gb B).

Untuk susunan kaki dari modul TLP dan RLP 434A dapat dilihat pada tabel di bawah:

 modul tampak dari depan	Pin 1: GND Pin 2: Data In Pin 3: VCC Pin 4: Antenna (RF Output)	 modul tampak dari depan	Pin 1: GND Pin 2: Digital Data Output Pin 3: Linear Output / Test Pin 4: VCC Pin 5: VCC Pin 6: GND Pin 7: GND Pin 8: Antenna (RF Output)

RANGE TESTING REPORT
by John Piskulic 6/25/00

PURPOSE:
To evaluate both range capability and noise immunity for several TX/RX pairs.

PROCEDURE:
All transmitters were powered from 12VDC and all receivers were powered from 5VDC. Three testing locations were chosen: field (open field with no structures with a 300' radius), shop1 (a commercial wood shop that had metal stud walls, tools, warehouse shelves), shop2 (home wood shop). Tests performed in shop2 were done with a minimum of 3 tools running to test for possible interference of AC motor generated RF noise. Range measurements are distances at which reception was on the border of being intermittent. Intermittent ranges could extend the apparent distance, however this was not considered for these tests. Shop 1 was limited to a [illegible] was still good the word "MAX" appears in the table. Shop 2 had a maximum testable distance of 60'.

Transmitter	Description	Antenna
TX1	LC based	loop
TX2	TLP-315	8.9" whip
TX3	TLP-315	4" PC trace

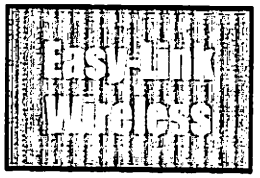
Receiver	Description	Antenna
RX1	Micrel MICRF001 eval brd	8.9" whip
RX2	RLP-315	8.9" whip
RX3	RLP-315	11' PC trace

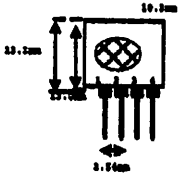
TRANSMITTER	RECEIVER	FIELD	SHOP1	SHOP2
TX1	RX1	144	125	max
TX2	"	330	max	max
TX3	"	138	130	max
TX1	RX2	405	max	max
TX2	"	500	max	max
TX2 w/ 5.6" whip	"	415	-	-
TX2 w/ 5" whip	"	405	-	-
TX2 w/ 3" whip	"	355	-	-
TX2 no whip	"	185	-	-
TX2 no whip	RX2 w/ 5.6" whip	159	-	-
TX3	RX2	350	max	max
TX1	RX3	320	max	max
TX2	RX3	340	max	max
TX3	RX3	300	max	max

CONCLUSION:
Product range goal is 100' indoors. All combinations passed this criteria. The Micrel MICRF001 required tuning in order to achieve best range for that receiver. The TLP and RLP modules did not seem to benefit from any tuning, but simply performed better with longer antennas.

RF ASK Hybrid Modules for Radio Control (New Version)

TLP434A Ultra Small Transmitter





pin 1 : GND
pin 2 : Data in
pin 3 : Vcc
pin 4 : Antenna (RF output)

Frequency 315, 418 and 433.92 Mhz

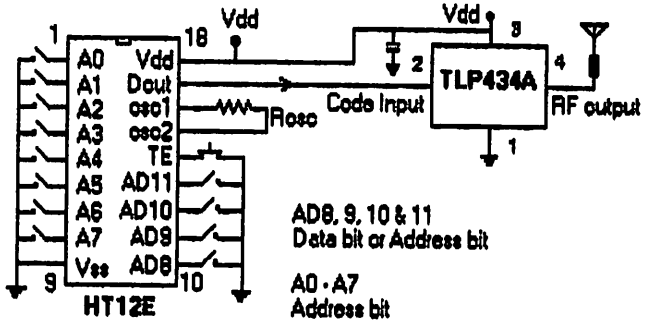
Modulation : ASK
Operation Voltage : 2 - 12 VDC

