SKRIPSI

PERENCANAAN DAN PEMBUATAN DATA LOGGER MENGGUNAKAN BLUETOOTH SEBAGAI SARANA PENGIRIMAN DATA

Disusun Oleh:

NAMA : DIKA YUDHA PURNAMA
NIM : 0412201

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

2015

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

LEMBAR PERSETUJUAN

PERENCANAAN DAN PEMBUATAN DATA LOGGER MENGGUNAKAN BLUETOOTH SEBAGAI SARANA PENGIRIMAN DATA

SKRIPSI

Disusun dan Diajukan Sebagai Salah Satu Syarat Untuk Memperoleh Gelar Sarjana Teknik Elektronika Strata Satu (S-1)

Disusun Oleh :

DIKA YUDHA PURNAMA

NIM : 04.12.201

Diperiksa dan Disetujui

Dosen Pembimbing I	Dosen Pembimbing II
Dr. Eng. I Komang Somawirata ST, MT	**M Ibrahim Ashari ST, MT**
NIP.Y 103 010 0361	**NIP.P 103 010 0358**

Mengetahui,

Ketua Jurusan Teknik Elektro S-1

M. Ibrahim Ashari ST, MT

NIP.P 103 010 0358

JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK ELEKTRONIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
2015

LEMBAR PERSETUJUAN

PERENCANAAN DAN PEMBUATAN DATA LOGGER MENGGUNAKAN BLUETOOTH SEBAGAI SARANA PENGIRIMAN DATA

SKRIPSI

Disusun dan Diajukan Sebagai Salah Satu Syarat Untuk Memperoleh Gelar Sarjana Teknik Elektronika Strata Satu (S-1)

Disusun Oleh :

DIKA YUDHA PURNAMA

NIM : 04.12.201

Diperiksa dan Disetujui

Dosen Pembimbing I	Dosen Pembimbing II
Dr. Eng. I Komang Somawirata ST, MT	**M Ibrahim Ashari ST, MT**
NIP.Y 103 010 0361	**NIP.P 103 010 0358**

Mengetahui,

Ketua Jurusan Teknik Elektro S-1

M Ibrahim Ashari ST, MT
NIP.P 103 010 0358

JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK ELEKTRONIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
2015

KATA PENGANTAR

Puji syukur kehadirat Allah SWT dan Nabi Muhammad SAW yang telah memberikan Rahmat serta HidayahNya sehingga saya dapat menyelesaikan skripsi ini dengan judul **"PERENCANAAN DAN PEMBUATAN DATA LOGGER MENGGUNAKAN BLUETOOTH SEBAGAI SARANA PENGIRIMAN DATA".**

Skripsi ini disusun sebagai syarat untuk mengikuti ujian akhir strata 1 program studi Teknik Elektronika S-1 di Institut Teknologi Nasional Malang.

Terselesaikannya skripsi ini tidak lepas dari bantuan banyak pihak. Untuk itu penulis menyampaikan ucapan terima kasih kepada:

1. Bapak Dr. Ir. Lalu Mulyadi, MT selaku Rektor Institut Teknologi Nasional Malang.
2. Bapak Ir. Anang Subardi, MT selaku Dekan Fakultas Teknologi Industri.
3. Bapak M. Ibrahim Ashari, ST, MT selaku Ketua Jurusan Teknik Elektro S-1 ITN Malang serta selaku Dosen Pembimbing II yang telah banyak memberikan pengarahan dan bimbinganya dalam menyusun skripsi ini.
4. Bapak Dr. Aryuanto, ST, MT selaku Dosen Wali.
5. Bapak Dr. Eng. I Komang Somawirata, ST, MT selaku Dosen Pembimbing II yang telah banyak memberikan pengarahan dan bimbingannya dalam menyusun skripsi ini.
6. Seluruh Staff dan Dosen pengajar di jurusan Teknik Elekstro S-1 Institut Teknologi Nasional Malang.
7. Orang Tua dan keluarga tercinta yang telah memberikan doa restu, kasih saying, semangat, dan dukungan sehingga skripsi ini dapat terselesaikan.

8. Teman-teman atas dukungan dan kerjasamanya selama ini.
9. Semua pihak yang telah membantu penyusunan skripsi ini.

Penulis menyadari bahwa laporan ini masih banyak yang perlu disempurnakan. Oleh sebab itu kritik dan saran yang membangun sangat diharapkan.

Akhir kata, penulis mohon maaf kepada semua pihak bilamana selama penyusunan skripsi ini penyusun membuat kesalahan secara tidak sengaja dan semoga skripsi ini dapat bermanfaat bagi kita semua.

Malang, Juli 2015

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

PERENCANAAN DAN PEMBUATAN DATA LOGGER MENGGUNAKAN BLUETOOTH SEBAGAI SARANA PENGIRIMAN DATA

DIka Yudha Purnama

Nim : 0412201

Program Studi Elektro S-1, Konsentrasi Teknik Elektronika
Fakultas Teknologi Industri
Institut Teknologi Nasional Malang
Jl. Raya Karanglo KM 2 Malang Telp (0341) 417634
Email : dikayudha.dk@gmail.com

Abstraksi

Pada saat ini pengiriman data masih menggunakan alat berupa kabel, hal tersebut menimbulkan pembengkakan dari segi biaya karena membutuhkan kabel dengan jumlah yang tidak sedikit, untuk itu perlu dibuat "Perencanaan dan Pembuatan Data Logger Menggunakan Bluetooth Sebagai Sarana Pengiriman Data".

Dalam skripsi ini memanfaatkan ADC 0808 sebagai converter dari analog ke digital, mikrokontroler AT89S52 sebagai pemroses data, Bluetooth sebagai sarana pengiriman data dan komputer sebagai display keluaran data dengan memakai program Delphi. Disini komputer dilengkapi dengan Bluetooth Dongle. Bluetooth Dongle akan difungsikan sebagai receiver dari komunikasi Bluetooth, untuk tranceivernya digunakan Bluetooth Serial Adapter RBT-001 yang terhubung dengan mikrokontroler AT89S52 yang dihubungkan langsung dengan ADC 0808 sebagai inputan dari data logger yang kemudian ditampilkan ke komputer sebagai outputanya.

Berdasarkan hasil pengujian didapatkan kesimpulan bahwa jangkauan komunikasi Bluetooth mencapai 13 meter pada kondisi tertutup (closed field), sedangkan pada kondisi terbuka (open field) mencapai 16 meter.

Kata Kunci : *ADC 0808, Mikrokontroler AT89S52, Bluetooth RBT001.*

BAB I

PENDAHULUAN

1.1. Latar belakang

Di masa kejayaan teknologi informasi seperti sekarang ini, perkembangan ilmu pengetahuan dan teknologi berjalan sangat cepat dan semakin canggih. Sejalan dengan itu manusia membutuhkan suatu alat pencatat data yang akurat maupun efisien.

Selain itu yang sedang marak dan digandrungi masyarakat sekarang ini adalah teknologi nirkabel atau *wireless*. Teknologi *wireless* merupakan teknologi baru yang sangat menguntungkan. Dengan teknologi ini kita bisa mengkoneksikan peralatan-peralatan kita tanpa menggunakan kabel, yang suatu saat mungkin membuat kita jengkel dengan kabel-kabel yang rumit dan akan merusak keindahan ruang kerja kita karena instalasi kabel-kabel yang kurang rapi.

Maka dari itu saya mencoba memberikan suatu alternatif yaitu dengan membuat suatu alat yang dapat mencatat suatu pengukuran secara akurat dan efisien, alat inilah yang disebut sebagai *data logger*, dikatakan efisien karena alat ini menggunakan *Bluetooth* sebagai media yang digunakan untuk keperluan pengiriman data.

1.2. Rumusan masalah

Mengacu pada permasalahan yang ada, maka dalam skripsi ini diutamakan pada hal-hal sebagai berikut:

1. Bagaimana membuat suatu perangkat yang menggunakan mikrokontroler AT89S52 agar dapat digunakan sebagai pemroses pengiriman dan penerimaan data dengan komputer.
2. Bagaimana menggunakan teknologi komunikasi data *wireless* yang berupa *Bluetooth* sebagai media transfer data antara komputer dengan alat pengukuran data.

1.3. Batasan masalah

Agar permasalahan yang dibahas tidak meluas, maka perlu adanya pembatasan masalah. Adapun batasan masalah meliputi::

1. Bagaimana membuat suatu perangkat yang menggunakan bluetooth agar dapat digunakan sebagai alat pengiriman data antara mikrokontroler AT89S52 dengan perangkat komputer .
2. Menggunakan Modul *Bluetooth Serial Adapter* yaitu dengan nama RBT-001.
3. Tidak membahas tentang *power supply.*
4. Hanya membahas alat yang ada pada blok diagram.

1.4. Tujuan

Tujuan yang ingin dicapai dari penulisan skripsi ini adalah membuat suatu perangkat yang memanfaatkan teknologi *Bluetooth* agar dapat memfungsikan perangkat komputer guna pencatat data dari suatu pengukuran tersebut..

1.5. Metodologi Penulisan

Metode yang digunakan dalam penulisan skripsi ini adalah sebagai berikut:

1. Studi Pustaka

   Memperoleh data dengan cara membaca dan mempelajari buku *literature* yang berhubungan dengan penyusunan skripsi ini.

2. Studi Analisa Alat

   Dimaksudkan untuk melakukan analisa pengujian alat yang telah dirancang. Apakah alat tersebut sesuai dengan fungsi kerja yang diharapkan.

3. Pengolahan Data

   Mengolah data dengan jalan membuat analisa dan menarik kesimpulan dari hasil pengujian yang ada.

1.6. Sistematika Penulisan

BAB I Pendahuluan

Dalam Bab ini dijelaskan hal-hal yang berhubungan dengan latar belakang, tujuan, perumusan masalah, ruang lingkup pembahasan serta metodologi penelitian yang digunakan dalam pembuatan skripsi ini.

BAB II Landasan Teori

Dalam Bab ini akan di jelaskan mengenai, ADC 0808, Mikrontroler AT89S52, dan *Bluetooth* RBT-001.

BAB III Perencanaan Sistem

Dalam Bab ini akan membahas mengenai perencanaan dan pembuatan tugas akhir ini yang meliputi seluruh sistem ini baik itu perangkat keras maupun perangkat lunak sistem.

BAB IV Pengujian Sistem

Dalam Bab ini membahas tentang pengujian dan hasil yang diperoleh dari sistem yang telah dibuat.

BAB V Kesimpulan dan Saran

Dalam Bab ini berisi kesimpulan-kesimpulan yang diperoleh dari perencanaan dan pembuatan tugas akhir ini serta saran-saran guna menyempurnakan dan mengembangkan sistem lebih lanjut.

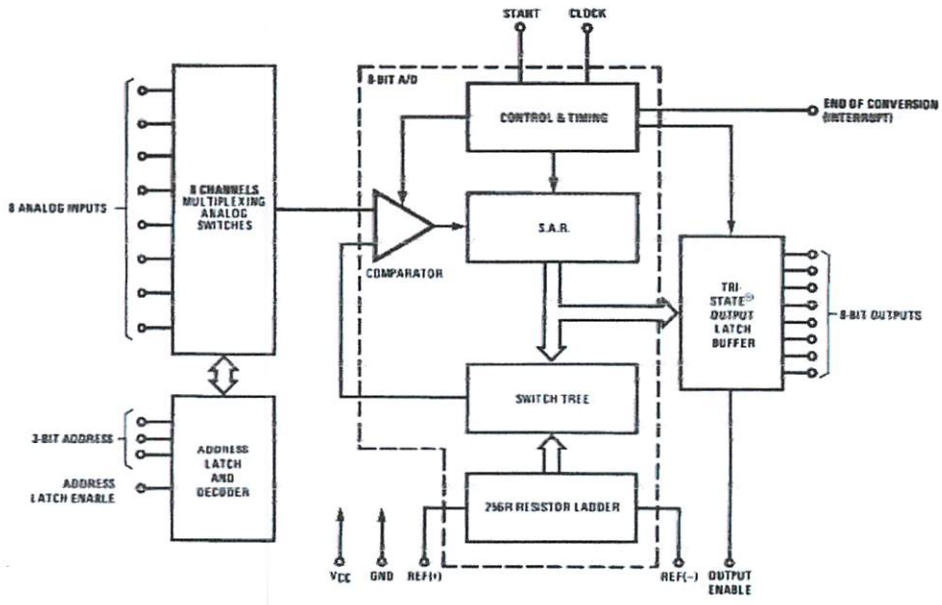
BAB II

LANDASAN TEORI

2.1. ADC 0808

ADC (*Analog to Digital Converter*) adalah sebuah piranti yang dirancang untuk mengubah sinyal-sinyal analog menjadi sinyal-sinyal digital yang nantinya akan masuk ke suatu koponen digital yaitu mikrokontroler.

ADC 0808 merupakan konverter A/D 8 bit yang dapat mengkonversi data input sebanyak 8 sinyal, sehingga dapat menerima 8 buah tranduser yang berbeda untuk setiap chipnya.



Gambar 2.1. Diagram Blok ADC 0808

(Sumber : *Datasheet* ADC 0808 *http://mekatronika.net*)

2.1.1. Arsitektur ADC 0808

ADC 0808 mempunyai spesifikasi sebagai berikut :

a. Mudah di *interface* kan dengan semua mikrokontroler maupun mikroprosesor.

b. Input berupa tegangan analog diferensial.

c. Tegangan referensi 5 Volt.

d. Tidak memerlukan *zero adjust*.

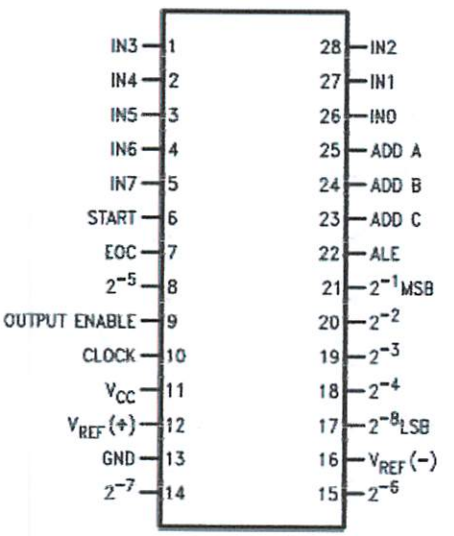
e. 8 *channel* multiplekser dengan pengalamatan logika..

f. Tegangan input 0 – 5 *Volt* dengan catu daya tunggal 5 *Volt*.

g. Resolusi 8 bit.

h. *Error* total ± ½ LSB.

2.1.2. Definisi Pin ADC 0808



Gambar 2.2. Skema Pin ADC 0808
(Sumber : *Datasheet* ADC 0808 *http://mekatronika.net*)

ADC 0808 dibangun dalam sebuah chip yang memiliki 28 pin. Penjelasan dari masing-masing pin tersebut adalah :

a. ***Power Pin*** : Terdiri dari pin 11 yang berfungsi sebagai *vcc* dan pin 13 yang berfungsi sebagai *Ground.*

b. ***Input/Output Pin*** : ADC 0808 memiliki output 8 bit dan memiliki 8 analog *input*. IN0 - IN7 merupakan inputan yang berupa sinyal-sinyal analog. Sedangkan $2^{-1} - 2^{-8}$ merupakan pin *output* yang akan keluar berupa sinyal-sinyal digital.

c. ***START*** : berfungsi untuk memulai proses konversi dari analog ke digital.

d. ***EOC*** : EOC (*End Of Convertion*) yang berfungsi sebagai pengontrol sinyal

e. ***ALE*** : ALE (*Address Lacth Enable*) merupakan pin output untuk melakukan proses *latching* terhadap pengaksesan *byte* rendah (*low byte*). *Latching* adalah proses pergantian fungsi antara pengiriman dan penerimaan alamat dan data.

f. ***Clock*** : merupakan inputan yang dapat ditambahkan sendiri pada ADC 0808 ini, yang berfungsi juga untuk menstabilkan outputan.

g. ***ADD A, B dan C*** : *Multiplexer address decoder* yang berfungsi untuk pemilihan *channel* ADC IN0 – IN7.

h. ***Vref***: Merupakan tegangan referensi.yang digunakan.

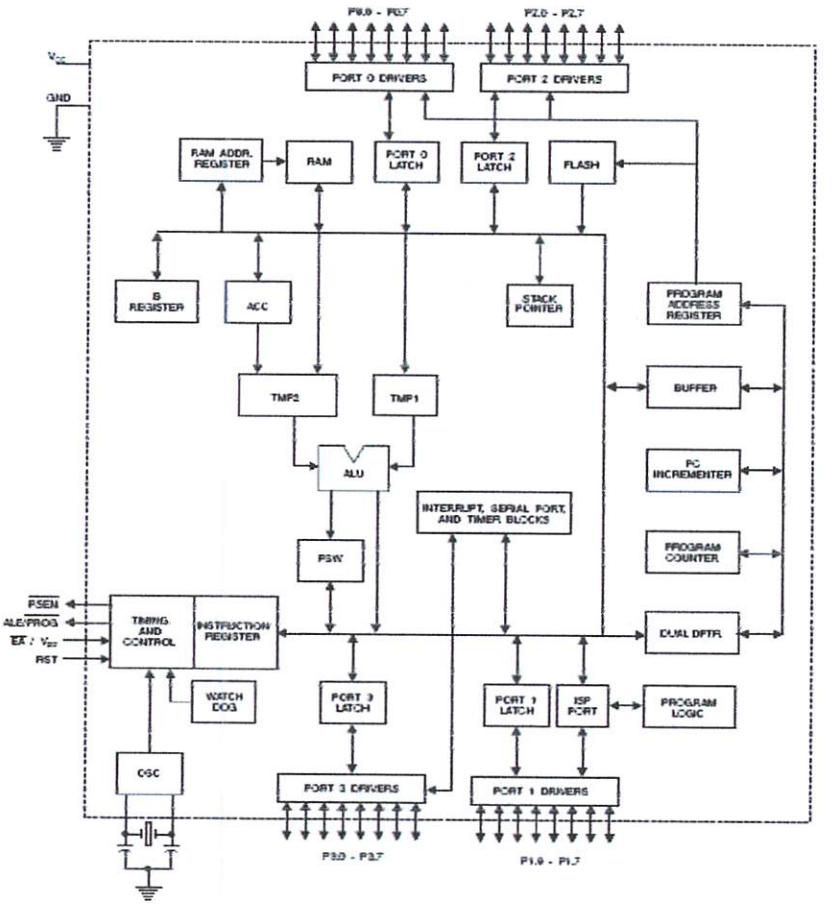
ADC 0808 yang mempunyai output 8 bit yaitu D0-D7 dan mempunyai 8 analog input.Pin A, B, C merupakan pin yang menetukan input pin berapa yang dipakai. sesuai dengan tabel berikut :

Tabel 2.1. Tabel kebenaran ADC 0808

SELECTED ANALOG CHANNEL	ADDRESS LINE		
	C	B	A
N0	L	L	L
NI	L	L	H
N2	L	H	L
N3	L	H	H
N4	H	L	L
N5	H	L	H
N6	H	H	L
N7	H	H	H

2.2. Mikrokontroler AT98S52

Mikrokontroller tipe Atmel AT89S52 termasuk kedalam keluarga MCS51 merupakan suatu mikrokomputer CMOS 8-bit dengan daya rendah, kemampuan tinggi, memiliki 8K *byte Flash Programable and Erasable Read* Only Memory (PEROM). Perangkat ini dibuat menggunakan teknologi memory nonvolatile (tidak kehilangan data bila kehilangan daya listrik). Set instruksi dan kaki keluaran AT89S52 sesuai dengan standar industri 80C51 dan 80C52. Atmel AT89S52 adalah mikrokomputer yang sangat bagus dan fleksibel dengan harga yang rendah untuk banyak aplikasi sistem kendali. Dibawah ini merupakan gambar diagram blok dari mikrokontroler AT89S52.



Gambar 2.3. Diagram Blok Mikrokontroler AT89S52
(Sumber : *Datasheet* AT89S52 *www.atmel.com*)

2.2.1. Arsitektur Mikrokontroler AT89S52

Mikrokontroler AT89S52 memiliki bagian-bagian sebagai berikut :

a. Memiliki sebuah CPU (*Central Processing Unit*) 8 bit.

b. Memiliki 256 *byte* RAM (*Random Access Memory*).

c. Memiliki empat buah port I/O, yang masing-masing terdiri dari 8 bit.

d. Memiliki osilator internal dan rangkaian pewaktu.

e. Memiliki 2 buah *timer/counter* 16 bit.

f. Memiliki 5 buah jalur interupsi (2 buah interupsi eksternal dan 3 buah interupsi internal).

g. Memiliki Sebuah port serial dengan *full duplex* UART (*Universal Asynchronous Receiver Transmitter*).

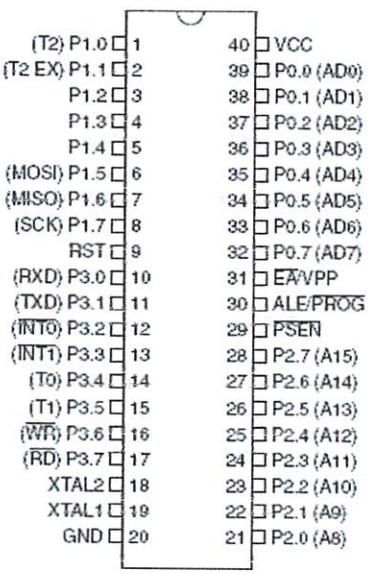
h. Memiliki *Full duplex serial port* sehingga memungkinkan pengiriman dan penerimaan data secara serial dapat berlangsung bersamaan.

i. Mampu melaksanakan proses perkalian, pembagian, dan Boolean.

j. Memiliki EPROM (*Eraseable Programmable Read Only Memory)* yang besarnya 8 *KByte* untuk memori program.

k. Kecepatan maksimum pelaksanaan instruksi per siklus adalah 0,5 µs pada frekuensi *clock* 24 MHz. Apabila frekuensi *clock* mikrokontroler yang digunakan adalah 12 MHz, maka kecepatan pelaksanaan instruksi adalah 1 µs.