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
Vcc	Operating supply voltage		2.0	-	12.0	V
Icc 1	Peak Current (2V)		-	-	1.64	mA
Icc 2	Peak Current (12V)		-	-	19.4	mA
Vh	Input High Voltage	Idata= 100uA (High)	Vcc-0.5	Vcc	Vcc+0.5	V
Vl	Input Low Voltage	Idata= 0 uA (Low)	-	-	0.3	V
FO	Absolute Frequency	315Mhz module	314.8	315	315.2	MHz
PO	RF Output Power- 50ohm	Vcc = 9V-12V	-	16	-	dBm
		Vcc = 5V-6V	-	14	-	dBm
DR	Data Rate	External Encoding	512	4.8K	200K	bps

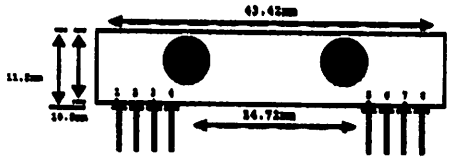
Notes : (Case Temperature = 25°C +- 2°C , Test Load Impedance = 50 ohm)

Application Circuit :

Typical Key-chain Transmitter using HT12E-18DIP, a Binary 12 bit Encoder from Holtek Semiconductor Inc.



RLP434A SAW Based Receiver



pin 1 : Gnd
pin 2 : Digital Data Output
pin 3 : Linear Output /Test
pin 4 : Vcc
pin 5 : Vcc
pin 6 : Gnd
pin 7 : Gnd
pin 8 : Antenna

Frequency 315, 418 and 433.92 Mhz

Modulation : ASK
Supply Voltage : 3.3 - 6.0 VDC
Output : Digital & Linear

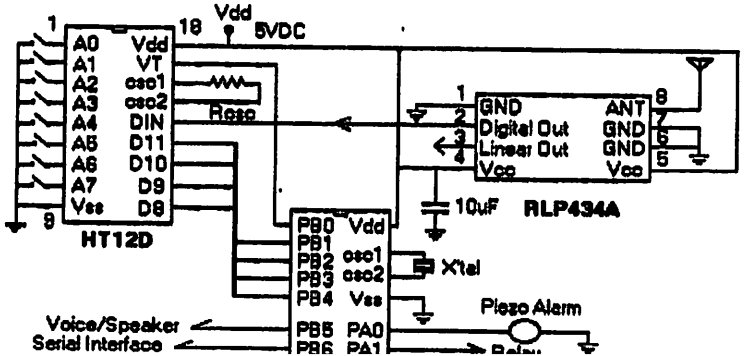
Symbol	Parameter	Conditions	Min	Typ	Max	
Vcc	Operating supply voltage		3.3	5.0V	6.0	V
Itot	Operating Current		-	4.5		mA
Vdata	Data Out	Idata = +200 uA (High)	Vcc-0.5	-	Vcc	V
		Idata = -10 uA (Low)	-	-	0.3	V