2.2.2. Definisi Pin Mikrokontroler AT89S51



Gambar 2.4. Skema Pin Mikrokontroler AT89S52
(Sumber : *Datasheet* AT89S52 www.atmel.com)

Mikrokontroler AT89S8253 dibangun dalam sebuah *chip* yang memiliki 40 buah pin. Penjelasan dari masing-masing pin adalah sebagai berikut :

a. ***Power Pin*** : Terdiri dari pin 40 yang berfungsi sebagai *Vcc*, dan Pin 20 berfungsi sebagai hubungan *Ground.*

b. ***Input/Output Pin*** : Mikrokontroler AT89S52 memiliki 4 buah *port parallel* dan 1 buah *port serial.* Total dari pin *input/output* adalah sebanyak 32 jalur. *Port* 0 menempati 8 buah pin dari nomor pin 32 sampai dengan nomor pin 39, *Port* 1 menempati 8 buah pin dari nomor pin 1 sampai dengan nomor pin 8, *Port* 2 menempati 8 buah pin dari nomor pin 21 sampai dengan nomor pin 28, *Port* 3 menempati 8 buah pin dari nomor pin 10 sampai dengan nomor pin 17. Perlu diketahui juga bahwa beberapa pin memiliki fungsi khusus, termasuk pin-pin *serial port* yang terdapat pada *port* 3.

c. ***Reset pin*** : Merupakan input dari tombol *reset* pada mikrokontroler, untuk melakukan *reset* pada proses yang sedang dilakukan oleh mikrokontroler, maka perlu memberikan nilai 1 (*high*) pda pin ini selama 2 siklus mesin.

d. ***Memory addressing pin*** : Pin-pin ini berfungsi apabila mikrokontroler melakukan pengaksesan terhadap alamat memory eksternal. *Port* yang dilibatkan dalam hal ini adalah *port* 0 dan *port* 2, sedangkan pin-pin fungsional yang terlibat adalah pin WR (pin nomor 3.6) dan pin RD (pin nomor 3.7)

e. **ALE/PROG** : ALE (*Address Latch Enable*) adalah pin *output* untuk melakukan proses *latching* terhadap pengaksesan *byte* rendah (*low byte*). *Latching* adalah proses penggantian fungsi antara pengiriman dan penerimaan alamat dan data memori eksternal. Selain itu pin ini juga memegang peranan dalam proses pengisian program dalam *flash* PEROM. ALE hanya bereaksi dengan perintah MOVX dan MOVC saja. ALE dapat diaktifkan dan di non-aktifkan dengan mengakses bit pertama (bit 0) dari SFR (*special Function Register*). Pada saat aktif ALE akan mengeluarkan sinyal dengan frekuensi 1/6 dari frekuensi osilator mikrokontroler.

f. **PESN** : PESN (*Program Store Enable*) berfungsi balam pemberiam sinyal *strobe* apabila mikrokontroler melakukan pengaksesan memori program yang berada pada memori eksternal. PESN diaktifkan sebanyak 2 kali setiap siklus mesin (*machine cycle*) pada saat mikrokontroler mengakses program memori internal dan eksternal.

g. **EA/VPP** : EA (*External Access Enable*) harus dihubungkan ke *ground* apabila ingin melakukan akses ke program memori pada memori eksternal. Namun apabila program memori kita ada di dalam memori eksternal mikrokontroler, dan tidak diperlukan akses ke memori eksternal maka EA harus diberi nilai 1 (*high*) dengan cara dihubungkan ke Vcc. Pin EA berfungsi ganda sebagai penerima tegangan 12 Volt DC (Vpp) dalam proses pengisian program dalam flash PEROM.

h. ***CPU Clock Pin*** : Berfungsi dalam pewaktuan prosesor dari mikrokontroler.

i. **XTAL 1** : Masukan *clock* dari kaki osilator ke untai penguat pembalik (*inverting)* dan untai pewaktu (*clock*) internal mikrokontroler.

j. **XTAL 2** : keluaran dari untaian penguat pembalik internal menuju osilator kristal.

2.2.3. *Port Parallel* mikrokontroler AT89S52

Mikrokontroler tersebut mempunyai 40 kaki, 32 kaki di antaranya adalah kaki untuk keperluan *port parallel*. Tiap port paralel terdiri dari 8 kaki, dengan demikian 32 kaki tersebut membentuk 4 buah *port parallel*, yang masing–masing dikenal sebagai *Port* 0, *Port* 1, *Port* 2 dan *Port* 3. Nomor dari masing–masing jalur (kaki) dari port paralel mulai dari dari 0 sampai 7, jalur (kaki) pertama Port 0 disebut sebagai P0.0 dan jalur terakhir untuk Port 3 adalah P3.7.

Berikut merupakan penjelasan yang lebih mendalam mengenai struktur dan fungsi 4 buah *port parallel* pada mikrokontroler AT89S52 :

a. ***Port* 0** : *Port* 0 merupakan port keluaran/masukan (I/O) bertipe *open drain bidirectional* (dua arah). Sebagai *port* keluaran, masing–masing kaki dapat menyerap arus (*sink*) delapan masukan TTL (sekitar 3,8 mA). Pada saat '1' dituliskan ke kaki-kaki *Port* 0 ini, maka kaki-kaki *Port* 0 dapat digunakan sebagai masukan-masukan berimpedansi

tinggi. Jika *Port* 0 dapat dikonfigurasikan sebagai bus alamat/data bagian rendah (*low byte*) selama proses pengaksesan memori data dan program eksternal. Jika digunakan dalam mode ini *Port* 0 memiliki *pullup* internal. *Port* 0 juga menerima kode-kode yang dikirim kepadanya selama proses pemrograman dan mengeluarkan kode-kode selama proses varifikasi program yang telah tersimpan dalam flash. Dalam hal ini dibutuhkan pullup eksternal selama proses verifikasi program.

b. ***Port* 1** : *Port* 1 merupakan I/O dua arah yang dilengkapi dengan *pull-up* internal. Penyangga keluaran *Port* 1 mampu memberikan/menyerap arus empat masukan TTL (sekitar 1,6 mA). Jika '1'dituliskan ke kaki-kaki Port 1, maka masing–masing kaki akan di-*pulled high* dengan *pull-up* internal sehingga dapat digunakan sebagai masukan. Sebagai masukan, jika kaki-kaki *Port* 1 dihubungkan ke ground (di-*low pulled*), maka masing-masing kaki akan memberikan arus (*source*) karena di-*pulled high* secara internal. *Port* 1 juga menerima alamat bagian rendah (*low byte*) selama pemrograman dan verifikasi *flash*.

Tabel 2.2. Fungsi *port* 1

Port Pin	Alternate Functions
P1.0	T2 (external count input to Timer/Counter 2), clock-out
P1.1	T2EX (Timer/Counter 2 capture/reload trigger and direction control)
P1.5	MOSI (used for In-System Programming)
P1.6	MISO (used for In-System Programming)
P1.7	SCK (used for In-System Programming)

(Sumber : *Datasheet* AT89S52 www.atmel.com)

c. ***Port* 2** : *Port* 2 merupakan I/O dua arah yang dilengkapi dengan pull-up internal. Penyangga keluaran *port* 1 mampu memberikan/menyerap arus empat masukan TTL (sekitar 1,6 mA). Jika '1' dituliskan ke kaki-kaki *Port* 2, maka masing–masing kaki akan di-*pulled high* dengan pullup internal sehingga dapat digunakan sebagai masukan. Sebagai masukan, jika kaki-kaki Port 2 dihubungkan ke *ground* (di-*pulled low*), maka masing-masing kaki akan memberikan arus (*source*) karena di-pulled high secara internal. Port 2 akan memberikan *byte* alamat bagian tinggi (*high byte*) selama pengambilan instruksi dari memori program eksternal dan selama pengaksesan memori data eksternal yang menggunakan perintah dengan alamat 16-bit. Dalam aplikasi ini, jika ingin mengirimkan '1', maka digunakan pullup internal yang sudah disediakan. Selama pengaksesan memori data eksternal yang menggunakan perintah dengan alamat 8-bit, *Port* 2 akan mengirimkan isi dari SFR P2. Port 2 juga menerima alamat bagian tinggi selama pemrograman dan verifikasi flash.

d. ***Port* 3** : *Port* 3 merupakan port I/O dua arah dengan dilengkapi pullup internal. Penyangga keluaran *Port* 3 mampu memberikan/menyerap arus empat masukan TTL (sekitar 1,6 mA). Jika '1' dituliskan ke kaki-kaki port 3, maka masing-masing kaki akan di-*pulled high* dengan pullup internal sehingga dapat digunakan sebagai masukan. Sebagai masukan, jika kaki-kaki port 3 dihubungkan ke *ground* (di-*pulled low*), maka masing–masing kaki akan memberikan arus (*source*) karena di-

pulled high secara internal. *Port* 3, sebagaimana *Port* 1, memiliki fungsi-fungsi alternatif antara lain menerima sinyal-sinyal kontrol (P3.6 dan P3.7), bersama-sama dengan *port* 2 (P2.6 dan P2.7) selama pemrograman dan verifikasi flash.

Tabel 2.3. Fungsi *port* 3

Port Pin	Alternate Functions
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{\text{INT0}}$ (external interrupt 0)
P3.3	$\overline{\text{INT1}}$ (external interrupt 1)
P3.4	T0 (timer 0 external input)
P3.5	T1 (timer 1 external input)
P3.6	$\overline{\text{WR}}$ (external data memory write strobe)
P3.7	$\overline{\text{RD}}$ (external data memory read strobe)

(Sumber : *Datasheet* AT89S52 www.atmel.com)

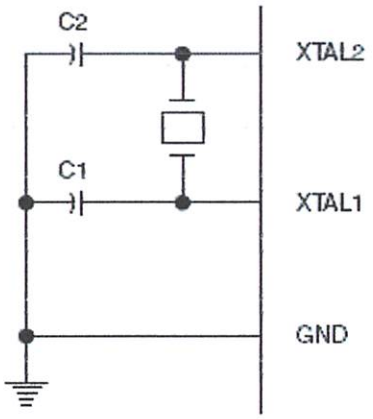
2.2.4. *Port Serial* Mikrokontroler AT89S52

Port serial pada mikrokontroler AT89S8253 dapat digunakan dalam mode *full-duplex*, artinya dapat menerima dan mengirim data secara serial bersamaan. Penerimaan dan pengiriman data *port serial* melalui sebuah *register* yang disebut SBUF (*Serial Data Buffer*). Dengan adanya SBUF, maka dimungkinkan juga untuk melakukan pembacaan atau pengiriman data pada pada lebih dari 1 *byte* data yang datang atau terkirim secara terpisah dan berurutan. *Port serial* dapat bekerja pada 4 mode yang berbeda, 1 mode diantaranya bersifat sinkron dan 3 mode lainnya bersifat asinkron. Perbedaan antara mode komunikasi sinkron dengan asinkron adalah bahwa pada pengiriman data sinkron, pengiriman data akan dikirim secara bersamaan dengan *clock*. Sedangkan pada data asinkron, tidak dikirim bersamaan dengan

clock. Rangkaian *receiver* akan harus membangkitkan sendiri *clock* yang diperlukan dalam pembacaan data *serial* tersebut.Keempat mode tersebut diatur dengan menggunakan SFR (*Special Function Register*) bernama SCON.

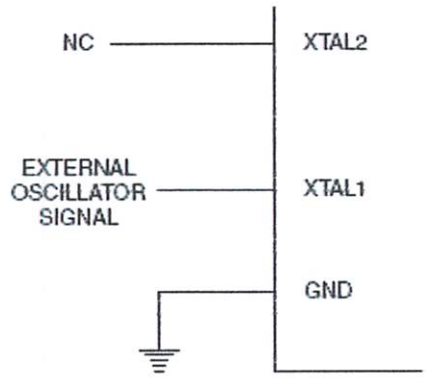
2.2.5. Sistem Pewaktuan (*Clock*) Mikrokontroler AT89S52

Clock merupakan sinyal yang digunakan oleh CPU (*Cental Processing Unit*) mikrokontroler untuk beroperasi. Setiap jenis mikrokontroler MCS-51 memiliki sebuah *on-chip oscillator*, namun untuk menggunakannya diperlukan sebuah resonator kristal (X-TAL) atau sebuah resonator keramik dan 2 buah kapasitor (C1 dan C2). Cara pemasangan resonator dan kapasitor dapat dilihat pada gambar berikut :



Gambar 2.5. Skema *Internal Clock* Mikrokontroler AT89S52
(Sumber : *Datasheet* AT89S52 www.atmel.com)

Selain dengan memanfaatkan osilator internal (*on-chip oscillator*) seperti diatas, dapat juga menggunakan *clock* eksternal. Cara pemakaiannya dapat dilihat pada gambar berikut:



Gambar 2.6. Skema *External Clock* Mikrokontroler AT89S52
(Sumber : *Datasheet* AT89S52 www.atmel.com)

2.2.6. Fasilitas *Reset* Mikrokontroler AT89S52

Fasilitas *reset* mikrokontroler berada pada pin nomor 9 pada skema *hardware* dan merupakan satu-satunya fasilitas untuk melakukan *reset* pada program mikrokontroler. Fungsinya menyerupai tombol *reset* pada komputer, yaitu menghentikan semua aktivitas ysng sedang dilakukan oleh CPU dan melakukan *restart* terhadap rangkaian kode program dari baris awal kembali. Cara melakukan *reset* ialah dengan memberikan sinyal 1 (*high*) terhadap pin 9 mikrokontroler selama 2 siklus mesin dan kemudian mengembalikannya ke kondisi semula (*low*).

Ada 2 macam sistem *reset* yang dapat dihubungkan ke pin 9 (RST) mikrokontroler. 2 sistem tersebut adalah *manual reset* dan *power-ON reset*. Perbedaan antara keduanya adalah pada *manual reset* kita dapat melakukan

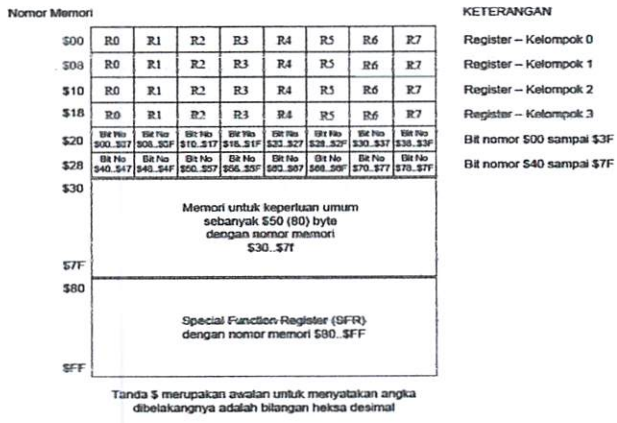
reset dengan menggunakan tombol atau saklar (S) yang tersendiri. Sedangkan apabila kita menggunakan mode *power-ON reset*, *reset* dilakukan dengan mematikan sementara suplai *power* (daya) ke mikrokontroler dan kemudian dihidupkan kembali.

2.2.7. Organisasi Memori

Terdapat 2 tipe memori pada mikrokontroler yaitu, RAM (*Random Access Memory*) dan ROM (*Read Only Memory*). Secara fungsi, pemisahan memori ini dapat digolongkan sebagai memori data (RAM) dan memori program (ROM). Adapula RAM eksternal yang bersifat *optional*, artinya bisa digunakan atau tidak digunakan.

2.2.7.1. Memori Internal

Memori internal (RAM) pada AT89S8253 sebanyak 256 byte dapat dipetakan sebagai berikut :



Gambar 2.7. Peta Memori RAM Internal Mikrokontroler AT89S52
(Sumber : *Datasheet* AT89S52 www.atmel.com)

Memori data dibagi menjadi 2 bagian, memori nomor $00 sampai $7F merupakan memori seperti RAM pada umumnya walaupun beberapa mempunyai kegunaan yang spesifik. Sedangkan memori nomor $80 sampai $FF digunakan untuk fungsi-fungsi khusus atau biasa disebut dengan SFR (*Special Function Register*). Memori data nomor $00 sampai $7F dapat digunakan sebagai memori penyimpan data biasa yang umumnya dapat dibagi menjadi 3 bagian, yaitu:

a. Memori nomor $00 sampai $18 dapat digunakan sebagai register serba guna (*General Purpose Register*).
b. Memori nomor $20 sampai $2F dapat digunakan untuk menyimpan informasi dalam level bit (*bit-addressable*). Setiap byte memori di daerah ini dapat digunakan untuk menampung 8 bit informasi yang masing-masing dinomori tersendiri. Dengan demikian dari 16 byte memori yang ada bisa dipakai untuk menyimpan 128 bit (16 x 8 bit).
c. Memori nomor $30 sampai $7F (sebanyak 80 byte) merupakan memori data biasa yang dapat digunakan untuk menyimpan data atau untuk *Stack*.

2.2.7.2. Memori Eksternal

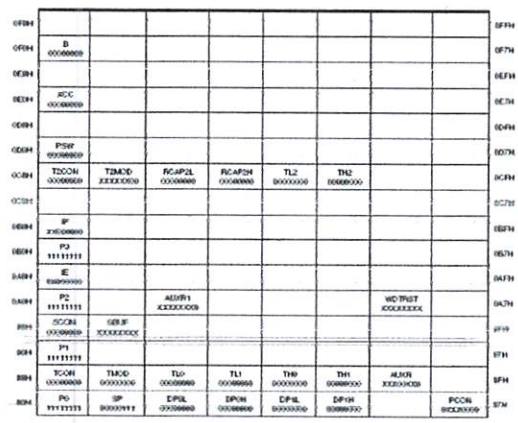
Memori eksternal bersifat optional atau bias dipakai bila diperlukan. Memori eksternal dapat diakses maksimal benilai kapasitas 64 Kb. Pengaksesan memori eksternal dilakukan secara paralel oleh port 0 dan port 2

dengan bantuan pin-pin lain seperti WR, RD, ALE, dan PSEN. Instruksi yang dipakai untuk mengakses ke memori eksternal adalah perintah MOVX.

2.2.8. *Special Function Register* (SFR)

Register-register internal dalam sebuah mikrokontroler pada umumnya dapat diakses dengan menggunakan set instruksi pemograman. Register internal mikrokontroler sebenarnya adalah bagian dari memori (RAM) mikrokontroler itu sendiri dan setiap register memiliki alamat tersendiri.

Special Function Register (SFR) adalah register internal yang terletak pada alamat memori (RAM) antara 80 H sampai FF H. Ada beberapa alamat (dari total 128 alamat) SFR yang tidak ada isinya. Meski letaknya pada RAM, namun ada beberapa alamat yang tidak dapat digunakan. Apabila dilakukan proses pembacaan maka akan dikembalikan nilai acak, sedangkan apabila ditulisi akan tidak tersimpan dengan benar pada alamat-alamat tersebut. AT89S8253 memiliki 33 buah SFR, seperti yang terlihat pada gambar berikut:



Gambar 2.8. Peta memori SFR Mikrokontroler AT89S52
(Sumber : *Datasheet* AT89S52 www.atmel.com)

2.2.9. Pengaturan *Baudrate Port* Serial

Baudrate ini merupakan *baud rate* pada komunikasi data yang digunakan pada port *serial* mikrokontroler AT89S52. *Baud rate* sebanding dengan frekuensi *clock* yang digunakan dalam pengiriman dan penerimaan data. Komunikasi asinkron tidak memerlukan sinyal *clock* sebagai sinkronisasi, namun pengiriman data ini harus diawali dengan *start bit* dan diakhiri dengan stop bit. Sinyal clock yang merupakan baud rate dari komunikasi data ini dibangkitkan oleh masing-masing baik penerima maupun pengirim data dengan frekuensi yang sama.

Satuan *baudrate* pada umumnya adalah bps (*bit per second*), yaitu jumlah bit yang dapat ditransmisikan per detik. *Baud rate* untuk mode 0 dan mode 2 bernilai tetap yaitu untuk mode 0 adalah 1/12 frekuensi osilator dan mode 2 adalah 1/64 frekuensi osilator. Dengan mengubah bit SMOD yang terletak pada register PCON menjadi set (kondisi awal saat sistem reset adalah clear), *baud rate* pada mode 1,2, dan 3 akan berubah menjadi dua kali lipat.

Baudrate untuk mode 0 nilainya tetap dan mengikuti persamaan berikut :

$$\textit{Baudrate}\ \text{Mode 0} = \frac{\textit{Frekuensi} - \textit{osilator}}{12}(\textit{bps})$$

dengan,

Baudrate Mode 0 = Frekuensi *clock* yang digunakan dalam pengiriman dan penerimaan data dengan mode 0 (bps) .

Frekuensi Kristal = Frekuensi sumber *clock* eksternal (MHz).

Satu hal yang harus diperhatikan dalam pengaturan *baud rate* adalah nilai *baud rate* dan nilai TH1 diusahakan harus tepat dan bukan merupakan pembulatan. Untuk komunikasi serial kecepatan tinggi, pembulatan terhadap nilai-nilai tersebut dapat mengakibatkan kekacauan dalam proses pengiriman atau penerimaan. Jika terdapat nilai pecahan, *user* disarankan untuk mengganti osilator dengan frekuensi yang sesuai. Untuk komunikasi dengan kecepatan rendah, toleransi terhadap kesalahan cukup besar sehingga pembulatan masih boleh dilakukan.

Tabel 2.4. *Baudrate* Komunikasi *Serial* Mikrokontroler

Baudrate (Bps)	Resonator Kristal (MHz)	Nilai Bit SMOD	Nilai Isi Ulang (reload) TH1
9600	12.000	1	-7 (F9 H)
2400	12.000	0	-13 (F3 H)
1200	12.000	0	-26 (E6 H)
19200	11.0592	1	-3 (FD H)
9600	11.0592	0	-3 (FD H)
2400	11.0592	0	-12 (F4 H)
1200	11.0592	0	-24 (E8 H)

(Sumber : *Datasheet* AT89S52 www.atmel.com)

2.3. Bluetooth

Nama Bluetooth berasal dari nama seorang raja yaitu Harald Blatand (ditranslasikan dalam bahasa Inggris sebagai Bluetooth), yang hidup pada pertengahan abad ke 10. Harald Blatand menyatukan dan mengendalikan Denmark dan Norwegia. Hal tersebut menjadikan inspirasi untuk menamakan peralatan yang terhubung secara bersama-sama menjadi Bluetooth.

Bluetooth merupakan *radio chip* yang dimasukkan ke dalam komputer, printer, handphone dan sebagainya. Chip Bluetooth ini dirancang untuk menggantikan kabel. Informasi yang biasanya dibawa oleh kabel

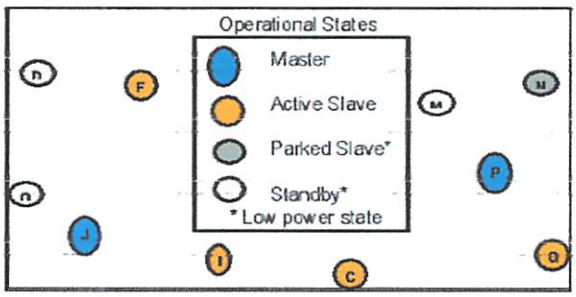
dengan bluetooth ditrasmisikan pada frekuensi tertentu kemudian diterima oleh chip bluetooth kemudian informasi tersebut diterima oleh komputer, handphone dan sebagainya.

Secara lebih rinci, Bluetooth merupakan nama yang diberikan untuk teknologi baru dengan menggunakan *short-range radio links* untuk menggantikan koneksi kabel *portable* atau alat elektronik yang sudah pasti. Tujuannya adalah mengurangi kompleksitas, *power* serta biaya.

Bluetooth diimplementasikan pada tempat-tempat yang tidak mendukung sistem *wireless* seperti di rumah atau di jalan untuk membentuk *Personal Area Networking* (PAN), yaitu peralatan yang digunakan secara bersama-sama.

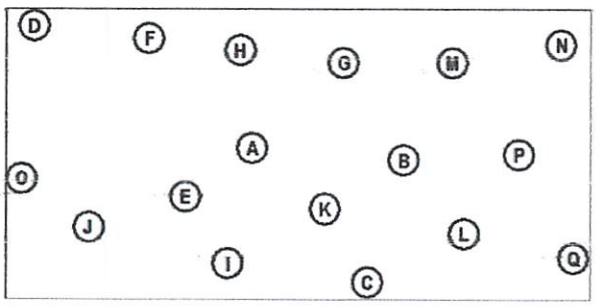
2.3.1. Cara Kerja Bluetooth

Pada Gambar 2.9 menunjukkan bagaimana *Bluetooth device* melakukan koneksi ke dalam *piconet. Piconet* terdiri dari sebuah *master device* dan *active slave devices*, dimana jumlah maksimum *active slaves* adalah 7. Kumpulan dari beberapa *piconet* yang saling berhubungan disebut dengan *scatternets.*



Gambar 2.9. *Operational State* dari Bluetooth
(Sumber : Studi dan Uji Coba Teknologi Bluetooth Sebagai Alternatif Komunikasi Data Nirkabel www.fportfolio.petra.ac.id)

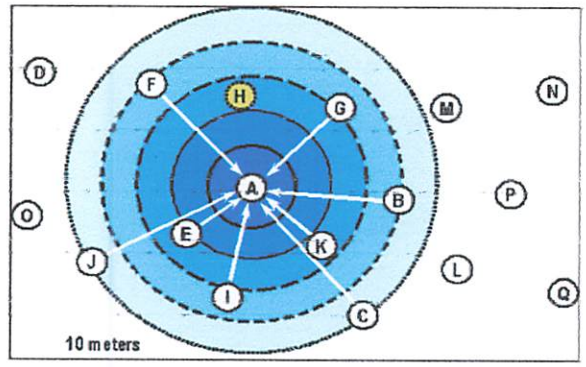
Bluetooth devices mempunyai 4 *basic states*. Antara lain adalah *master* (yang mengontrol sebuah *piconet*), *active slave* (terhubung dan secara aktif memonitor *Piconet*), *parked slave* (secara logika masih bagian dari *piconet* tetapi *low power*) dan *standby* (tidak terhubung dengan *piconet*).



Gambar 2.10. Bluetooth pada Awalnya
(Sumber : Studi dan Uji Coba Teknologi Bluetooth Sebagai Alternatif Komunikasi Data Nirkabel www.fportfolio.petra.ac.id)

Bluetooth device pada awalnya mengetahui hanya sekitar diri mereka dan di dalam status ini mereka akan berada di *mode Standby*. *Standby* adalah suatu mode pasif di mana suatu *Bluetooth device* sekali – kali mendengarkan jika ada *Bluetooth device* lain yang ingin berkomunikasi, hal ini disebut *Inquiry*. Proses ini dilakukan selama 10 miliseconds tiap 1.28 detik. Di dalam

mode *Standby*, *Bluetooth device* dapat mengurangi konsumsi kekuatannya sebesar 98%.

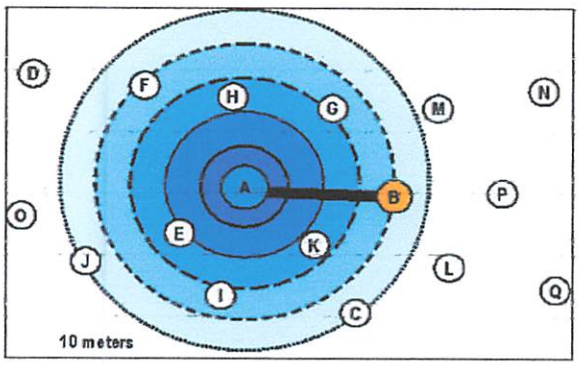


Gambar 2.11. Proses *Inquiry* Bluetooth
(Sumber : Studi dan Uji Coba Teknologi Bluetooth Sebagai Alternatif Komunikasi Data Nirkabel www.fportfolio.petra.ac.id)

Inquiry adalah suatu proses bagaimana *bluetooth device* belajar tentang *bluetooth devices* lain yang ada di dalam jangkauannya. Pada Gambar 2.10, *node* A mengeluarkan fungsi *page* pada BT *inquiry ID* dan menerima balasan dari *devices* B, C, E, F, G, I, J, and K. Dari balasan ini, A mengetahui identitas dari *devices* lain (contohnya, *bluetooth device ID* mereka yang unik).

Selama proses *inquiry*, *device* A secara terus-menerus melakukan *broadcasts page command* dengan menggunakan *reserved inquiry ID*. *Broadcasts* ini tersebar sepanjang pola standard dari 32 *standby radio frequencies* dimana semua *devices* pada mode *standby* memonitor pada sebuah *occasional basis*. Kemudian setiap *standby device* dalam jangkauannya akan menerima *inquiry page*. Dengan melakukan persetujuan, *node-node* ini akan merespon dengan sebuah standar *FHS packet* yang menyediakan bluetooth ID-nya yang unik *dan clock offset*nya. Node H pada

Gambar 2.11 menunjukkan bagaimana sebuah *bluetooth device* dapat diprogram sebagai anonymous (*Undiscoverable*).

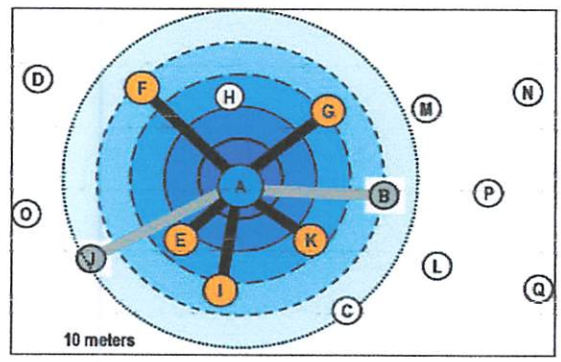


Gambar 2.12. Proses *Paging* Bluetooth
(Sumber : Studi dan Uji Coba Teknologi Bluetooth Sebagai Alternatif Komunikasi Data Nirkabel www.fportfolio.petra.ac.id)

Setelah proses *inquiry*, akan dilakukan proses *paging*, dimana pada proses ini akan dibangun hubungan antar *device* (antar *master* sebagai pemula dengan sebuah *slave*). Hubungan *master/slave* pada bluetooth dikenal dengan sebutan *piconet*. Untuk menciptakan *piconet*, *device A* melakukan *broadcasts Page command* dengan *explicit device ID* dari *slave target* (pada gambar di atas adalah B) yang telah siap. Semua *Bluetooth devices* kecuali B akan mengabaikan perintah ini karena tidak ditujukan pada mereka.

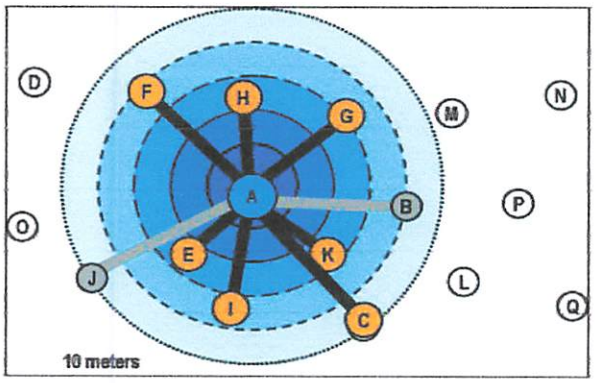
Ketika *device* B membalas, *device* A akan mengirim sebuah *FHS packet* kembali dan menetapkannya sebagai *Active Member Address* pada *Piconet*. Sebagai *active slave*, *device* B mulai memonitor secara terus – menerus perintah selanjutnya dari *device* A. Sebuah *Bluetooth master* dapat melakukan proses *paging* ini dengan maksimum 7 *active slaves*. 7 merupakan batas atas karena hanya disediakan 3 bits pada Bluetooth untuk *Active*

Member Address (AMA) dengan 000 disediakan untuk *master* dan sisanya untuk *slaves*. Sekali lagi, semua *active slaves* ke A akan memonitor secara terus-menerus untuk perintah yang ditujukan kepada mereka dalam sinkronisasi dengan *device A's hopping pattern*.



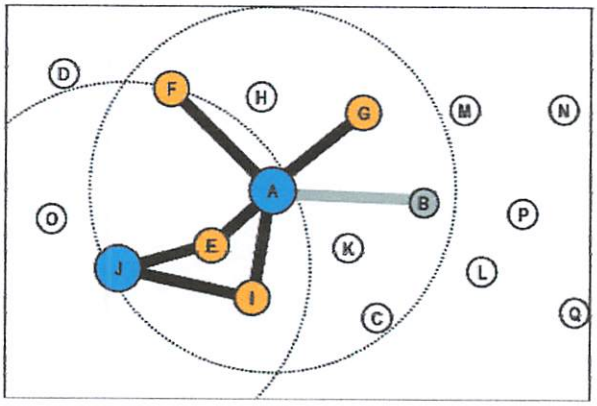
Gambar 2.13. Proses *Parking* Bluetooth
(Sumber : Studi dan Uji Coba Teknologi Bluetooth Sebagai Alternatif Komunikasi Data Nirkabel www.fportfolio.petra.ac.id)

Parking merupakan mekanisme yang mengijinkan *Bluetooth Master* untuk berhubungan dengan 256 *devices* tambahan. 256 adalah batas atas karena disediakan 8 bits pada Bluetooth untuk *Parked Member Address* (PMA). Untuk memarkirkan sebuah *device*, *Bluetooth Master* mengeluarkan *Park command* terhadap sebuah *active slave* dan menetapkannya sebagai PMA. *Slave* ini kemudian memasuki mode *parked* dan menyerahkan AMA-nya. Sebagai sebuah *parked slave*, *device* akan berubah ke dalam mode *passive* dan hanya memonitor perintah-perintah pada *occasional basis*.



Gambar 2.14. Proses Mengembangkan *Piconet*
(Sumber : Studi dan Uji Coba Teknologi Bluetooth Sebagai Alternatif Komunikasi Data Nirkabel www.fportfolio.petra.ac.id)

Dengan adanya *Active Member Addresses* yang dilepaskan oleh sebuah *active slaves*, *Bluetooth Master* dapat melakukan proses *paging* dengan *device* lain untuk menjadi *Active Slaves*. Pada Gambar 2.14, *device* A menambahkan H dan C ke dalam *piconet*-nya dengan AMA yang dilepaskan oleh *parking nodes* B dan J.

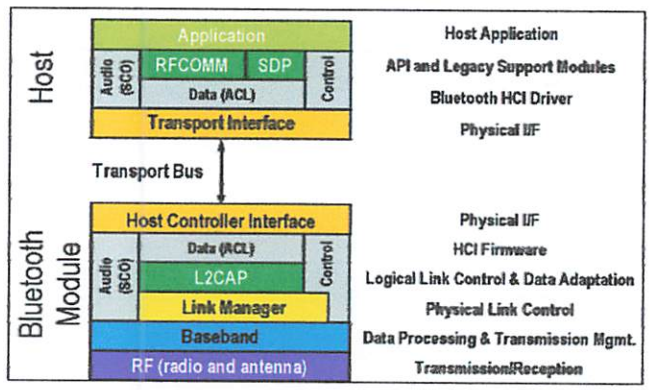


Gambar 2.15. Struktur *Scatternet*
(Sumber : Studi dan Uji Coba Teknologi Bluetooth Sebagai Alternatif Komunikasi Data Nirkabel www.fportfolio.petra.ac.id)

Tiap *Bluetooth node* dapat menjadi bagian dari beberapa *piconet* sekaligus dalam satu waktu. Hal ini membuat beberapa *piconet* dapat bergabung membentuk sebuah struktur yang disebut *scatternet*. Pada Gambar 2.15, dua *piconet* bergabung menjadi sebuah *scatternet* melalui *slaves* E dan I.

2.3.2. *Bluetooth Protocol Stack*

Tiap sistem Bluetooth terdiri dari sebuah aplikasi berbasis *host* dan sebuah *Bluetooth module*. *Host* dapat berupa apapun, dari sebuah *standalone computer* sampai dengan sebuah *embedded controller* seperti pada sebuah *cell phone*.

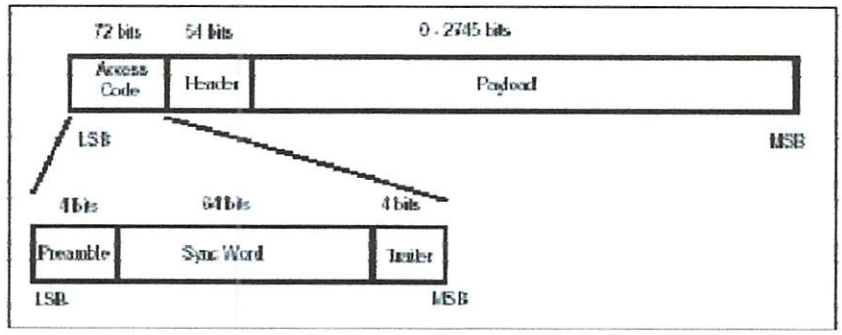


Gambar 2.16. *Bluetooth Protocol Stack*
(Sumber : Studi dan Uji Coba Teknologi Bluetooth Sebagai Alternatif Komunikasi Data Nirkabel www.fportfolio.petra.ac.id)

Gambar 2.16. menunjukkan bagaimana tugas – tugas dibagi dari mulai *host* sampai ke RF dan sebaliknya. Tiap *layer* melakukan fungsi yang spesifik, sama seperti pada sebuah *Ethernet stack*. Arsitektur ini akan membuat desain sistem menjadi lebih mudah dan membuat banyak muncul implementasi.

2.3.3. Struktur *Frame* Data Bluetooth

Struktur *frame* data dari Bluetooth dapat dilihat pada Gambar 2.17 berikut:



Gambar 2.17. Struktur *Frame* Data Bluetooth
(Sumber : Studi dan Uji Coba Teknologi Bluetooth Sebagai Alternatif Komunikasi Data Nirkabel www.fportfolio.petra.ac.id)

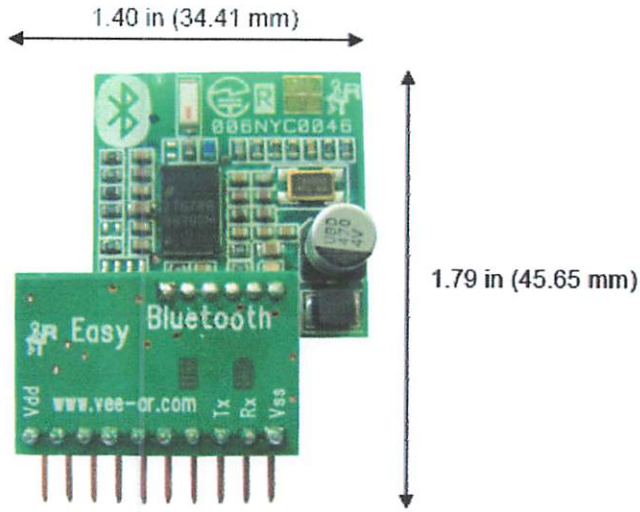
Struktur *frame* data Bluetooth terdiri dari beberapa bagian, yaitu:

1. *Channel Access Code* (CAC): mengidentifikasikan sebuah *piconet*, kode ini digunakan dengan semua *traffic exchanged* pada sebuah *piconet.*
2. *Device Access Code* (DAC): Digunakan untuk *signaling*, seperti *paging* dan *respons* terhadap *paging.*
3. *Inquiry Access Code* (IAC), *Inquiry process "finds" bluetooth device* dalam *range*, *Inquiry Access Code* (IAC) terbagi menjadi 2 yaitu:
   a. *General Inquiry Access Code* (GIAC), umum untuk semua *Bluetooth devices.*
   b. *Dedicated Inquiry Access Code* (DIAC), umum untuk sebuah kelas dari *Bluetooth devices.*

4. *Packet Header*, terdiri dari 2 bagian yaitu:
    a. AM_ADDR: 3 bit alamat *member* menunjukkan *active members* dari sebuah *piconet.*
    b. *Data Type*: Menunjukkan bermacam-macam tipe paket dan panjangnya. Memperbolehkan *non-addressed slaves* untuk menentukan kapan mereka dapat *transmit.*
5. *Flow Control*, terdiri dari 2 bagian yaitu:
    a. *Acknowledgement*: ACK/NAK *field.*
    b. HEC: *header error check*, jika *error* ditemukan, keseluruhan paket dibuang.

2.4. Bluetooth RBT-001

RBT-001 merupakan suatu perangkat *Serial Bluetooth* yang dapat berkomunikasi dengan perangkat Bluetooth lainnya seperti telepon selular, komputer, dan PDA dengan memanfaatkan gelombang radio Bluetooth sebagai media transfer datanya. RBT-001 dirancang untuk penggunaan koneksi secara *point to point* seperti halnya koneksi dengan menggunakan kabel biasa. Koneksi yang dapat digunakan adalah koneksi antara dua buah perangkat RBT-001 maupun koneksi antara perangkat RBT-001 dengan perangkat Bluetooth lainnya. Dibawah ini merupakan bentuk fisik dari Bluetooth RBT-001:



Gambar 2.18. Bentuk Fisik dari RBT-001
(Sumber : *Datasheet* RBT-001 Parallax)

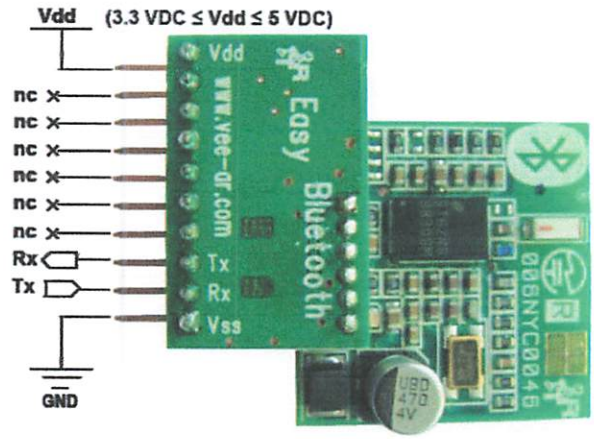
2.4.1. Karakteristik RBT-001

Komunikasi Bluetooth yang digunakan RBT-001 ini memiliki karakteristik sebagai berikut:

a. Menggunakan teknologi FHSS (*Frequency Hopping Spread Spectrum*).
b. Bekerja pada kisaran frekuensi 2.4 Ghz.
c. Menggunakan tipe komunikasi serial dengan jangkauan komunikasi pada lapangan terbuka dapat lebih dari 100 meter (328 kaki).
d. RBT-001 dapat bekerja dengan baik pada *temperature* hingga 45° C.
e. *Supply Power* sebesar 3,3 hingga 5,5 Vdc.
f. *Baudrate* verkisar antara 9,6 Kbps – 230,4 Kbps

2.4.2. Deskripsi Pin RBT-001

Dibawah ini merupakan gambaran koneksi diagram dari Bluetooth RBT-001.



Gambar 2.19. Koneksi Diagram dari RBT-001
(Sumber : *Datasheet* RBT-001 Parallax)

Penjelasan tiap-tiap konektor RBT-001 dari gambar diatas dapat dilihat melalui tabel 2.5 dibawah ini.

Tabel 2.5. Konfigurasi Pin RBT-001

Pin	Name	Function
1	Vdd	+3.3 to +5 VDC (+5.5 max voltage)
2	RX	Receive Serial Communication (CMOS & TTL Level Compatible)
3	TX	Transmit Serial Communication (CMOS & TTL Level Compatible)
4	NC	Not Connected
5	NC	Not Connected
6	NC	Not Connected
7	NC	Not Connected
8	NC	Not Connected
9	NC	Not Connected
10	Vss	Ground

(Sumber : *Datasheet* RBT-001 Parallax)

Pin yang dipakai adalah Vdd, RX, TX, dan Vss, pin lainnya tidak digunakan karena format data serial yang dipakai tidak menggunakan sistem *flow control* dan untuk perubahan mode pada RBT-001 dapat dilakukan secara *software*.

2.4.3. Sistem Komunikasi RBT-001

Sistem komunikasi yang digunakan antar RBT-001 dengan mikrokontroler adalah secara *Serial* UART level TTL (standar pabrik = 9600 *Baudrate*, 8 *Data Bits*, 1 *Stop Bits*, *No Parity*, dan *No Flow Control*) dimana *baudrate* dan konfigurasi *flow control*nya dapat dimodifikasi sesuai keinginan pemrogram. RBT-001 dapat digunakan dengan nilai baudrate berkisar antara 9,6 Kbps sampai dengan 230,4 Kbps.

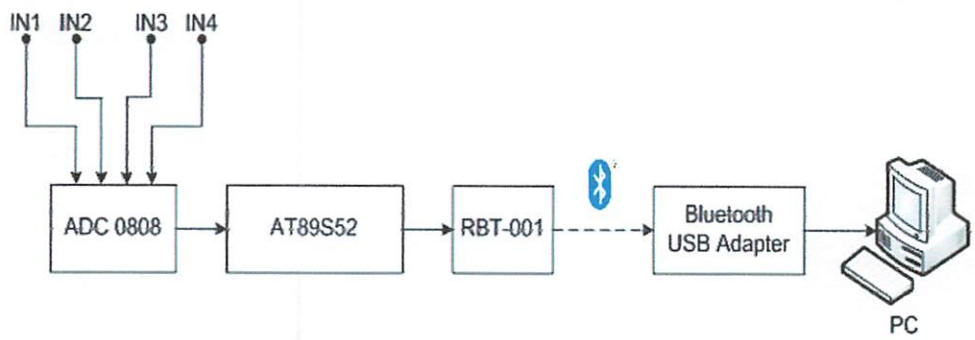
BAB III

PERANCANGAN DAN PEMBUATAN ALAT

Dalam bab ini akan membahas tentang perencanaan dan pembuatan keseluruhan sistem perangkat keras (*hardware*) dan perangkat lunak (*software*) yang digunakan dalam suatu perencanaan dan pembuatan data logger menggunakan Bluetooth sebagai sarana pengiriman data. Pembahasan dilakukan pada setiap blok rangkaian yang terdiri atas: cara kerja masing-masing blok rangkaian serta fungsi masing-masing blok rangkaian.