Electrical Characteristics

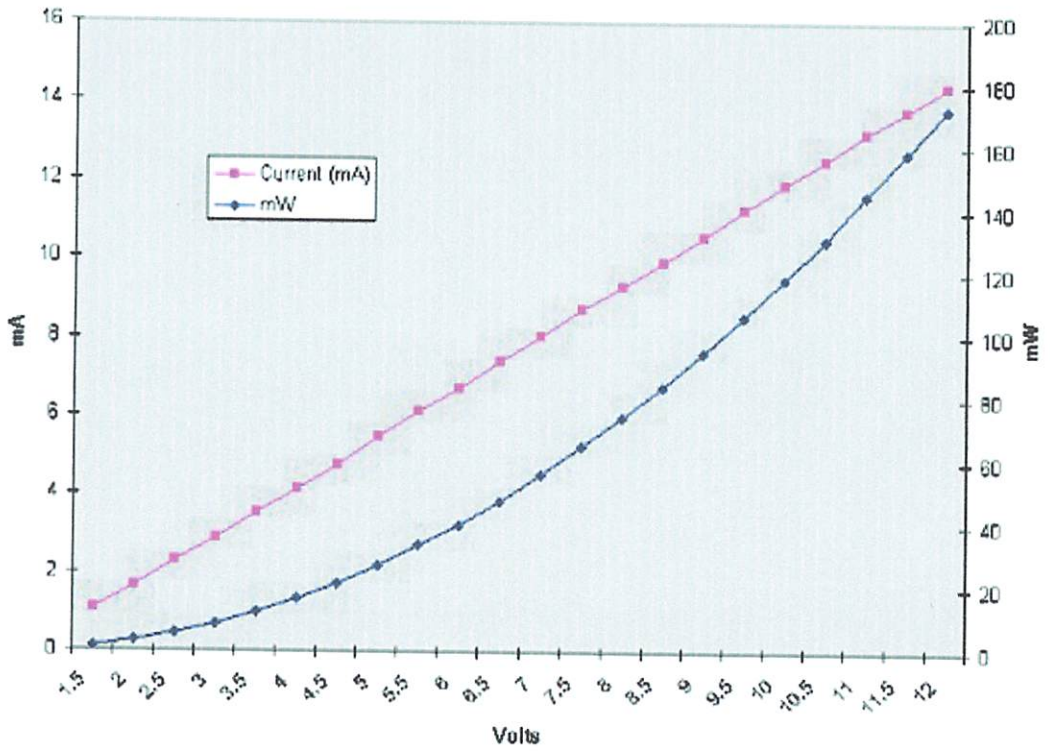
Characteristics	SYM	Min	Typ	Max	Unit
Operation Radio Frequency	FC		315, 418 and 433.92		MHz
Sensitivity	Pref		-110		dBm
Channel Width			+-500		Khz
Noise Equivalent BW			4		Khz
Receiver Turn On Time			5		ms
Operation Temperature	Top	-20	-	80	C
Baseboard Data Rate			4.8		KHz