3.1. Blok Diagram Keseluruhan Sistem

Perancangan dan pembuatan alat untuk skripsi ini ditunjukkan dengan gambar blok diagram dibawah ini :



Gambar 3.1. Diagram Blok Keseluruhan Sistem

Fungsi dari masing-masing blok diagram diatas dapat dilihat dari penjelasan berikut ini :

a. **IN1, IN2, IN3, dan IN4** berfungsi sebagai inputan data yang akan dikirimkan ke PC, inputan dapat berupa sensor suhu, sensor putaran, dan lain sebagainya yang dapat diinputkan ke ADC 0808.
b. **ADC 0808** digunakan untuk mengubah sinyal analog menjadi suatu sinyal digital dari suatu pengukuran agar dapat dibaca oleh mikrokontroler.
c. **Mikrokontroler AT89S52** berfungsi sebagai pengatur seluruh sistem pengiriman maupun penerimaan data antara komputer dengan ADC 0808. Selain itu mikrokontroler ini juga berfungsi menyimpan data inputan sementara sebelum dikirimkan ke komputer.
d. **Bluetooth RBT-001** berfungsi sebagai jalur penerimaan maupun pengiriman data dari mikrokontroler ke komputer dan sebaliknya.
e. **Bluetooth USB Adapter (USB Bluetooth Dongle)** berfungsi sebagai jalur penerimaan data dari mikrokontroler ke komputer. Data yang diterima merupakan data yang dikirim RBT-001 yang kemudian akan diterima computer melalui Bluetooth USB Adapter.
f. **PC (komputer)** berfungsi sebagai output, yaitu menampilkan data dari pengukuran tersebut. Selain itu juga berfungsi sebagai pengontrol keseluruhan sistem.

3.2. Prinsip Kerja Alat

Inputan pada data *logger* ini disimulasikan menggunakan potensiometer. Dari potensiometer data masih berbentuk analog yang kemudian diubah ke digital oleh ADC 0808, stelah itu data dikirim ke mikrokontroler AT89S52, data tersebut dapat disimpan sementara di mikrokontroler yang kemudian di *download* oleh komputer ataupun dapat dikirim ke komputer secara langsung. Bluetooth RBT-001 sebagai media untuk mengirim data antar mikrokontroler AT89S52 dengan komputer. Untuk mengirim data dari mikrokontroler ke komputer pertama-tama data dari mikrokontroler dikirim ke RBT-001, kemudian selanjutnya dikirim ke *Bluetooth USB Adapter* yang sudah terpasang sebelumnya di komputer, *software* antarmuka yang digunakan adalah Delphi. Komputer disini digunakan sebagai outputan dari sistem ini yaitu berupa *database* yang dibuat dengan *Microsoft access* yang dikoneksikan ke Delphi.

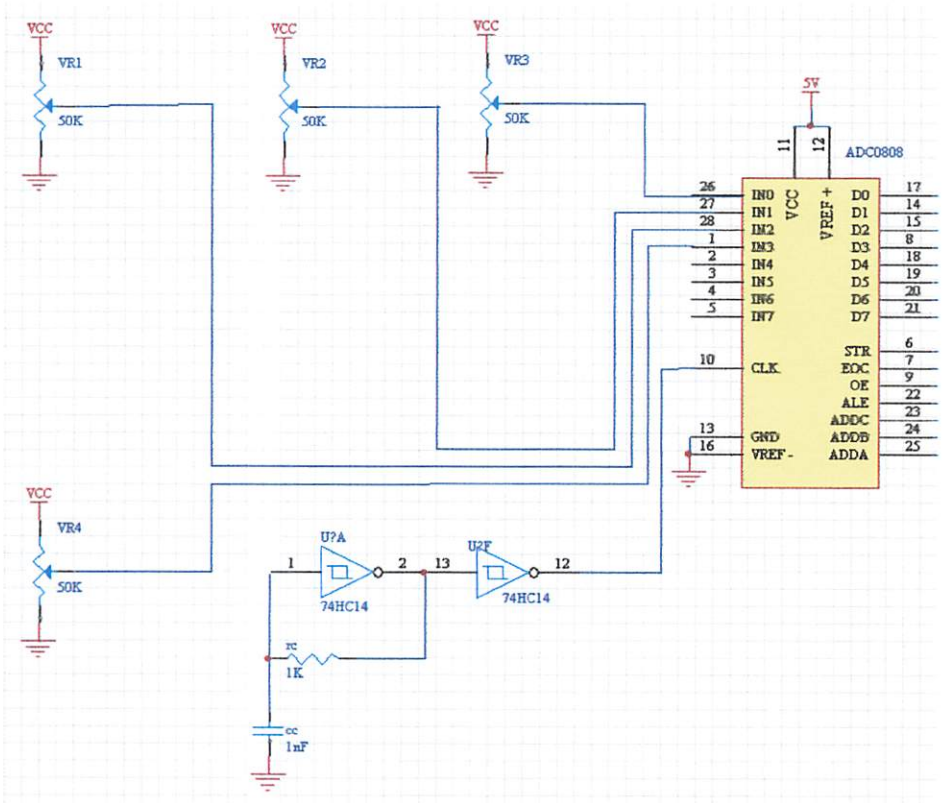
3.3. Perancangan Perangkat Keras (*Hardware*)

Berikut ini akan dibahas tentang perancangan perangkat keras (*hardware*) yang terdapat dalam sistem ini, yaitu antara lain ADC 0808, mikrokontroler AT89S52, dan Bluetooth RBT-001.

3.3.1. ADC 0808

Disini ADC berfungsi sebagai inputan dari mikrokontroler tetapi agar dapat terbaca oleh mikrokontroler data yang sebelumnya berupa data analog diubah oleh ADC menjadi digital.

Berikut ini merupakan gambar rangkaian dari ADC beserta inputanya :



Gambar 3.2. Rangkaian ADC 0808 Beserta Inputannya

a. Pin 1, 26, 27, dan 28

Digunakan sebagai inputan data yang akan ditampilkan pada PC

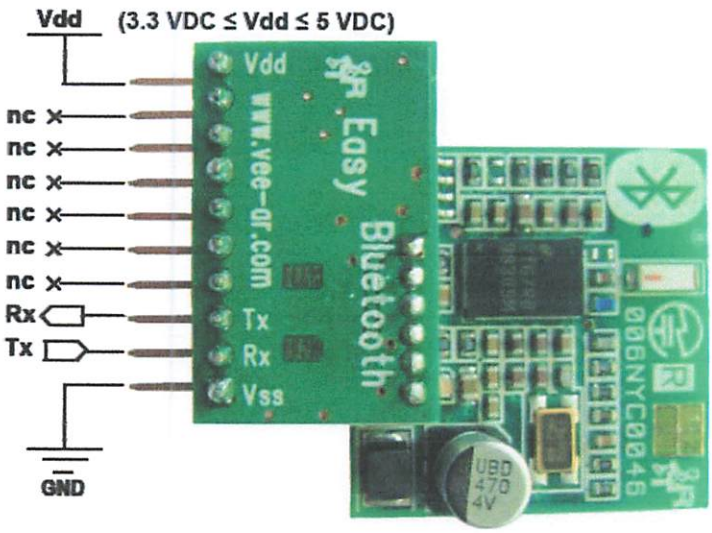
b. Pin 10

Pin ini merupakan CLK yang digunakan untuk mengatur outputan data dari ADC yang kemudian dikirim ke mikrokotroler.

3.3.2. *Bluetooth Serial Adapter* RBT-001

Pada pembuatan skripsi ini, *Bluetooth Serial Adapter* akan difungsikan sebagai *receiver* dari perangkat Bluetooth tersebut. *Bluetooth Serial Adapter* ini harus terkoneksi dengan mikrokontroler AT89S52 dengan memanfaatkan pin – pin yang terdapat pada mikrokontroler AT89S52 dan *Bluetooth Serial Adapter*.

Adapun pin-pin yang digunakan pada komunikasi data antara *Bluetooth Serial Adapter* dengan mikrokontroler dapat ditunjukan pada gambar di bawah ini :



Gambar 3.3. Pin-pin Pada Bluetooth

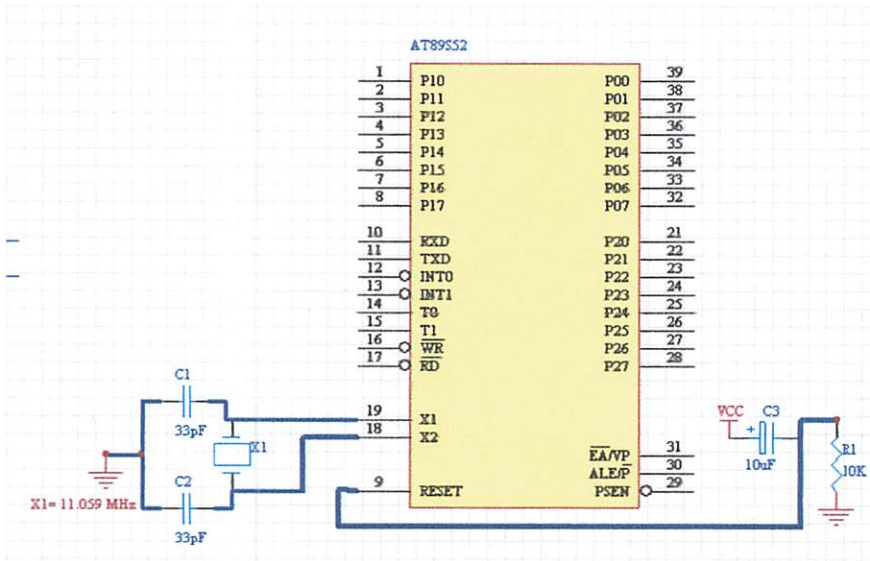
a. Pin TX digunakan untuk komunikasi serial yaitu mengirim data ke Mikrokontroler yang diterima RXD pada Mikrokontroler.

b. Pin RX digunakan untuk Komunikasi data yaitu menerima data dari Mikrokontroler yang dikirimkan melalui pin TXD pada Mikrokontroler.

3.3.3. Mikrokontroler AT89S52

Disini mikrokontroler berfungsi sebagai pengolah data dan pengendali dari alat, agar dapat melakukan prosesnya harus didukung oleh beberapa komponen tambahan, yakni berupa rangkaian *clock* dan rangkaian *reset*.

Dibawah ini merupakan gambar rangkaian sistem minimum mikrokontroler AT89S52 :



Gambar 3.4. Rangkaian Sistem Minimum Mikrokontroler AT89S52

Untuk mereset mikrokontroler AT89S52, maka pin RST diberi logika tinggi selama sekurangnya dua siklus mesin. Kecepatan proses yang dilakukan oleh mikrokontroler ditentukan oleh sumber *clock* yang mengendalikan mikrokontroler tersebut. Sistem yang dirancang ini menggunakan osilator internal yang telah tersedia dalam *chip* AT89S52. Untuk menentukan frekuensi osilatornya cukup dengan menghubungkan kristal dalam pin 19 (X_1) dan pin 18 (X_2) serta dua buah kapasitor ke *ground*.

3.3.3.1. Perancangan dan Pengaturan *Port* Mikrokontroler AT89S52

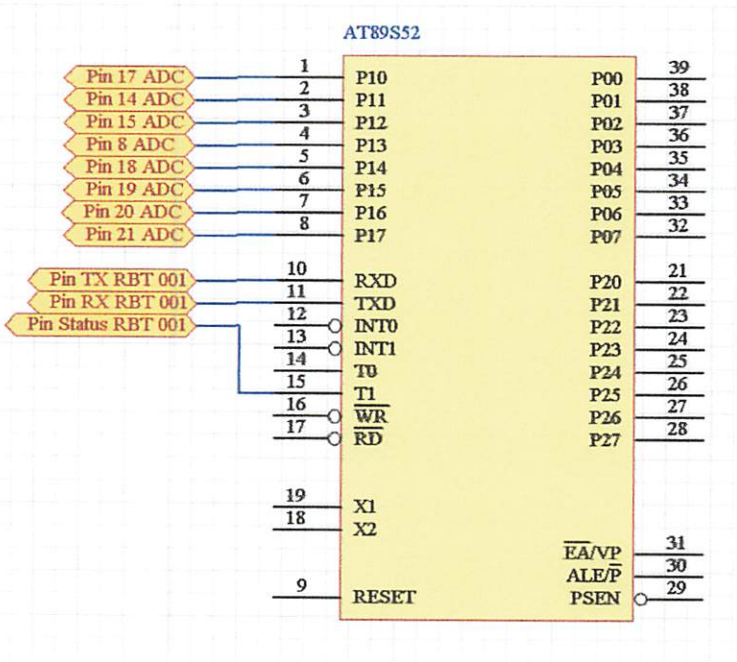
Pada perancangan port ini akan dipaparkan port apa saja yang akan digunakan pada Mikokontroler AT89S52 yaitu:

a. P1.0 sampai dengan P1.7 digunakan sebagai pengambilan data yang dikrim oleh ADC.

b. RXD digunakan sebagai port TX untuk *Bluetooth Serial Adapter.*

c. TXD digunakan sebagai port RX untuk *Bluetooth Serial Adapter.*

d. T1 digunakan sebagai port status untuk *Bluetooth Serial Adapter.*



Gambar 3.5. Port Pada Mikrokontroler AT89S52

3.4. Perancangan Perangkat Lunak (*Software*)

Perangkat lunak adalah sebuah jembatan yang menghubungkan keseluruhan komponen yang ada pada sebuah komputer. Pada perancangan ini ada dua macam, yaitu perancangan perangkat lunak untuk mikrokontroler

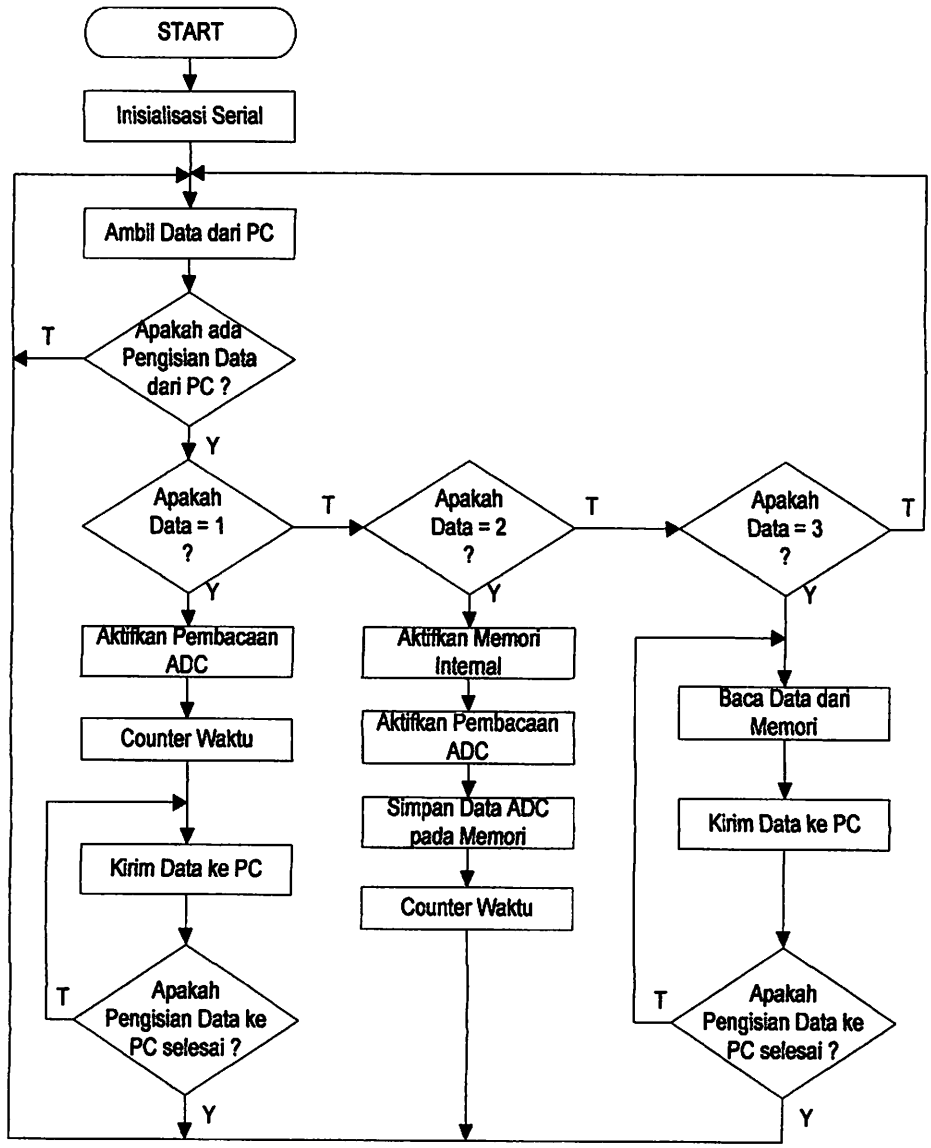
menggunakan bahasa *Assembly* Mikrokontroler standard MCS-51 dan perancangan perangkat lunak untuk komputer menggunakan *Delphi.*

3.4.1. Program Aplikasi Mikrokontroler

Program mikrokontroler bertujuan untuk mengontrol masukan dan keluaran. Hal-hal yang dikontrol mikrokontroler adalah proses komunikasi serial antara *Bluetooth Serial Adapter* dengan *Bluetooth USB Adapter* yang terdapat pada Komputer.

Di dalam proses komunikasi serial antara mikrokontroler dengan komputer terlebih dahulu ditentukan *baudrate* yang digunakan. *Baudrate* yang dibangkitkan *Timer1* dengan *Timer2* (8 bit *auto reload*) yang hanya menggunakan *register* TH1. Pada sistem ini digunakan *baudrate* sebesar 9600 dengan SMOD = 0 dengan menggunakan Frekuensi_osilator = 11.0592 MHz.

3.4.1.1. *Flowchart* Program Mikrokontoler

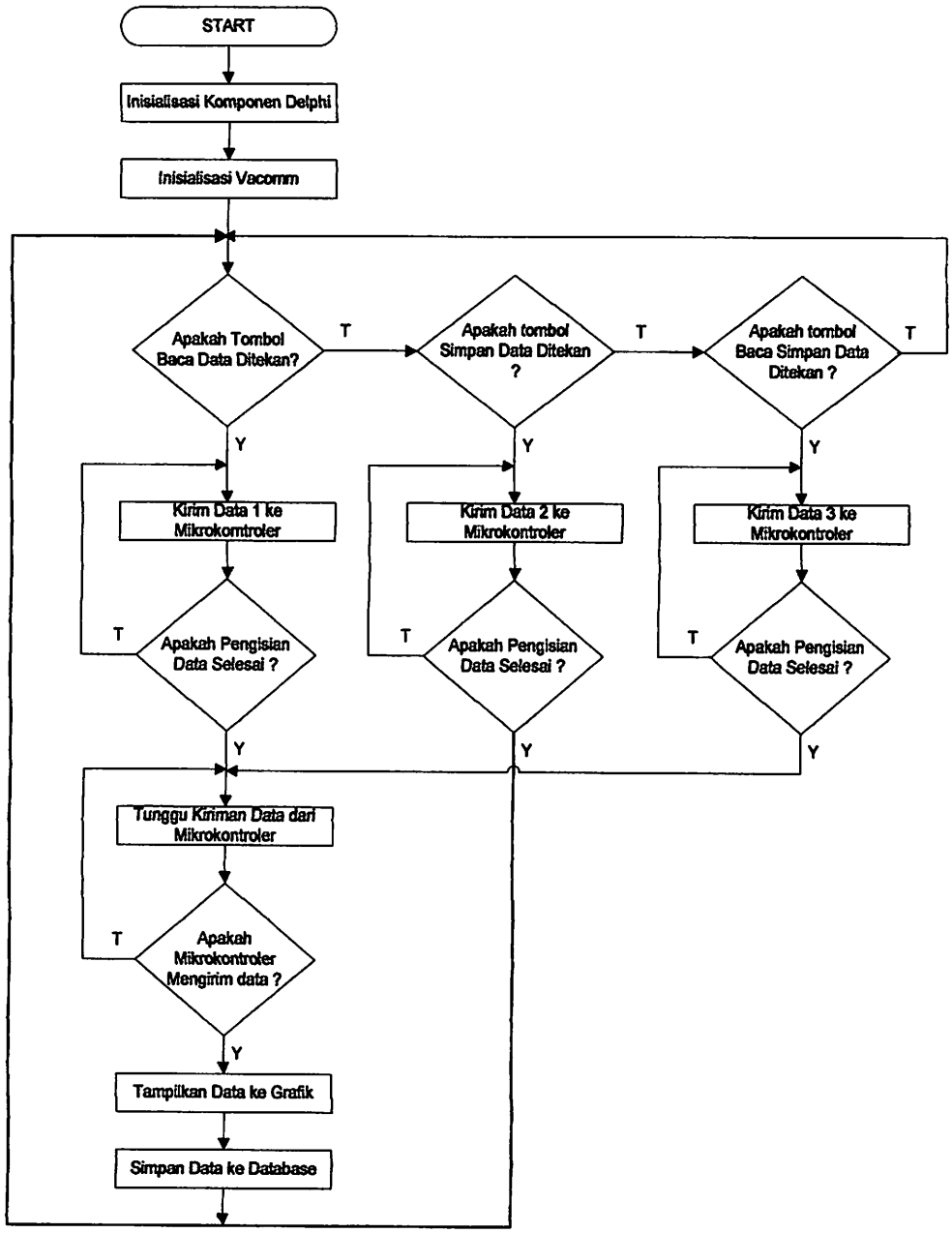


Gambar 3.6. *Flowchart* Program Mikrokontroler

3.4.2. Program Aplikasi Komputer

Program aplikasi adalah *software* di dalam komputer yang berfungsi untuk melakukan pengendalian dengan dunia luar. Program komputer ini bertujuan untuk mengorganisasi komunikasi antara komputer sebagai pengolah data dengan mikrokontroler, sehingga perangkat lunak dalam program komputer perlu diketahui terlebih dahulu aturan-aturan atau protokol komunikasi yang digunakan untuk mengatur jalannya komunikasi antara komputer dengan mikrokontroler. Dalam perencangan ini menggunakan bahasa pemrograman *visual* yaitu *Delphi*. Pada program aplikasi ini menggunakan *procedure delay* untuk mengambil sistem pewaktuan pada komputer serta sebagai jeda waktu program pengiriman data dari komputer ke mikrokontroler.

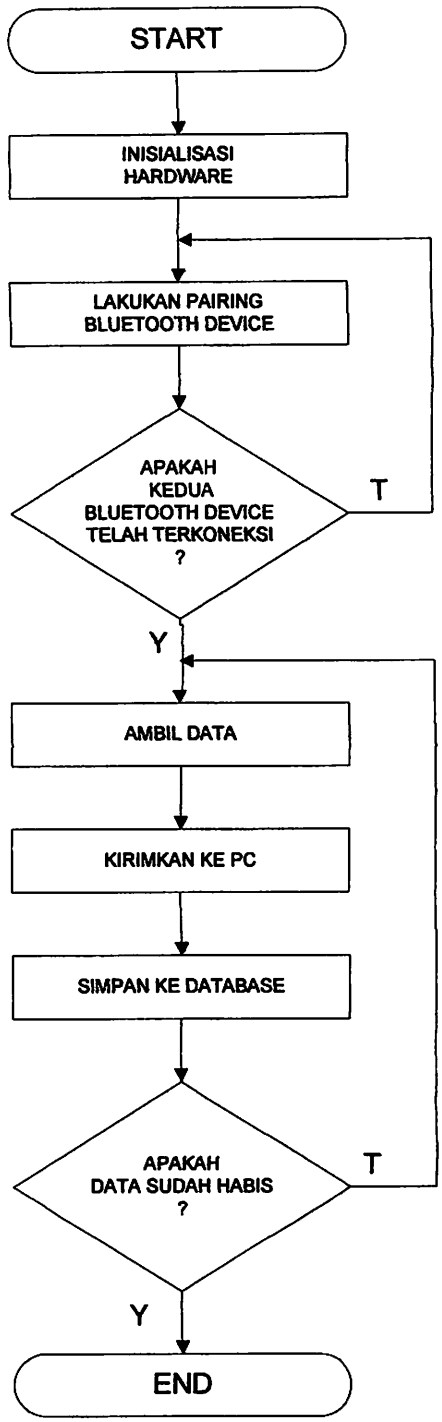
3.4.2.1. *Flowchart* Program Komputer



Gambar 3.7. *Flowchart* Program Komputer

3.4.3. *Flowchart* Sistem

Dibawah ini merupakan gambar dari flowchart keseluruhan sistem :



Gambar 3.8. *Flowchart* Keseluruhan Sistem

BAB IV

PENGUJIAN SISTEM DAN PEMBAHASAN

4.1. Pendahuluan

Dalam bab ini dibahas tentang pengukuran dan pengujian alat yang telah dibuat, yang meliputi perangkat keras (*hardware*) dan perangkat lunak (*software*). Hal ini bertujuan untuk mengetahui apakah sistem yang dibuat tersbut telah berfungsi sesua dengan yang diharapkan. Pengujian yang dilakukan terhadap sistem tersebut meliputi sistem keseluruhan maupun subsistemnya. Berikut ini merupakan penjelasan mengenai prosedur pengujian dan data hasil pengujian.

4.2. Tujuan Pengujian

Tujuan pengujian pada masing-masing blok pada perancangan dan pembuatan data logger menggunakan bluetooth ini adalah:

a. Untuk mengetahui apakah tiap blok rangkaian berfungsi dengan baik.
b. Untuk mengetahui prinsip kerja pada perancangan dan pembuatan data logger menggunakan bluetooth sebagai sarana pengiriman data.

4.3. Pengujian Mikrokontroler AT89S52

Pada pengujian mikrokontroler ini bertujuan untuk mengetahui kondisi awal dari mikrokontroler apakah sudah sesuai dengan yang diharapkan.

Adapun peralatan yang digunakan adalah sebagai berikut:

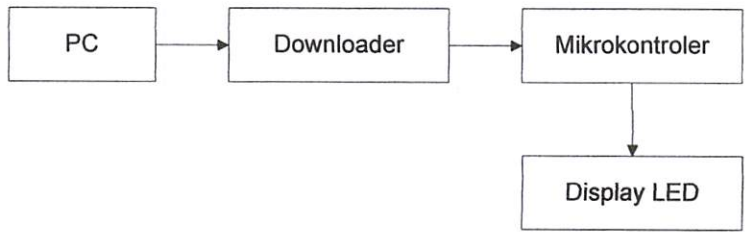
a. Komputer (PC).

b. Downloader Mikrokontroler MCS 51.

c. Catu daya 5 volt.

Tahapan pengujian mikrokontroler AT89S52 adalah sebagai berikut:

1. Rangkaian dibuat seperti gambar di bawah ini:



Gambar 4.1. Diagram Blok Pengujian Mikrokontroler

2. Memberikan catu daya 5 volt.
3. Membuat program yang digunakan dalam pengujian mikrokontroler. Program yang digunakan merupakan program sederhana yang meletakan data F0 H dan 0F H pada *port* 0 mikrokontroler AT89S52.
4. Mengamati keluaran pada Display LED.

Dibawah ini merupakan hasil pengujian dari Mikrokontroler AT89S52 dengan menggunakan LED.

Tabel 4.1. Hasil Pengujian Mikrokontroler AT89S52

Kondisi	Keluaran pada LED Display							
	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7
Pertama	0	0	0	0	1	1	1	1
Kedua	1	1	1	1	0	0	0	0

Keterangan:

a. Kondisi *bit low* (0) = LED mati.

b. Kondisi *bit high* (1) = LED nyala.

Dari hasil pengujian, dapat dilihat bahwa di *port* 0 akan memberikan logika 0F H dan F0 H secara bergantian.

4.4. Pengujian *Bluetooth* RBT-001

Pada Pengujian *Bluetooth* RBT-001 ini bertujuan untuk mengetahui apakah antara RBT-001 dengan perangkat USB *Bluetooth dongle* dapat terkoneksi dengan baik.

Adapun peralatan yang diigunakan adalah sebagai berikut:

a. Komputer (PC).

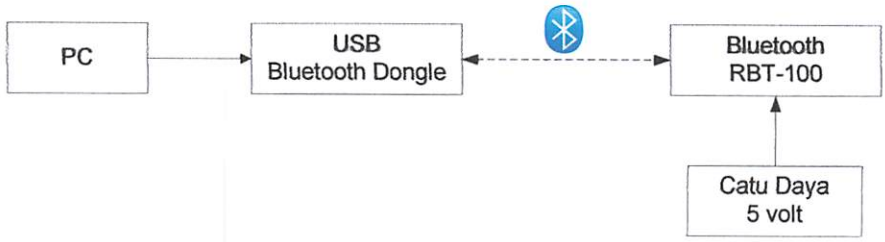
b. *Bluetooth* RBT-001.

c. USB *Bluetooth dongle*

d. Catu daya 5 volt.

Tahapan pengujian *Bluetooth* RBT-001 adalah sebagai berikut:

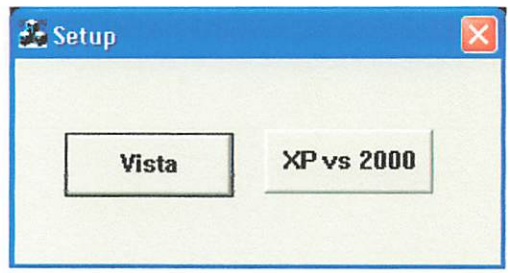
1. Rangkaian dibuat seperti gambar dibawah ini:



Gambar 4.2. Diagram Blok Pengujian Bluetooth RBT-001

2. Melakukan instalasi *driver* Bluetooth Dongle ke computer, langkah-langkahnya adalah sebagai berikut:

a) Masukan *CD driver* ke dalam *CD ROM* pada computer, maka secara otomatis computer akan mendeteksi *CD driver* tersebut sehingga akan tampil di layar monitor sebagai berikut:



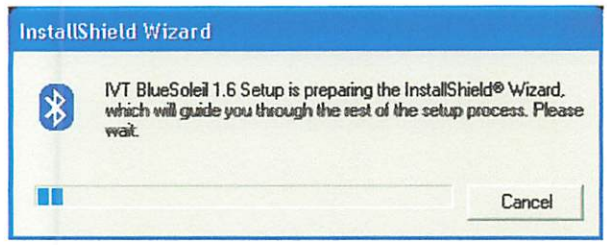
Gambar 4.3. Kotak Dialog *Setup*

b) Pilih *Vista* atau *XP vs 2000*. Sesuaikan dengan *Operating system (OS)* yang digunakan. Dalam hal ini kita memilih *XP vs 2000, karena Operating system (OS)* yang digunakan adalah *Windows XP*.

c) Kemudian tentukan bahasa yang digunakan dah pilih "*OK*", seperti yang ditampilkan pada gambar berikut:

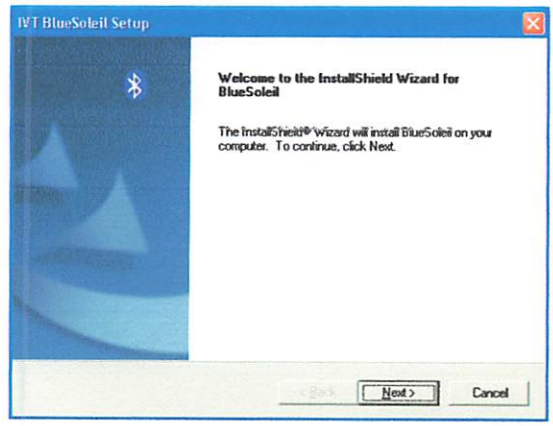
Gambar 4.4. Kotak Dialog *Choose Setup Language*

d) Kemudian pada proses awal instalasi akan ditampilkan pada layar sebagai berikut:



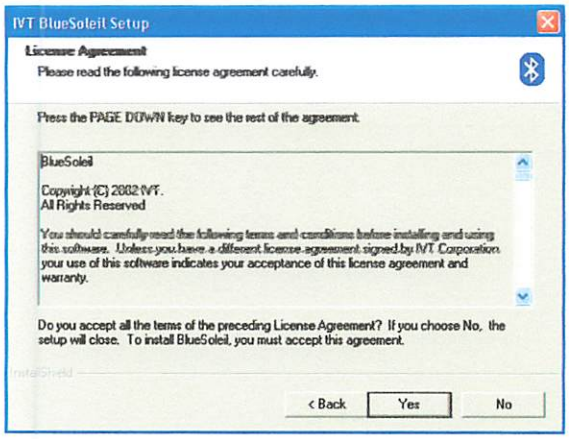
Gambar 4.5. Kotak Dialog *InstallShield Wizard*

e) Setelah proses awal selesai, maka pada layar akan tampil sesuai dengan gambar di bawah ini, kemudian klik "*Next*" untuk melanjutkan instalasi ke tahap selanjutnya.



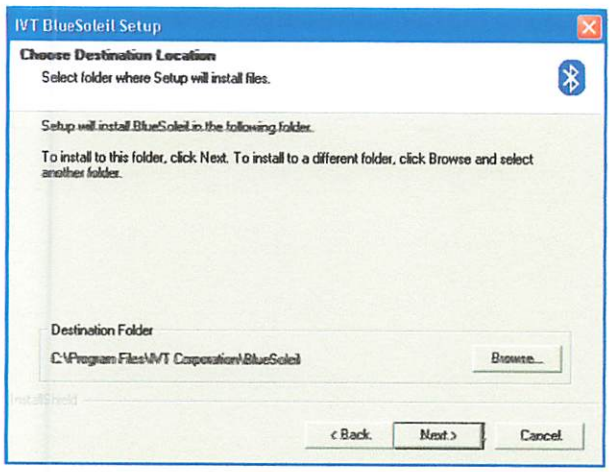
Gambar 4.6. Kotak Dialog *IVT BlueSoleil Setup*

f) Kemudian pada tahap selanjutnya akan tampil kotak dialog *license agreement*, seperti gambar di bawah ini, dan klik "*Yes*" umtuk melanjutkan.



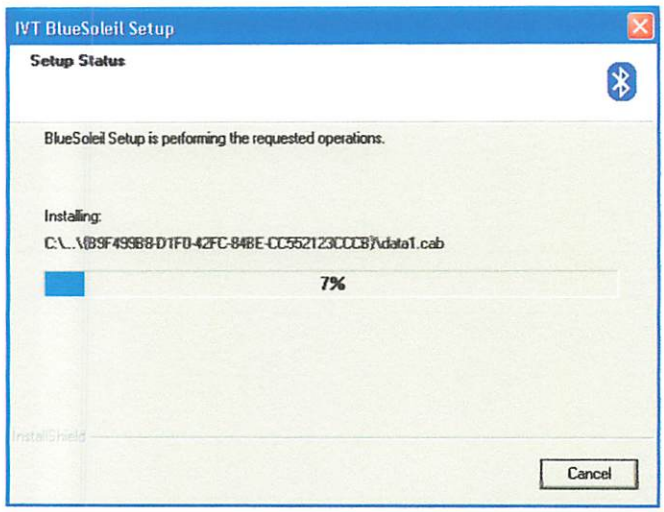
Gambar 4.7. Kotak Dialog *license agreement*

g) Pada proses instalasi selanjutnya menentukan lokasi instalasi pada computer, untuk mengubah lokasi intalasi pilih "*Browse*" kemudian pilih lokasi yang diinginkan, kemudian klik "*Next*" untuk melanjutkan instalasi, seperti yang terlihat pada gambar di bawah ini



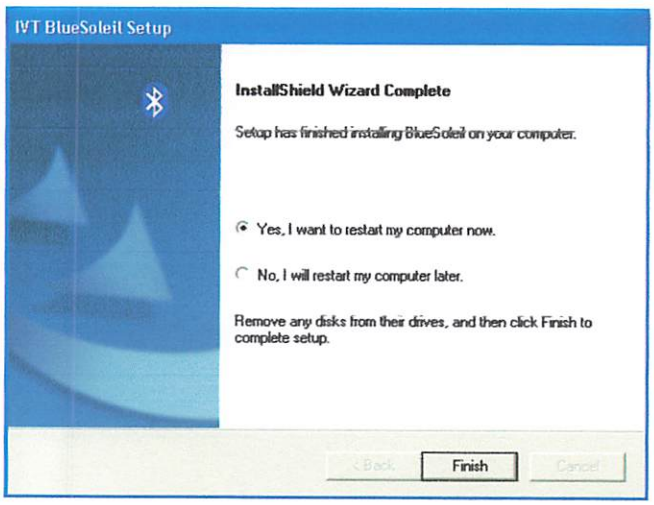
Gambar 4.8. Kotak Dialog *Choose Destination Location*

h) Pada proses intalasi maka pada layar akan tampil gambar seperti dibawah ini:



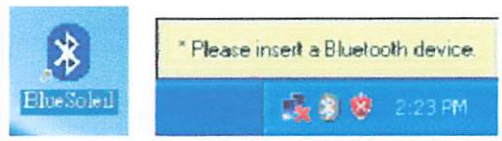
Gambar 4.9. Kotak Dialog *Setup Status*

i) Setelah proses instalasi selesai maka akan tampil dilayar seperti gambar di bawah ini:



Gambar 4.10. Kotak Dialog *Installshield Wizard Complete*

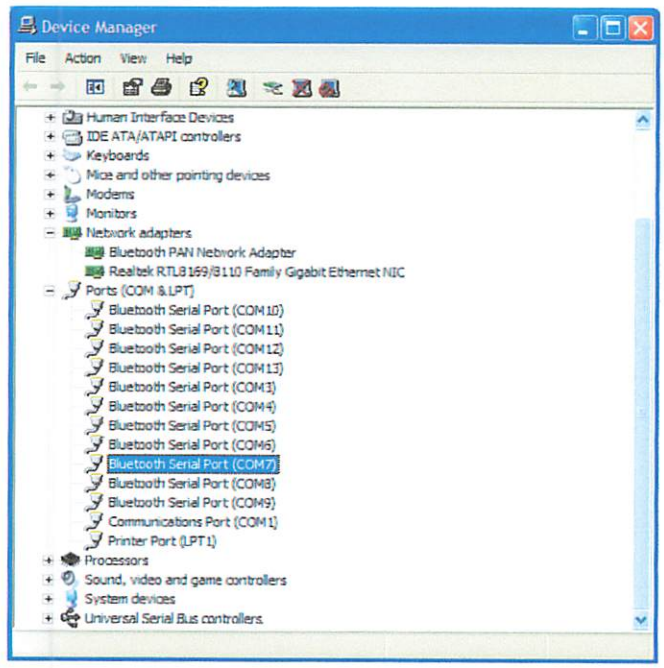
j) Setelah klik "*Finish*" maka akan muncul 2 buah *bluetooth icons* pada *desktop* dan *taskbar windows*, seperti yang terlihat pada gambar berikut:



Gambar 4.11.

Bluetooth Icons pada *Desktop* dan *Taskbar Windows*

k) Dengan demikian maka komputer akan mengenali adanya *virtual serial port (VSP)* yang dapat dilihat dengan membuka *device manager* dilokasi sistem control panel kemudian pilih *hardware* dan klik *device manager*. Pilih pada bagian jenis *device*, dan *device* akan terlihat menanmbah serial port dengan nama "*Bluetooth serial port*".

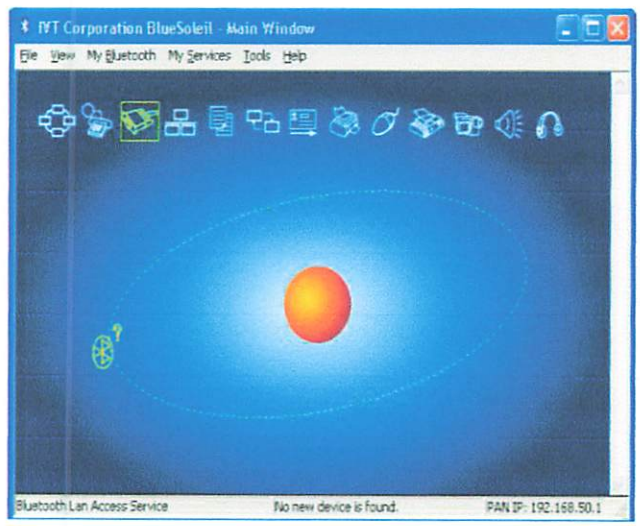


Gambar 4.12. Kotak Dialog *Device Manager* pada Komputer

3. Memasang USB *Bluetooth Dongle* ke port USB komputer.
4. Menghubungkan Pin RX dan TX pada *Bluetooth* RBT-001.
5. Menghubungkan *Bluetooth* RBT-001 dengan catu daya.

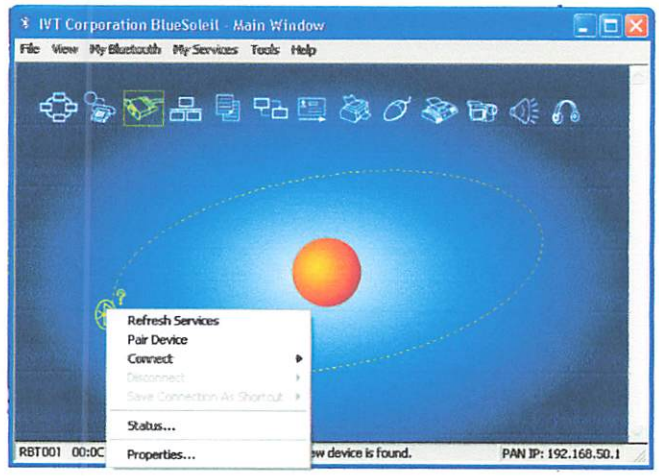
6. Melakukan proses pencarian *Bluetooth* RBT-001 melalui komputer. Langkah-langkahnya adalah sebagai berikut:

   a) Buka *icon* Bluesoil pada *desktop*, maka secara otomatis *Bluetooth Dongle* akan mendeteksi adanya *device Bluetooth* lain yang aktif dalam jangkauanya. Dalam hal ini *device* RBT-001, seperti yang terlihat pada gambar di bawah ini



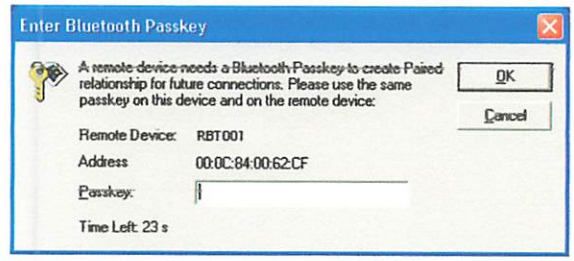
Gambar 4.13. Kotak Dialog *Main Window*

   b) Melakukan proses *pairing* antara *Bluetooth Dongle* dengan RBT-001. Dengan cara klik kanan pada *icon* RBT-001 pilih *pair device*. Seperti yang terlihat pada gambar dibawah ini.



Gambar 4.14. Kotak Dialog *Proses Pairing*

Maka *Bluetooth Dongle* secara otomatis akan meminta untuk memasukan kode *passkey* agar kedua *device Bluetooth* tersebut dapat terkoneksi. *Passkey* standar dari RBT-001 adalah "0000". Seperti yang terlihat pada gambar di bawah ini:

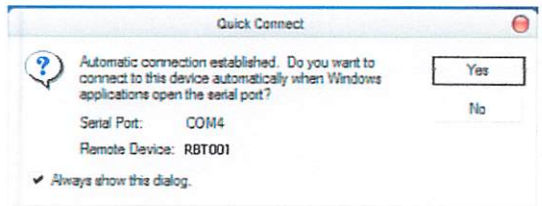


Gambar 4.15. Kotak Dialog *Enter Bluetooth Passkey*

c) Setelah proses *pairing* selesai, selanjutnya adalah melakukan koneksi antara kedua *device* tersebut dengan cara klik kanan pada *icon* RBT-001 kemudian pilih *connect, Bluetooth Serial Port Service*. *Virtual Com* yang digunakan kedua *device* ini agar dapat berkomunikasi adalah *Com 4*. Seperti yang terlihat pada gambar dibawah ini:

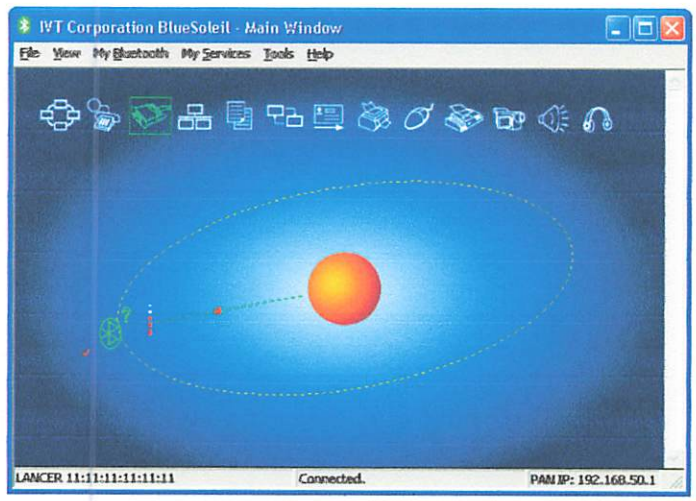
Gambar 4.16.

Kotak Dialog Proses Koneksi Kedua *Bluetooth Device*



Gambar 4.17.