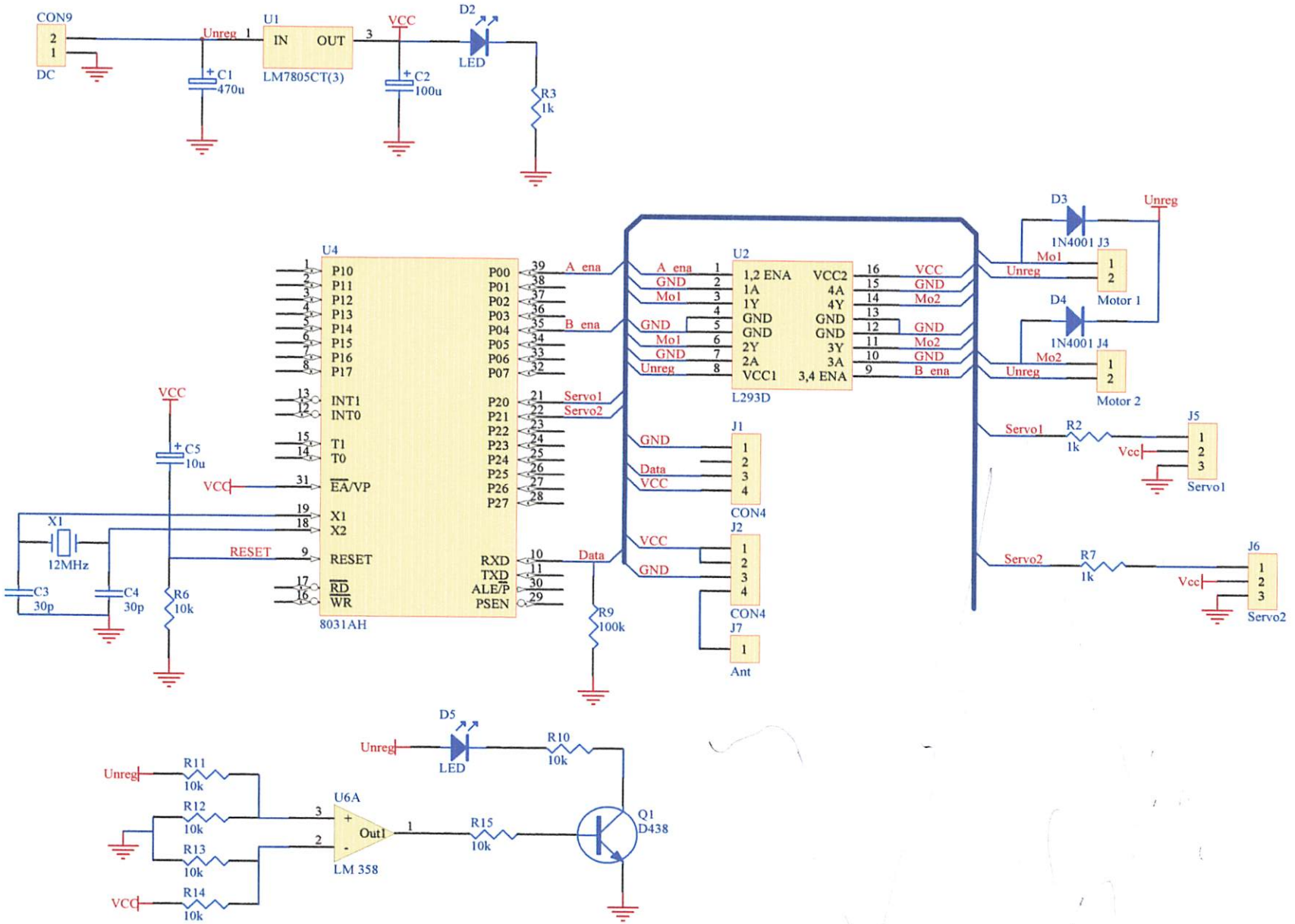
Application Circuit :

Typical RF Receiver using HT12D-18DIP, a Binary 12 bit Decoder with 8 bit uC HT48RXX from Holtek Semiconductor Inc.



Graph ?


Current (mA)
mW
mA
mW
Volts
0
2
4
6
8
10
12
14
16
0
20
40
60
80
100
120
140
160
180
200
1.5
2
2.5
3
3.5
4
4.5
5
5.5
6
6.5
7
7.5
8
8.5
9
9.5
10
10.5
11
11.5
12


CON9
DC
Unreg
C1
470u
U1
IN
OUT
LM7805CT(3)
VCC
C2
100u
D2
LED
R3
1k
U4
P10
P11
P12
P13
P14
P15
P16
P17
INT1
INT0
T1
T0
EA/VP
X1
X2
RESET
RD
WR
P00
P01
P02
P03
P04
P05
P06
P07
P20
P21
P22
P23
P24
P25
P26
P27
RXD
TXD
ALE/P
PSEN
8031AH
A_ena
B_ena
Servo1
Servo2
Data
C5
10u
12MHz
C3
30p
C4
30p
R6
10k
R9
100k
U2
1,2 ENA
1A
1Y
GND
GND
2Y
2A
VCC1
VCC2
4A
4Y
GND
GND
3Y
3A
3,4 ENA
L293D
Mo1
Mo2
GND
Unreg
VCC
B_ena
J1
CON4
J2
CON4
J7
Ant
Data
D3
1N4001
J3
Motor 1
D4
1N4001
J4
Motor 2
J5
Servo1
R2
1k
Vcc
J6
Servo2
R7
1k
D5
LED
R10
10k
R11
10k
R12
10k
R13
10k
R14
10k
U6A
Out1
LM 358
R15
10k
Q1
D438