Kotak Dialog *Virtual Com* yang Digunakan Oleh RBT-001

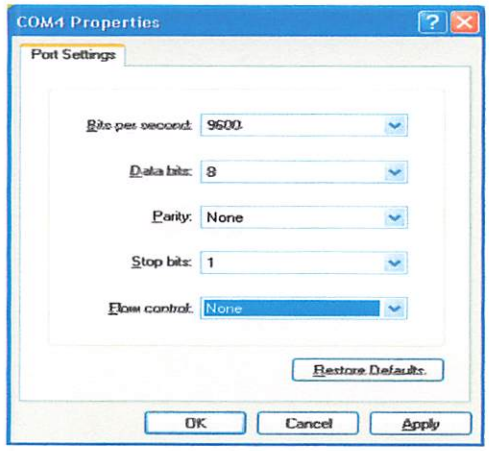


Gambar 4.18. Kotak Dialog Kedua *Device* Telah Terkoneksi

7. Membuka program *Hyper Terminal* dengan tujuan untuk mengetahui apakah pengiriman secara serial tersebut sudah sesuai dengan apa yang diinginkan. Langkah-langkahnya adalah dengan menentukan *Com Port* yang digunakan dan menentukan nilai *baudrate*nya terlebih dahulu. Seperti yang terlihat pada gambar di bawah ini:

Gambar 4.19.

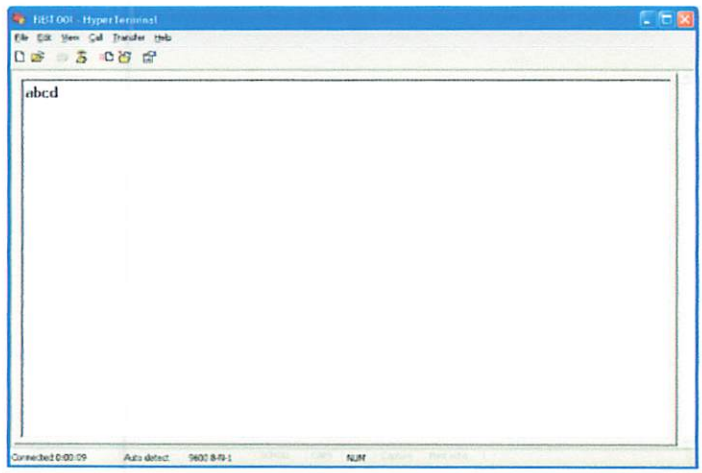
Kotak Dialog Menentukan *Com Port* yang Digunakan



Gambar 4.20.

Kotak Dialog Menentukan Nilai *Baudrate*

Dari hasil pengujian RBT-001 dengan menggunakan Hyper Terminal, apabila tombol pada keyboard ditekan maka karakter yang ditekan akan ditampilkan pada program Hyper Terminal tersebut. Seperti yang terlihat pada gambar di bawah ini:



Gambar 4.21. Tampilan pada Program *Hyper Terminal*

4.5. Pengujian Jangkauan Komunikasi Bluetooth

Pengujian ini bertujuan untuk mengetahui jarak maksimal yang mampu dicapai oleh koneksi antara *Bluetooth* RBT-001 dengan perangkat *USB Bluetooth Dongle*. Adapun peralatan yang dibutuhkan adalah sebagai berikut:

a. Komputer (PC).

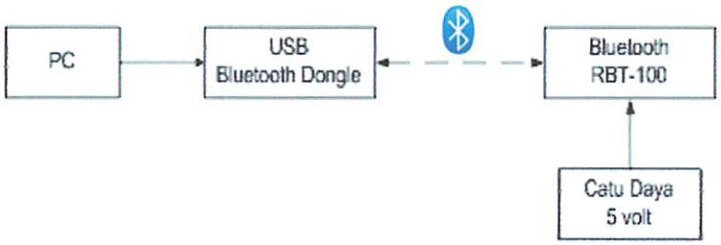
b. *Embedded Bluetooth* RBT-001.

c. *USB Bluetooth Dongle*.

d. Catu daya 5 *Volt*.

Adapun tahapan dari pengujian jangkauan komunikasi Bluetooth ini adalah sebagai berikut:

1. Rangkaian dibuat seperti gambar dibawah ini:



Gambar 4.22.
Blok Diagram Pengujian Jangkauan Komunikasi *Bluetooth*

2. Melakukan proses pairing antara *USB Bluetooth Dongle* dengan *Embedded Bluetooth* RBT-001.
3. Mengatur jarak antara *USB Bluetooth Dongle* dengan *Embedded Bluetooth* RBT-001.

Dari hasil pengujian komunikasi Bluetooth ini didapatkan bahwa jarak jangkauan maksimal antara *USB Bluetooth Dongle* dengan *Embedded Bluetooth* RBT-001 adalah 13 meter dalam kondisi *closed field*, sedangkan dalam kondisi *open field* jarak jangkauannya mencapai 16 meter. Seperti yang terlihat pada table di bawah ini:

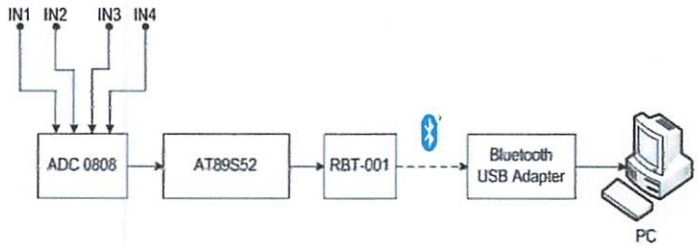
Tabel 4.2. Hasil Pengujian Komunikasi *Bluetooth*

Jarak (Meter)	Kondisi	
	Closed Field	*Open Field*
1-3	Terkoneksi	Terkoneksi
4-6	Terkoneksi	Terkoneksi
7-9	Terkoneksi	Terkoneksi
10-13	Terkoneksi	Terkoneksi
14-16	Tidak Terkoneksi	Terkoneksi

4.6. Hasil dan Analisa Pengujian Sistem

Tujuan dari pengujian sistem ini adalah untuk mengetahui apakah alat yang dirancang dapat bekerja sesuai dengan yang diinginkan. Tahapan pengujian sistem adalah sebagai berikut :

1. Rangkaian dibuat seperti gambar di bawah ini:



Gambar 4.23.
Blok Diagram Pengujian Sistem Secara Keseluruhan

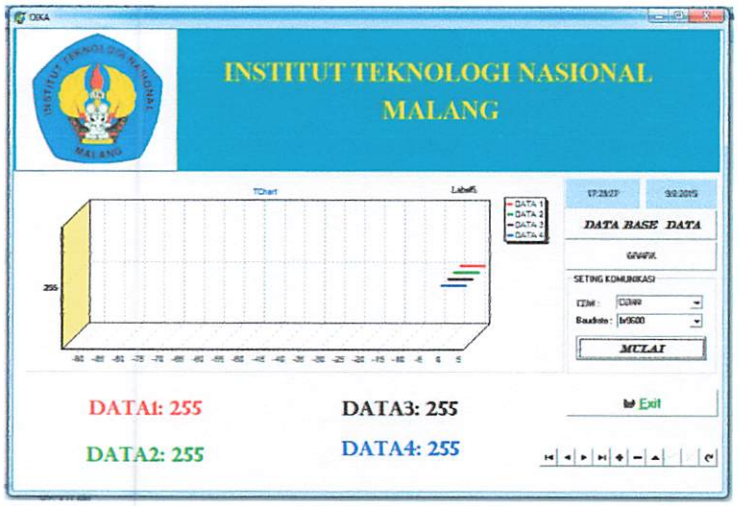
2. Melakukan proses pencarian RBT-001 melalui komputer.
3. Membuka program aplikasi Delphi yang telah dibuat sebelumnya. Seperti yang terlihat pada gambar di bawah ini:

Gambar 4.24. Program Aplikasi Delphi

4. Melakukan proses identifikasi COM. Dilakukan dengan cara membuka program aplikasi Delphi, kemudian pilih COM yang digunakan, seperti yang terlihat pada gambar di bawah ini:

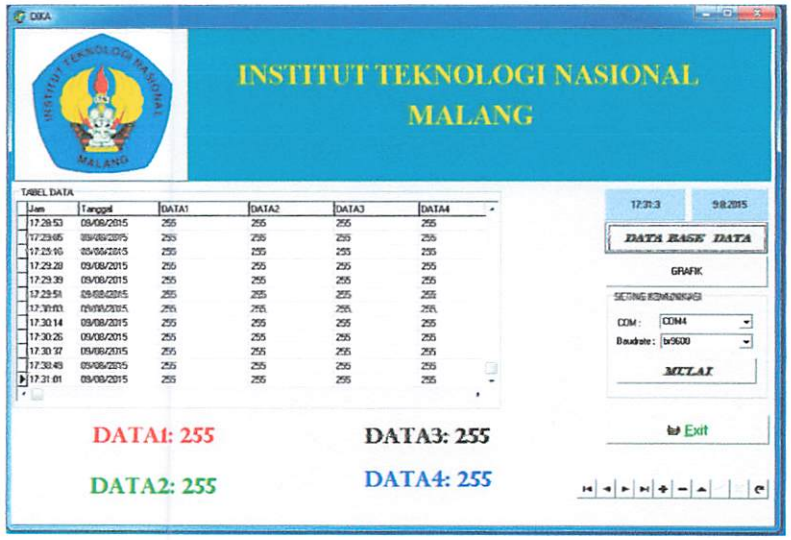
Gambar 4.25. Proses Identifikasi COM

5. Tekan tombol mulai pada program aplikasi Delphi, kemudian akan terlihat grafik keluaran dari potensiometer seperti gambar dibawah ini:



Gambar 4.26. Tampilan Saat Program Berjalan

6. Langkah berikutnya adalah menekan tombol "Data Base Data". Kemudian disitu akan terlihat tabel data logger yang tersimpan, seperti yang terlihat pada gambar di bawah ini:



Gambar 4.27. Tampilan Data Logger Pada Program Delphi

Dari hasil pengujian sistem yang telah dilakukan, didapatkan bahwa data yang ingin disimpan oleh *Data Logger* dapat dikirimkan dengan baik oleh *Bluetooth* RBT-001 yang terhubung dengan mikrokontroler AT89S52, dan dapat diterima dengan baik oleh *Bluetooth USB Dongle.*

4.7. Spesifikasi Alat

Adapun spesifikasi dari Alat Data Logger menggunakan Bluetooth adalah sebagai berikut:

a. Komputer (PC).

b. USB Bluetooth Dongle.

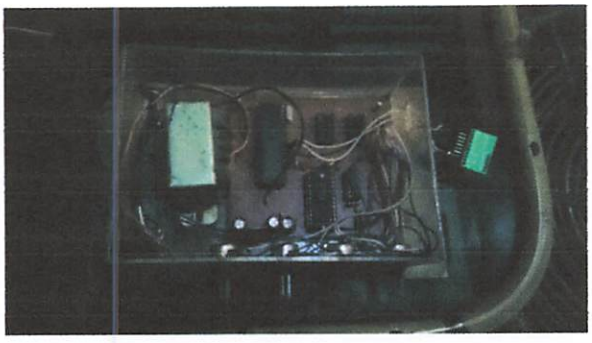
c. Bluetooth RBT-001.

d. Mikrokontroler AT89S52.

e. ADC 0808.

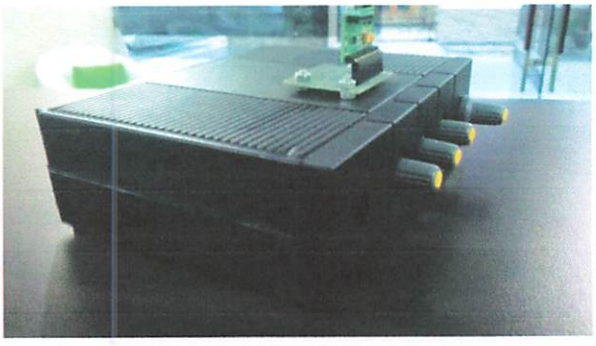
f. 4 buah potensiometer.

Dibawah ini merupakan gambar dari alat pembuatan Data Logger menggunakan Bluetooth sebagai sarana pengiriman data:



Gambar 4.28.

Alat Data Logger Menggunakan Bluetooth



Gambar 4.29.

Alat Data Logger Menggunakan Bluetooth Keseluruhan

BAB V

PENUTUP

5.1. Kesimpulan

Dari pembuatan "Data Logger Menggunakan Bluetooth Sebagai Sarana Pengiriman Data" ini dapat di ambil kesimpulan sebagai berikut :

1. Alat ini hanya digunakan untuk menampilkan data yang hanya dibuat untuk 4 inputan saja. Meskipun dapat di *expand* menjadi 8 inputan. Karena sudah terprogram pada aplikasi deplhi itu sendiri.
2. Jangkauan komunikasi antara *USB Bluetooth Dongle* dengan *Bluetooth* RBT-001 pada alat ini mencapai 13 meter dalam kondisi tertutup (*Closed Field*), sedangkan pada kondisi terbuka (*Open Field*) mencapai 16 meter.
3. Alat ini hanya dapat digunakan untuk pembacaan data tergantung pada potensiometer, meskipun dapat diaplikasikan pada pembacaan data lainnya.
4. Telah berhasil dibuat suatu alat *Data Logger* nirkabel dengan menggunakan media *Bluetooth* sebagai sarana pengiriman data ke komputer.

5.2. Saran

Alat "Data Logger Menggunakan Bluetooth Sebagai Sarana Pengiriman Data" ini masih memiliki banyak kekurangan dan keterbatasan. Sehingga nantinya diharapkan dapat dikembangkan

untuk mengatasi keterbatasan tersebut. Untuk pengembangan nantinya agar dapat digunakan lebih spesifik seperti pencatatan suhu, getataran, cahaya, dan lain sebagainya tanpa mensimulasikannya menggunakan potensiometer.

DAFTAR PUSTAKA

1. Romy Budhi Widodo, Joseph Dedy Irawan, *INTERFACING PARAREL DAN SERIAL MENGGUNAKAN DELPHI*, Edisi Pertama, Graha Ilmu Yogyakarta,2007.
2. Wahana Komputer, *Teknik Antarmuka Mikrokontroller Dengan Komputer Berbasis DELPHI*, Edisi Pertama, Salemba Infotek,2006.
3. *AT89S51 Datasheet*. www.atmel.com
4. *Datasheet* RBT-001 Parallax
5. Studi dan Uji Coba Teknologi Bluetooth Sebagai Alternatif Komunikasi Data Nirkabel www.fportfolio.petra.ac.id
6. *Datasheet* ADC 0808 *http://mekatronika.net*
7. *Parallax Easy Bluetooth.* www.parallax.com
8. *Robotech Bluetooth Serial Module RBT*-001

LAMPIRAN

PERKUMPULAN PENGELOLA PENDIDIKAN UMUM DAN TEKNOLOGI NASIONAL MALANG
INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
FAKULTAS TEKNIK SIPIL DAN PERENCANAAN
PROGRAM PASCASARJANA MAGISTER TEKNIK
PT BNI (PERSERO) MALANG
BANK NIAGA MALANG
Kampus I : Jl. Bendungan Sigura-gura No.2 Telp. (0341)551431 (Hunting), Fax. (0341)553015 Malang 65145
Kampus II : Jl. Raya Karanglo, Km 2 Telp. (0341)417636 Fax. (0341)417634 Malang

BERITA ACARA UJIAN SKRIPSI
FAKULTAS TEKNOLOGI INDUSTRI

Nama	: **Dika Yudha Purnama**
Nim	: **04.12.201**
Jurusan	: **Teknik Elektro**
Konsentrasi	: **Teknik Elektronika S-1**
Masa Bimbingan	: **Semester Genap 2014-2015**
Judul	: **PERENCANAAN DAN PEMBUATAN DATA LOGGER MENGGUNAKAN BLUETOOTH SEBAGAI SARANA PENGIRIMAN DATA**

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

Pada Hari	: Jumat
Tanggal	: 10 Juli 2015
Dengan Nilai	: 74,7 (B+)

Panitia Ujian Skripsi :

Ketua Majelis Penguji

M. Ibrahim Ashari ST, MT
NIP.P. 103 010 0358

Sekertaris Majelis Penguji

Dr. Eng. I Komang Somawirata ST, MT
NIP.Y. 103 010 0361

Anggota Penguji :

Dosen Penguji I

Ir Eko Nurcahyo MT
NIP.Y. 102 870 0172

Dosen Penguji II

Yuli Wahyuni ST, MT
NIP.P. 103 010 0358

PROGRAM STUDI TEKNIK ELEKTRO S-1

FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
Kampus II : Jl, Raya Karanglo Km. 2 Malang

SURAT PERNYATAAN ORISINALITAS

Yang bertanda tangan d bawah ini:

Nama Mahasiswa	: Dika Yudha Purnama
NIM	: 04.12.201
Program Studi	: Teknik Elektro S-1
Konsentrasi	: Teknik Elekronika
Judul Skripsi	: Perencanaan dan Pembuatan Data Logger Menggunakan Bluetooth Sebagai Sarana Pengiriman Data

Dengan ini menyatakan bahwa skripsi yang saya buat adalah hasil karya sendiri, tidak merupakan plagiasi dari karya orang lain. Dalam skripsi ini tidak memuat karya orang lain, kecuali dicantumkan sumbernya sesuai ketentuan yang berlaku.

Demikian surat pernyataan ini saya buat, dan apabila di kemudian hari ada pelanggaran atas surat pernyataan ini, saya bersedia menerima sangsinya.

Malang, 20 Oktober 2015
Yang membuat pernyataan

Dika Yudha Purnama
NIM. 0412201

PROGRAM STUDI TEKNIK ELEKTRO S-1
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
Kampus II : Jl. Raya Karanglo Km. 2 Telp (0341)417636 Malang

FORMULIR BIMBINGAN SKRIPSI

Nama : Dika Yudha Purnama
Nim : 04.12.201
Masa Bimbingan : Semester Ganjil Tahun Akademik 2014-2015
Judul Skripsi : **Perencanaan dan Pembuatan Data Logger Menggunakan Bluetooth Sebagai Sarana Pengiriman Data**

No	Tanggal	Uraian	Paraf Pembimbing
1	8/07	Acc Seminar Hasil.	
2	30/07	Bab I, II, III	
3	3/08	Bab IV	
4	10/08	Bab V	
5	14/08	Acc Kompre	
6			

Malang, Agustus 2015
Dosen pembimbing,

(Dr. Eng. I Komang Somawirata ST, MT)
NIP.Y 103 010 0361

Form S-4b

PROGRAM STUDI TEKNIK ELEKTRO S-1
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
Kampus II : Jl. Raya Karanglo Km. 2 Telp (0341)417636 Malang

FORMULIR BIMBINGAN SKRIPSI

Nama : Dika Yudha Purnama
Nim : 04.12.201
Masa Bimbingan : Semester Ganjil Tahun Akademik 2014-2015
Judul Skripsi : **Perencanaan dan Pembuatan Data Logger Menggunakan Bluetooth Sebagai Sarana Pengiriman Data**

No	Tanggal	Uraian	Paraf Pembimbing
1	8/07	Acc seminar hasil	
2	7/08	Bab I. II. III	
3	4/08	BAB IV	
4	11/08	Acc Bab V	
5	19/08	Acc Ujian Skripsi	
6			

Malang, Agustus 2015
Dosen pembimbing,

(M Ibrahim Ashari ST, MT)
NIP.P 103 010 0358

Form S-4b

INSTITUT TEKNOLOGI NASIONAL
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO S-1
Jl. Raya Karanglo, Km. 2 MALANG

Formulir Perbaikan Ujian Skripsi

Dalam Pelaksanaan Ujian Skripsi Jenjang Strata 1 Jurusan Teknik Elektro Konsentrasi T.Energi Listrik,/ T. Elektronika, /T. Komputer, / T.Telekomunikasi, Maka Perlu Adanya Perbaikan Skripsi Untuk Mahasiswa:

Nama : DIKA YUDHA
NIM : 04.12201
Perbaikan Meliputi :

- Revisi Alat

Malang, 19 - 8 - 2015

(..... EKO .....)

INSTITUT TEKNOLOGI NASIONAL
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO S-1
Jl. Raya Karanglo, Km. 2 MALANG

Formulir Perbaikan Ujian Skripsi

Dalam Pelaksanaan Ujian Skripsi Jenjang Strata 1 Jurusan Teknik Elektro Konsentrasi T.Energi Listrik,/ T. Elektronika, /T. Komputer, / T.Telekomunikasi, Maka Perlu Adanya Perbaikan Skripsi Untuk Mahasiswa:

Nama : Dika Yudha Purnama
NIM : 04.12.201
Perbaikan Meliputi :

1. Abstrak
2. penulis
3. Refisi Alat.

Malang, 19 Agustus 2015

(Yuli Wahyuni, ST. MT)

INSTITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO S-1
Jl. Raya Karanglo Km 2 Malang

FORMULIR PERBAIKAN SKRIPSI

Nama : Dika Yudha Purnama
Nim : 04.12.201
Jurusan : Teknik Elektro S-1
Konsentrasi : Teknik Elektronika
Judul Skripsi : **PERENCANAAN DAN PEMBUATAN DATA LOGGER MENGGUNAKAN BLUETOOTH SEBAGAI SARANA PENGIRIMAN DATA**

Penguji	Tanggal	Uraian	Paraf
Penguji I	31 Agustus 2015	1. Alat	

Penguji	Tanggal	Uraian	Paraf
Penguji II	31 Agustus 2015	1. Abstrak 2. Penulisan 3. Alat	

Disetujui Oleh

Penguji I

Ir. Eko Nurcahyo MT
NIP.Y. 102 870 0172

Penguji II

Yuli Wahyuni ST, MT
NIP.P. 103 120 0456

Mengetahui

Dosen Pembimbing I

Dr. Eng. I Komang Somawirata ST, MT
NIP.Y. 103 010 0361

Dosen Pembimbing II

M Ibrahim Ashari ST, MT
NIP.P. 103 010 0358

PROGRAM MIKROKONTROLER

```
'Inisialisasi

$regfile = "89s52.dat"
$baud = 9600
$crystal = 11059200
$sim
$large
Startt Alias P3.3
Eoc Alias P3.4
Oe Alias P3.5
Dda Alias P3.0
Ddb Alias P3.1
Ddc Alias P3.2


Dim Adc_sensor1 As Word
Dim Adc_sensor2 As Word
Dim Adc_sensor3 As Word
Dim Adc_sensor4 As Word
Dim Data_Adc As Integer
Dim J As Integer
Dim K As Integer

P0.0 = 0
P0.1 = 0
P0.2 = 0
P0.3 = 0
Adc_sensor1 = 0
Adc_sensor1 = 0
Adc_sensor1 = 0
Adc_sensor1 = 0

Mulai:


   Baca_adc:
   Dda = 0
   Ddb = 0
   Ddc = 0
   Startt = 0
   Oe = 1

 Waitms 0.255


 Dda = 0
 Ddb = 0
 Ddc = 0
 Startt = 1
 Oe = 1

Waitms 0.255


 Dda = 0
```

```
Ddb = 0
Ddc = 0
Startt = 0
Oe = 1

 Do
  Loop Until Eoc = 0
  Waitms 66
  Data_Adc = P1
  Startt = 0
  Oe = 0
  Waitms 5
Adc_sensor1 = Data_adc



  Dda = 1
  Ddb = 0
  Ddc = 0
  Startt = 0
  Oe = 1

Waitms 0.255


Dda = 1
Ddb = 0
Ddc = 0
Startt = 1
Oe = 1

Waitms 0.255


Dda = 1
Ddb = 0
Ddc = 0
Startt = 0
Oe = 1

  Do
  Loop Until Eoc = 0
  Waitms 66
     Data_adc = P1
  Startt = 0
  Oe = 0
  Waitms 5
Adc_sensor2 = Data_adc


   Dda = 1
  Ddb = 0
  Ddc = 0
  Startt = 0
  Oe = 1

Waitms 0.255
```

```
Dda = 1
Ddb = 0
Ddc = 0
Startt = 1
Oe = 1

Waitms 0.255


Dda = 1
Ddb = 0
Ddc = 0
Startt = 0
Oe = 1

  Do
  Loop Until Eoc = 0
  Waitms 66
     Data_adc = P1
  Startt = 0
  Oe = 0
  Waitms 5
Adc_sensor3 = Data_adc

   Dda = 1
  Ddb = 0
  Ddc = 0
  Startt = 0
  Oe = 1

Waitms 0.255


Dda = 1
Ddb = 0
Ddc = 0
Startt = 1
Oe = 1

Waitms 0.255


Dda = 1
Ddb = 0
Ddc = 0
Startt = 0
Oe = 1

  Do
  Loop Until Eoc = 0
  Waitms 66
     Data_adc = P1
  Startt = 0
  Oe = 0
  Waitms 5
Adc_sensor4 = Data_adc
```

```
Print Adc_sensor1 ; Adc_sensor2 ; Adc_sensor3 ; Adc_sensor4
Goto Mulai
```

```
PROGRAM DELPHI

unit Unit1;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms,
  Dialogs, ExtCtrls, StdCtrls, TeEngine, Series, TeeProcs, Chart,
Buttons,
  VaClasses, VaComm, DB, DBTables, DBCtrls, Grids, DBGrids;

type
  TForm1 = class(TForm)
    Panel1: TPanel;
    Image1: TImage;
    Label34: TLabel;
    BitBtn2: TBitBtn;
    Panel3: TPanel;
    Panel2: TPanel;
    BitBtn4: TBitBtn;
    Button1: TButton;
    Timer1: TTimer;
    Timer2: TTimer;
    Memo1: TMemo;
    Timer4: TTimer;
    VaComm1: TVaComm;
    Chart1: TChart;
    Label5: TLabel;
    Series1: TFastLineSeries;
    DataSource1: TDataSource;
    Series2: TFastLineSeries;
    Series3: TFastLineSeries;
    Series4: TFastLineSeries;
    Label12: TLabel;
    Label13: TLabel;
    Label14: TLabel;
    Label15: TLabel;
    GroupBox1: TGroupBox;
    Edit1: TEdit;
    Edit2: TEdit;
    Edit3: TEdit;
    Edit5: TEdit;
    Edit4: TEdit;
    Edit6: TEdit;
    Edit7: TEdit;
    Edit8: TEdit;
    Timer3: TTimer;
    Edit9: TEdit;
    Edit10: TEdit;
    GroupBox2: TGroupBox;
    DBGrid1: TDBGrid;
    DBNavigator1: TDBNavigator;
    Button2: TButton;
    Edit11: TEdit;
    Edit12: TEdit;
    Edit13: TEdit;
```

```
Edit14: TEdit;
Edit15: TEdit;
Edit16: TEdit;
Edit17: TEdit;
Edit19: TEdit;
Edit20: TEdit;
Edit21: TEdit;
Edit22: TEdit;
Edit23: TEdit;
Edit24: TEdit;
Edit25: TEdit;
Edit26: TEdit;
Edit27: TEdit;
Edit28: TEdit;
Edit29: TEdit;
Edit30: TEdit;
Edit31: TEdit;
Edit32: TEdit;
Edit33: TEdit;
Edit34: TEdit;
Edit35: TEdit;
Edit36: TEdit;
Timer5: TTimer;
Memo2: TMemo;
Memo3: TMemo;
Button7: TButton;
Edit37: TEdit;
Edit38: TEdit;
Edit39: TEdit;
Edit40: TEdit;
Edit41: TEdit;
Edit42: TEdit;
Timer6: TTimer;
Edit43: TEdit;
Button8: TButton;
Edit18: TEdit;
Button3: TButton;
Button5: TButton;
Button6: TButton;
Edit44: TEdit;
Edit45: TEdit;
Edit46: TEdit;
Table1: TTable;
Table1Data: TStringField;
Table1Jam: TTimeField;
Table1Tanggal: TDateField;
Table1Data1: TStringField;
Table1Data2: TStringField;
Table1Data3: TStringField;
Table1Data4: TStringField;
dbnvgr1: TDBNavigator;
grp1: TGroupBox;
cbb1: TComboBox;
lbl1: TLabel;
lbl2: TLabel;
cbb2: TComboBox;
btn1: TButton;
lbl3: TLabel;
```

```
    procedure Timer1Timer(Sender: TObject);

    procedure BitBtn4Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure VaComm1RxChar(Sender: TObject; Count: Integer);

    procedure Timer4Timer(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure btn1Click(Sender: TObject);
    procedure Button6Click(Sender: TObject);
    procedure Timer5Timer(Sender: TObject);
    procedure Button7Click(Sender: TObject);
    procedure Button8Click(Sender: TObject);
    procedure Button12Click(Sender: TObject);
    procedure Button13Click(Sender: TObject);
    procedure Button14Click(Sender: TObject);


  private
    { Private declarations }
      Pesan,Pesan1 : String;
       A,B,C,D,E,F,G,H : REAL;
  public

    { Public declarations }
  end;

var
  Form1: TForm1;
  w:string;
   x:string;
   x1:string;
implementation

//uses Unit2, Unit3, Unit4, Unit5;

{$R *.dfm}

procedure TForm1.Timer1Timer(Sender: TObject);
var
 datajam,perwaktu,conter : integer;
 Jam,Menit,Detik,MiliDetik,year,month,day : Word;
 Jam1,menit1,detik1,msec1:word;
 Totalwaktu          :TDateTime;

begin
    DecodeTime(Time, Jam , Menit, Detik, MiliDetik);
    decodeDate( Date,day,month,year);

  inc (datajam);

  Panel3.Caption := IntToStr(Jam) +':'+
  //FORM1.Edit6.Text := IntToStr(Jam) +':'+
                      IntToStr(Menit)+':'+
                      IntToStr(Detik);
```

```
  panel2.Caption := inttostr(year)+':'+
                        IntToStr(month)+':'+
                      IntToStr(day);
   inc   (perwaktu);
     if form1.Edit3.Text = form1.Edit5.Text then
    begin
    end
    else
    begin

    end;



  end;

procedure TForm1.BitBtn4Click(Sender: TObject);
begin
application.Terminate ;
end;

procedure TForm1.FormCreate(Sender: TObject);
begin
A := 0;
B := 0;

FORM1.Memo2.Text := '';
FORM1.Memo1.Hide ;
FORM1.Memo2.Hide ;
FORM1.Memo3.Hide ;
FORM1.GroupBox1.Hide;
FORM1.GroupBox2.Hide;
//FORM1.Chart1.Hide;
form1.Timer4.Enabled := false;
form1.Timer2.Enabled := false;
form1.Timer3.Enabled := false;
form1.Timer5.Enabled := false;
form1.Timer6.Enabled := false;
end;

procedure TForm1.VaComm1RxChar(Sender: TObject; Count: Integer);
VAR
  Hasil : array [1..16] of integer;
  Data,data1,data2 : string;
  k,a,b,s,JUMKARAK,JUMKARAK1,JUMKARAK2 : integer;
  lembaran,lembaran1,lembaran2 : string;
  I: Integer;
  Tmp: string;
  dataseri : byte;


begin
  //baca data yang diterima oleh comport
  Tmp := VaComm1.ReadText;

    for I := 1 to Length(Tmp) do
    case Tmp[I] of
      #10:; //lewatkan
```

```
      #13:  //Tunggu Enter
        begin
            form1.Memo3.Lines.Add(pesan);;
          pesan := '';
        end;
      else
        pesan := pesan + Tmp[I];
        form1.Edit18.Text := Pesan;
        form1.Edit19.Text := form1.Memo3.Text;
        form1.Memo3.Text := '';
        //pesan := '';
        data2 := edit18.Text;
        JUMKARAK2 := LENGTH (EDIT18.Text);
        form1.Timer5.Enabled := true;
      begin
        for s := 1 to jumkarak2 do

      Begin
         lembaran2 := Data2[s];
         form1.Edit20.Text := inttostr(s);
         case s of
         1: edit21.Text :=lembaran2;
         2: edit22.Text :=lembaran2;
         3: edit23.Text :=lembaran2;

         4: edit24.Text :=lembaran2;
         5: edit25.Text :=lembaran2;
         6: edit26.Text :=lembaran2;

         7: edit27.Text :=lembaran2;
         8: edit28.Text :=lembaran2;
         9: edit29.Text :=lembaran2;

         10: edit30.Text :=lembaran2;
         11: edit31.Text :=lembaran2;
         12: edit32.Text :=lembaran2;
       end;
       end;
       end;
 end;
  end;



procedure TForm1.Timer4Timer(Sender: TObject);
begin
 form1.Timer4.Enabled := false;
 form1.Timer2.Enabled := true;
end;

procedure TForm1.Button1Click(Sender: TObject);
begin
FORM1.GroupBox2.Show;
FORM1.Chart1.Hide;
end;

procedure TForm1.Button3Click(Sender: TObject);
begin
```

```
VaComm1.WriteText(Edit15.Text);
VaComm1.WriteChar(chr(13));  // tanda enter sebagai akhir pengiriman
form1.Edit40.Text := form1.Edit37.Text;
end;

procedure TForm1.Button2Click(Sender: TObject);
begin
FORM1.GroupBox2.Hide;

end;

procedure TForm1.FormShow(Sender: TObject);
begin
form1.Table1.DatabaseName:= extractfilepath(application.ExeName);
table1.TableName := 'dika.db';
table1.IndexFieldNames:= '';
table1.Open;
end;

procedure TForm1.btn1Click(Sender: TObject);

begin
FORM1.VaComm1.DeviceName := form1.cbb1.Text;
//FORM1.VaComm1.Baudrate := form1.cbb2.Text;
FORM1.VaComm1.Open;
VaComm1.WriteText(Edit41.Text);

end;

procedure TForm1.Button6Click(Sender: TObject);
begin
VaComm1.WriteText(Edit17.Text);
VaComm1.WriteChar(chr(13));  // tanda enter sebagai akhir pengiriman
form1.Edit40.Text := form1.Edit39.Text;
end;


procedure TForm1.Timer5Timer(Sender: TObject);
var

   t:longint;
y:array[1..4] of REAL;
Z:array[5..8] of REAL;
V1,V2,V3,V4,V5,V6,V7,V8:string;
V21,V22 : STRING;
  I: Integer;
  Tmp: string;
  Pesan1 : String;
begin
   form1.Timer5.Enabled := false;
   form1.Edit18.Text := '';
   Pesan1:= '';
   pesan := '';
   form1.Memo2.Lines.Add(form1.Edit21.Text);
   form1.Memo2.Lines.Add(form1.Edit22.Text);
   form1.Memo2.Lines.Add(form1.Edit23.Text);
   Tmp := memo2.Text;
```

```
for I := 1 to Length(Tmp) do
  case Tmp[I] of
    #10:; //lewatkan
    #13:  //Tunggu Enter
      begin
      end;
    else //bukan #10 atau #13
     Pesan1 := Pesan1 + Tmp[I];
     form1.Edit1.Text := Pesan1;
     //form1.memo2.Text := '';
     //Pesan1 := '';
   END;
 form1.memo2.Text := '';
 Pesan1 := '';
 form1.Memo2.Lines.Add(form1.Edit24.Text);
 form1.Memo2.Lines.Add(form1.Edit25.Text);
 form1.Memo2.Lines.Add(form1.Edit26.Text);
 Tmp := memo2.Text;

 for I := 1 to Length(Tmp) do
  case Tmp[I] of
    #10:; //lewatkan
    #13:  //Tunggu Enter
      begin

      end;
    else //bukan #10 atau #13
      Pesan1 := Pesan1 + Tmp[I];
      form1.Edit2.Text := Pesan1;
     //form1.memo2.Text := '';
     //Pesan1 := '';
      END;
  form1.memo2.Text := '';
 Pesan1 := '';
 form1.Memo2.Lines.Add(form1.Edit27.Text);
 form1.Memo2.Lines.Add(form1.Edit28.Text);
 form1.Memo2.Lines.Add(form1.Edit29.Text);
 Tmp := memo2.Text;

for I := 1 to Length(Tmp) do
  case Tmp[I] of
    #10:; //lewatkan
    #13:  //Tunggu Enter
      begin

      end;
    else //bukan #10 atau #13
      Pesan1 := Pesan1 + Tmp[I];
      form1.Edit3.Text := Pesan1;

   END;
 form1.memo2.Text := '';
 Pesan1:= '';
 form1.Memo2.Lines.Add(form1.Edit30.Text);
 form1.Memo2.Lines.Add(form1.Edit31.Text);
 form1.Memo2.Lines.Add(form1.Edit32.Text);
 Tmp := memo2.Text;
```

```
  for I := 1 to Length(Tmp) do
    case Tmp[I] of
      #10:; //lewatkan
      #13:  //Tunggu Enter
        begin

        end;
      else //bukan #10 atau #13
       Pesan1 := Pesan1 + Tmp[I];
       form1.Edit4.Text := Pesan1;

  end;
  form1.memo2.Text := '';
  Pesan1:= '';



   Table1.Append;
   D := D+1;
   form1.Edit10.Text := FLOATToStr(D);
   Table1.FieldByName('Data').AsString :=  form1.Edit10.Text;
   Table1.FieldByName('Data1').AsString :=  form1.Edit1.Text;
   Table1.FieldByName('Data2').AsString :=  form1.Edit2.Text;
   Table1.FieldByName('Data3').AsString :=  form1.Edit3.Text;
   Table1.FieldByName('Data4').AsString :=  form1.Edit4.Text ;
   Table1.FieldByName('Tanggal').AsDateTime:=date;
   Table1.FieldByName('Jam').AsDateTime:=TIME;
   Table1.Post;
x := form1.Edit4.Text;
y[1] := strtofloat(form1.Edit1.Text);
y[2] := strtofloat(form1.Edit2.Text);
y[3] := strtoint(form1.Edit3.Text);
y[4] := strtoint(foRm1.Edit4.Text);
V1 := floattostrf(y[1],fffixed,4,0);
V2 := floattostrf(y[2],fffixed,4,0);
V3 := floattostrf(y[3],fffixed,4,0);
V4 := floattostrf(y[4],fffixed,4,0);


form1.memo1.Lines.Add((x)+''+V1+''+V2+''+V3+''+V4);
        LABEL12.Caption:='DATA1: ' +V1+  '';
        LABEL13.Caption:='DATA2: ' +V2+  '';
        LABEL14.Caption:='DATA3: ' +V3+  '';
        LABEL15.Caption:='DATA4: ' +V4+  '';

        for t := 0 to chart1.SeriesCount -1 do
        with chart1.Series[t] do
add(y[t+1],'',clteecolor);
        with chart1.BottomAxis do
        begin
        automatic := false;
        maximum := series1.XValues.Last;
        minimum:=maximum - 100;
        END;

 form1.Edit1.Text := form1.Edit11.Text;
 form1.Edit2.Text := form1.Edit12.Text;
 form1.Edit3.Text := form1.Edit13.Text;
```

```
 form1.Edit4.Text := form1.Edit14.Text;



  end;

procedure TForm1.Button7Click(Sender: TObject);
begin
form1.Timer5.Enabled := TRUE;
end;

procedure TForm1.Button8Click(Sender: TObject);
begin
FORM1.GroupBox2.Hide;
FORM1.Chart1.Show;
end;

procedure TForm1.Button12Click(Sender: TObject);
begin
VaComm1.WriteText(Edit44.Text);
end;

procedure TForm1.Button13Click(Sender: TObject);
begin
VaComm1.WriteText(Edit45.Text);
end;

procedure TForm1.Button14Click(Sender: TObject);
begin
VaComm1.WriteText(Edit46.Text);
end;

end.
```

October 1999

National Semiconductor

ADC0808/ADC0809
8-Bit µP Compatible A/D Converters with 8-Channel Multiplexer

General Description

The ADC0808, ADC0809 data acquisition component is a monolithic CMOS device with an 8-bit analog-to-digital converter, 8-channel multiplexer and microprocessor compatible control logic. The 8-bit A/D converter uses successive approximation as the conversion technique. The converter features a high impedance chopper stabilized comparator, a 256R voltage divider with analog switch tree and a successive approximation register. The 8-channel multiplexer can directly access any of 8-single-ended analog signals.

The device eliminates the need for external zero and full-scale adjustments. Easy interfacing to microprocessors is provided by the latched and decoded multiplexer address inputs and latched TTL TRI-STATE® outputs.

The design of the ADC0808, ADC0809 has been optimized by incorporating the most desirable aspects of several A/D conversion techniques. The ADC0808, ADC0809 offers high speed, high accuracy, minimal temperature dependence, excellent long-term accuracy and repeatability, and consumes minimal power. These features make this device ideally suited to applications from process and machine control to consumer and automotive applications. For 16-channel multiplexer with common output (sample/hold port) see ADC0816 data sheet. (See AN-247 for more information.)

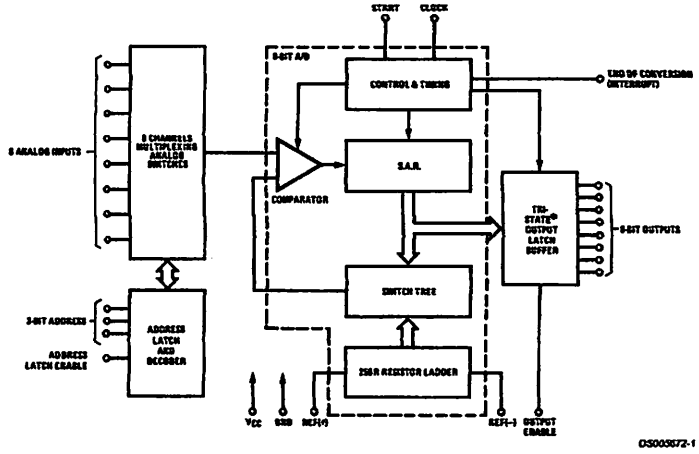
Features

- Easy interface to all microprocessors
- Operates ratiometrically or with 5 V_{DC} or analog span adjusted voltage reference
- No zero or full-scale adjust required
- 8-channel multiplexer with address logic
- 0V to 5V input range with single 5V power supply
- Outputs meet TTL voltage level specifications
- Standard hermetic or molded 28-pin DIP package
- 28-pin molded chip carrier package
- ADC0808 equivalent to MM74C949
- ADC0809 equivalent to MM74C949-1

Key Specifications

■ Resolution	8 Bits
■ Total Unadjusted Error	±½ LSB and ±1 LSB
■ Single Supply	5 V_{DC}
■ Low Power	15 mW
■ Conversion Time	100 µs

Block Diagram

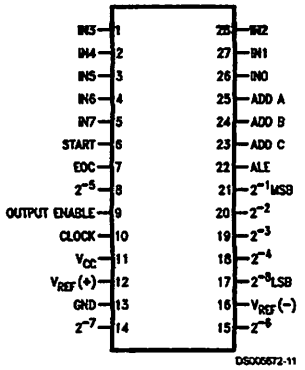


See Ordering Information

TRI-STATE® is a registered trademark of National Semiconductor Corp.

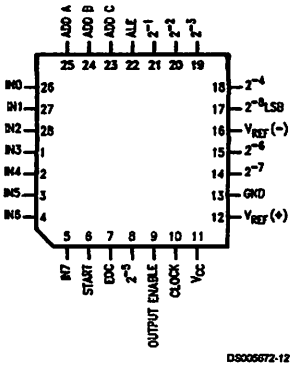
Connection Diagrams

Dual-In-Line Package



Order Number ADC0808CCN or ADC0809CCN
See NS Package J28A or N28A

Molded Chip Carrier Package



Order Number ADC0808CCV or ADC0809CCV
See NS Package V28A

Ordering Information

TEMPERATURE RANGE		−40°C to +85°C			−55°C to +125°C
Error	±½ LSB Unadjusted	ADC0808CCN	ADC0808CCV	ADC0808CCJ	ADC0808CJ
	±1 LSB Unadjusted	ADC0809CCN	ADC0809CCV		
Package Outline		N28A Molded DIP	V28A Molded Chip Carrier	J28A Ceramic DIP	J28A Ceramic DIP

Absolute Maximum Ratings (Notes 2, 1)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage (V_{CC}) (Note 3)	6.5V
Voltage at Any Pin Except Control Inputs	−0.3V to (V_{CC}+0.3V)
Voltage at Control Inputs (START, OE, CLOCK, ALE, ADD A, ADD B, ADD C)	−0.3V to +15V
Storage Temperature Range	−65°C to +150°C
Package Dissipation at T_A=25°C	875 mW
Lead Temp. (Soldering, 10 seconds)	
Dual-In-Line Package (plastic)	260°C
Dual-In-Line Package (ceramic)	300°C
Molded Chip Carrier Package	
Vapor Phase (60 seconds)	215°C
Infrared (15 seconds)	220°C
ESD Susceptibility (Note 8)	400V

Operating Conditions (Notes 1, 2)

Temperature Range (Note 1)	$T_{MIN} \le T_A \le T_{MAX}$
ADC0808CCN,ADC0809CCN	−40°C≤T_A≤+85°C
ADC0808CCV, ADC0809CCV	−40°C ≤ T_A ≤ +85°C
Range of V_{CC} (Note 1)	4.5 V_{DC} to 6.0 V_{DC}

Electrical Characteristics

Converter Specifications: V_{CC}=5 V_{DC}=V_{REF+}, $V_{REF(-)}$=GND, $T_{MIN} \le T_A \le T_{MAX}$ and f_{CLK}=640 kHz unless otherwise stated.

Symbol	Parameter	Conditions	Min	Typ	Max	Units
	ADC0808					
	Total Unadjusted Error	25°C			±½	LSB
	(Note 5)	T_{MIN} to T_{MAX}			±¾	LSB
	ADC0809					
	Total Unadjusted Error	0°C to 70°C			±1	LSB
	(Note 5)	T_{MIN} to T_{MAX}			±1¼	LSB
	Input Resistance	From Ref(+) to Ref(−)	1.0	2.5		kΩ
	Analog Input Voltage Range	(Note 4) V(+) or V(−)	GND−0.10		V_{CC}+0.10	V_{DC}
$V_{REF(+)}$	Voltage, Top of Ladder	Measured at Ref(+)		V_{CC}	V_{CC}+0.1	V
$\frac{V_{REF(+)}+V_{REF(-)}}{2}$	Voltage, Center of Ladder		V_{CC}/2-0.1	V_{CC}/2	V_{CC}/2+0.1	V
$V_{REF(-)}$	Voltage, Bottom of Ladder	Measured at Ref(−)	−0.1	0		V
I_{IN}	Comparator Input Current	f_c=640 kHz, (Note 6)	−2	±0.5	2	µA

Electrical Characteristics

Digital Levels and DC Specifications: ADC0808CCN, ADC0808CCV, ADC0809CCN and ADC0809CCV, 4.75≤V_{CC}≤5.25V, −40°C≤T_A≤+85°C unless otherwise noted

Symbol	Parameter	Conditions	Min	Typ	Max	Units
ANALOG MULTIPLEXER						
$I_{OFF(+)}$	OFF Channel Leakage Current	V_{CC}=5V, V_{IN}=5V,				
		T_A=25°C		10	200	nA
		T_{MIN} to T_{MAX}			1.0	µA
$I_{OFF(-)}$	OFF Channel Leakage Current	V_{CC}=5V, V_{IN}=0,				
		T_A=25°C	−200	−10		nA
		T_{MIN} to T_{MAX}	−1.0			µA
CONTROL INPUTS						
$V_{IN(1)}$	Logical "1" Input Voltage		V_{CC}−1.5			V
$V_{IN(0)}$	Logical "0" Input Voltage				1.5	V
$I_{IN(1)}$	Logical "1" Input Current (The Control Inputs)	V_{IN}=15V			1.0	µA
$I_{IN(0)}$	Logical "0" Input Current (The Control Inputs)	V_{IN}=0	−1.0			µA
I_{CC}	Supply Current	f_{CLK}=640 kHz		0.3	3.0	mA

Electrical Characteristics (Continued)

Digital Levels and DC Specifications: ADC0808CCN, ADC0808CCV, ADC0809CCN and ADC0809CCV, $4.75 \le V_{CC} \le 5.25V$, $-40^\circ C \le T_A \le +85^\circ C$ unless otherwise noted

Symbol	Parameter	Conditions	Min	Typ	Max	Units
DATA OUTPUTS AND EOC (INTERRUPT)						
$V_{OUT(1)}$	Logical "1" Output Voltage	V_{CC} = 4.75V I_{OUT} = −360µA I_{OUT} = −10µA		 2.4 4.5		 V(min) V(min)
$V_{OUT(0)}$	Logical "0" Output Voltage	I_O=1.6 mA			0.45	V
$V_{OUT(0)}$	Logical "0" Output Voltage EOC	I_O=1.2 mA			0.45	V
I_{OUT}	TRI-STATE Output Current	V_O=5V V_O=0	 −3		3	µA µA

Electrical Characteristics

Timing Specifications $V_{CC}=V_{REF(+)}=5V$, $V_{REF(-)}=GND$, $t_r=t_f=20$ ns and $T_A=25^\circ C$ unless otherwise noted.

Symbol	Parameter	Conditions	Min	Typ	Max	Units
t_{WS}	Minimum Start Pulse Width	(*Figure 5*)		100	200	ns
t_{WALE}	Minimum ALE Pulse Width	(*Figure 5*)		100	200	ns
t_s	Minimum Address Set-Up Time	(*Figure 5*)		25	50	ns
t_H	Minimum Address Hold Time	(*Figure 5*)		25	50	ns
t_D	Analog MUX Delay Time From ALE	R_S=0Ω (*Figure 5*)		1	2.5	µs
t_{H1}, t_{HD}	OE Control to Q Logic State	C_L=50 pF, R_L=10k (*Figure 8*)		125	250	ns
t_{1H}, t_{0H}	OE Control to Hi-Z	C_L=10 pF, R_L=10k (*Figure 8*)		125	250	ns
t_c	Conversion Time	f_c=640 kHz, (*Figure 5*) (Note 7)	90	100	116	µs
f_c	Clock Frequency		10	640	1280	kHz
t_{EOC}	EOC Delay Time	(*Figure 5*)	0		8+2 µS	Clock Periods
C_{IN}	Input Capacitance	At Control Inputs		10	15	pF
C_{OUT}	TRI-STATE Output Capacitance	At TRI-STATE Outputs		10	15	pF

Note 1: Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications do not apply when operating the device beyond its specified operating conditions.

Note 2: All voltages are measured with respect to GND, unless otherwise specified.

Note 3: A zener diode exists, internally, from V_{CC} to GND and has a typical breakdown voltage of 7 V_{DC}.

Note 4: Two on-chip diodes are tied to each analog input which will forward conduct for analog input voltages one diode drop below ground or one diode drop greater than the V_{CC} supply. The spec allows 100 mV forward bias of either diode. This means that as long as the analog V_{IN} does not exceed the supply voltage by more than 100 mV, the output code will be correct. To achieve an absolute $0V_{DC}$ to $5V_{DC}$ input voltage range will therefore require a minimum supply voltage of 4.900 V_{DC} over temperature variations, initial tolerance and loading.

Note 5: Total unadjusted error includes offset, full-scale, linearity, and multiplexer errors. See *Figure 3*. None of these A/Ds requires a zero or full-scale adjust. However, if an all zero code is desired for an analog input other than 0.0V, or if a narrow full-scale span exists (for example: 0.5V to 4.5V full-scale) the reference voltages can be adjusted to achieve this. See *Figure 13*.

Note 6: Comparator input current is a bias current into or out of the chopper stabilized comparator. The bias current varies directly with clock frequency and has little temperature dependence (*Figure 6*). See paragraph 4.0.

Note 7: The outputs of the data register are updated one clock cycle before the rising edge of EOC.

Note 8: Human body model, 100 pF discharged through a 1.5 kΩ resistor.

Functional Description

Multiplexer. The device contains an 8-channel single-ended analog signal multiplexer. A particular input channel is selected by using the address decoder. *Table 1* shows the input states for the address lines to select any channel. The address is latched into the decoder on the low-to-high transition of the address latch enable signal.

TABLE 1.

SELECTED ANALOG CHANNEL	ADDRESS LINE		
	C	B	A
IN0	L	L	L
IN1	L	L	H
IN2	L	H	L
IN3	L	H	H
IN4	H	L	L
IN5	H	L	H
IN6	H	H	L
IN7	H	H	H

CONVERTER CHARACTERISTICS

The Converter

The heart of this single chip data acquisition system is its 8-bit analog-to-digital converter. The converter is designed to give fast, accurate, and repeatable conversions over a wide range of temperatures. The converter is partitioned into 3 major sections: the 256R ladder network, the successive approximation register, and the comparator. The converter's digital outputs are positive true.

The 256R ladder network approach (*Figure 1*) was chosen over the conventional R/2R ladder because of its inherent monotonicity, which guarantees no missing digital codes. Monotonicity is particularly important in closed loop feedback control systems. A non-monotonic relationship can cause oscillations that will be catastrophic for the system. Additionally, the 256R network does not cause load variations on the reference voltage.

The bottom resistor and the top resistor of the ladder network in *Figure 1* are not the same value as the remainder of the network. The difference in these resistors causes the output characteristic to be symmetrical with the zero and full-scale points of the transfer curve. The first output transition occurs when the analog signal has reached +½ LSB and succeeding output transitions occur every 1 LSB later up to full-scale.

The successive approximation register (SAR) performs 8 iterations to approximate the input voltage. For any SAR type converter, n-iterations are required for an n-bit converter. *Figure 2* shows a typical example of a 3-bit converter. In the ADC0808, ADC0809, the approximation technique is extended to 8 bits using the 256R network.

The A/D converter's successive approximation register (SAR) is reset on the positive edge of the start conversion (SC) pulse. The conversion is begun on the falling edge of the start conversion pulse. A conversion in process will be interrupted by receipt of a new start conversion pulse. Continuous conversion may be accomplished by tying the end-of-conversion (EOC) output to the SC input. If used in this mode, an external start conversion pulse should be applied after power up. End-of-conversion will go low between 0 and 8 clock pulses after the rising edge of start conversion.

The most important section of the A/D converter is the comparator. It is this section which is responsible for the ultimate accuracy of the entire converter. It is also the comparator drift which has the greatest influence on the repeatability of the device. A chopper-stabilized comparator provides the most effective method of satisfying all the converter requirements.

The chopper-stabilized comparator converts the DC input signal into an AC signal. This signal is then fed through a high gain AC amplifier and has the DC level restored. This technique limits the drift component of the amplifier since the drift is a DC component which is not passed by the AC amplifier. This makes the entire A/D converter extremely insensitive to temperature, long term drift and input offset errors.

Figure 4 shows a typical error curve for the ADC0808 as measured using the procedures outlined in AN-179.

Functional Description (Continued)

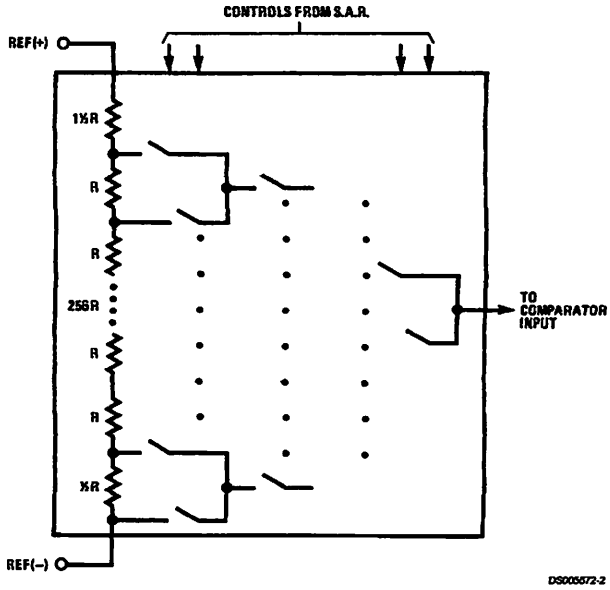


FIGURE 1. Resistor Ladder and Switch Tree

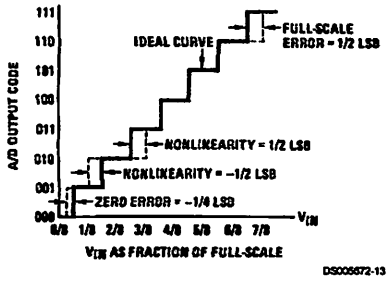


FIGURE 2. 3-Bit A/D Transfer Curve

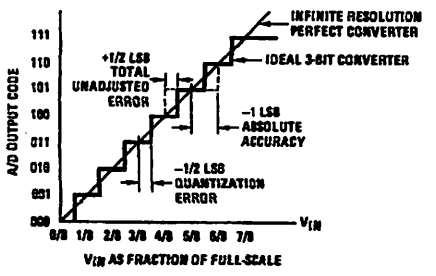


FIGURE 3. 3-Bit A/D Absolute Accuracy Curve

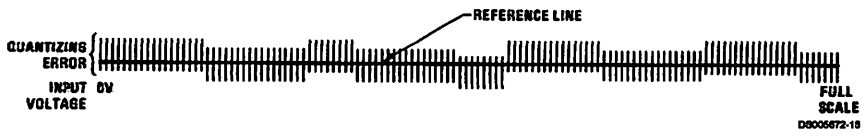


FIGURE 4. Typical Error Curve

Timing Diagram

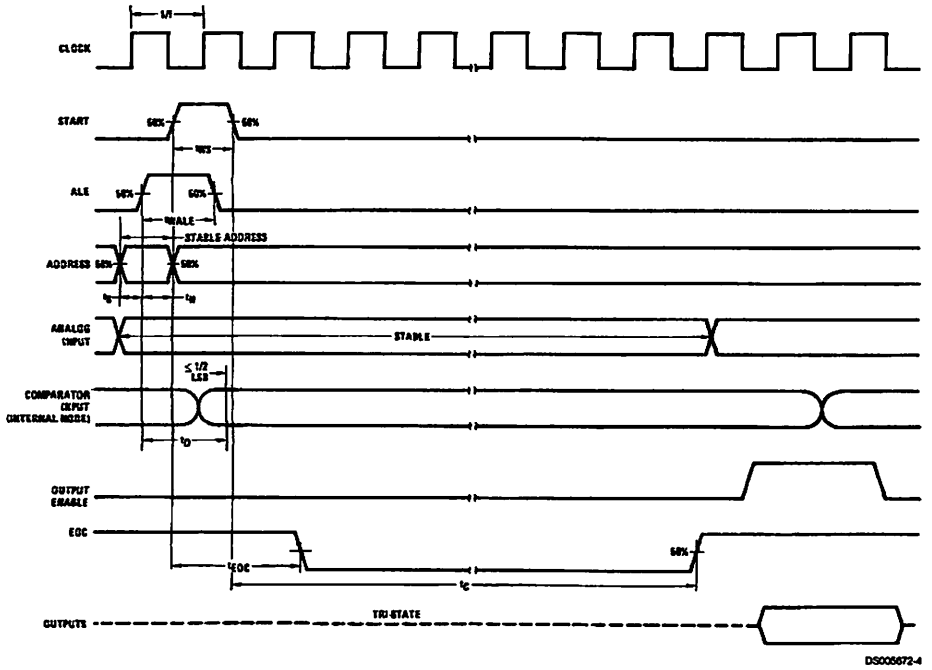


FIGURE 5.

Typical Performance Characteristics

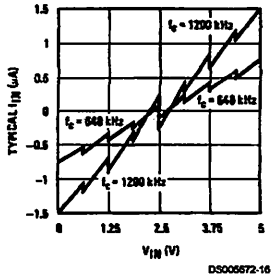


FIGURE 6. Comparator I_{IN} vs V_{IN} (V_{CC}=V_{REF}=5V)

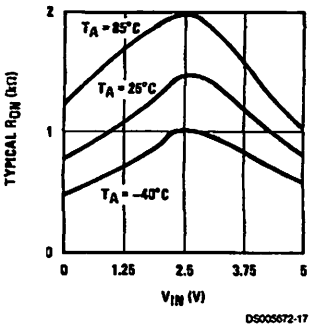


FIGURE 7. Multiplexer R_{ON} vs V_{IN} (V_{CC}=V_{REF}=5V)

TRI-STATE Test Circuits and Timing Diagrams

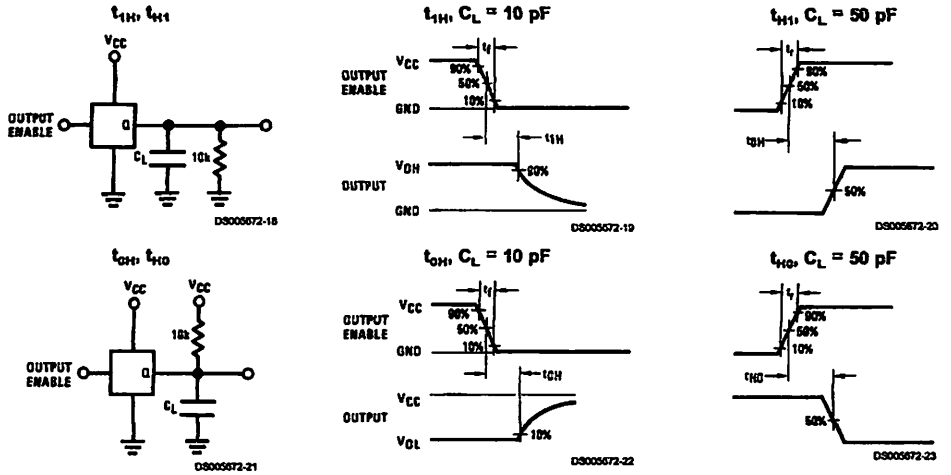


FIGURE 8.

Applications Information

OPERATION

1.0 RATIOMETRIC CONVERSION

The ADC0808, ADC0809 is designed as a complete Data Acquisition System (DAS) for ratiometric conversion systems. In ratiometric systems, the physical variable being measured is expressed as a percentage of full-scale which is not necessarily related to an absolute standard. The voltage input to the ADC0808 is expressed by the equation

$$\frac{V_{IN}}{V_{fs}-V_Z}=\frac{D_X}{D_{MAX}-D_{MIN}} \qquad (1)$$

V_{IN}=Input voltage into the ADC0808

V_{fs}=Full-scale voltage

V_Z=Zero voltage

D_X=Data point being measured

D_{MAX}=Maximum data limit

D_{MIN}=Minimum data limit

A good example of a ratiometric transducer is a potentiometer used as a position sensor. The position of the wiper is directly proportional to the output voltage which is a ratio of the full-scale voltage across it. Since the data is represented as a proportion of full-scale, reference requirements are greatly reduced, eliminating a large source of error and cost for many applications. A major advantage of the ADC0808, ADC0809 is that the input voltage range is equal to the supply range so the transducers can be connected directly across the supply and their outputs connected directly into the multiplexer inputs, (*Figure 9*).

Ratiometric transducers such as potentiometers, strain gauges, thermistor bridges, pressure transducers, etc., are suitable for measuring proportional relationships; however, many types of measurements must be referred to an absolute standard such as voltage or current. This means a sys-

Applications Information (Continued)

tem reference must be used which relates the full-scale voltage to the standard volt. For example, if $V_{CC}=V_{REF}=5.12V$, then the full-scale range is divided into 256 standard steps. The smallest standard step is 1 LSB which is then 20 mV.

2.0 RESISTOR LADDER LIMITATIONS

The voltages from the resistor ladder are compared to the selected into 8 times in a conversion. These voltages are coupled to the comparator via an analog switch tree which is referenced to the supply. The voltages at the top, center and bottom of the ladder must be controlled to maintain proper operation.

The top of the ladder, Ref(+), should not be more positive than the supply, and the bottom of the ladder, Ref(−), should not be more negative than ground. The center of the ladder voltage must also be near the center of the supply because the analog switch tree changes from N-channel switches to P-channel switches. These limitations are automatically satisfied in ratiometric systems and can be easily met in ground referenced systems.

Figure 10 shows a ground referenced system with a separate supply and reference. In this system, the supply must be trimmed to match the reference voltage. For instance, if a 5.12V is used, the supply should be adjusted to the same voltage within 0.1V.

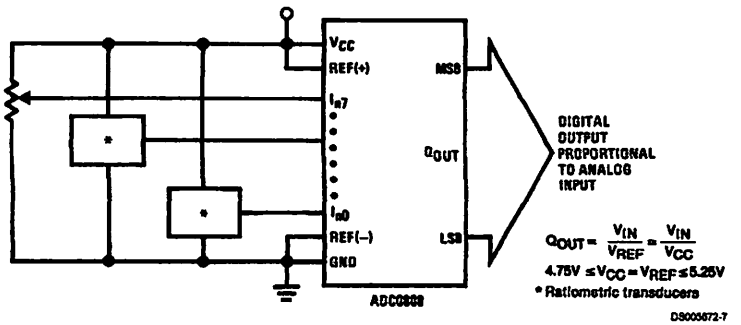
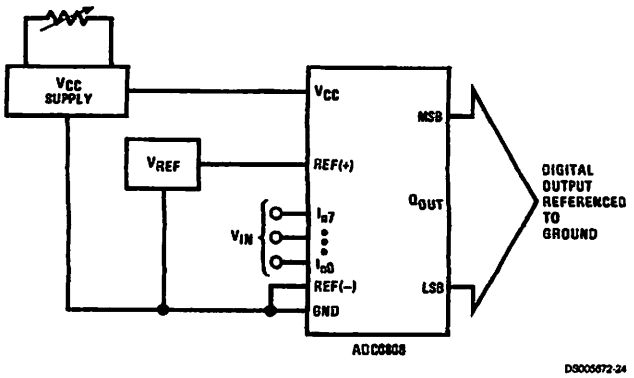


FIGURE 9. Ratiometric Conversion System

The ADC0808 needs less than a milliamp of supply current so developing the supply from the reference is readily accomplished. In *Figure 11* a ground referenced system is shown which generates the supply from the reference. The buffer shown can be an op amp of sufficient drive to supply the milliamp of supply current and the desired bus drive, or if a capacitive bus is driven by the outputs a large capacitor will supply the transient supply current as seen in *Figure 12*. The LM301 is overcompensated to insure stability when loaded by the 10 µF output capacitor.

The top and bottom ladder voltages cannot exceed V_{CC} and ground, respectively, but they can be symmetrically less than V_{CC} and greater than ground. The center of the ladder voltage should always be near the center of the supply. The sensitivity of the converter can be increased, (i.e., size of the LSB steps decreased) by using a symmetrical reference system. In *Figure 13*, a 2.5V reference is symmetrically centered about $V_{CC}/2$ since the same current flows in identical resistors. This system with a 2.5V reference allows the LSB bit to be half the size of a 5V reference system.

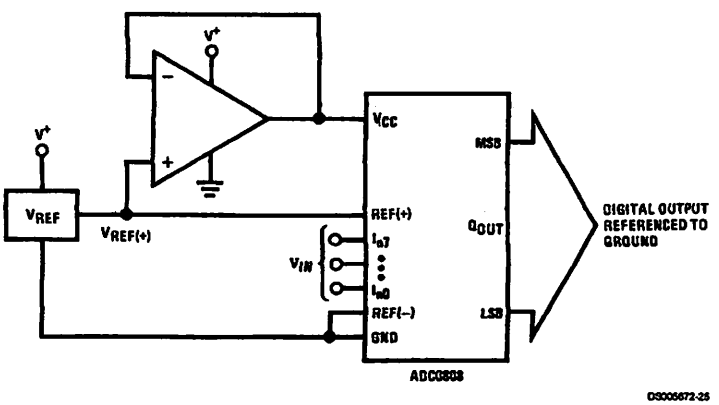
Applications Information (Continued)



$$Q_{OUT} = \frac{V_{IN}}{V_{REF}}$$

$4.75V \leq V_{CC} = V_{REF} \leq 5.25V$

FIGURE 10. Ground Referenced Conversion System Using Trimmed Supply



$$Q_{OUT} = \frac{V_{IN}}{V_{REF}}$$

$4.75V \leq V_{CC} = V_{REF} \leq 5.25V$

FIGURE 11. Ground Referenced Conversion System with Reference Generating V_{CC} Supply

Applications Information (Continued)

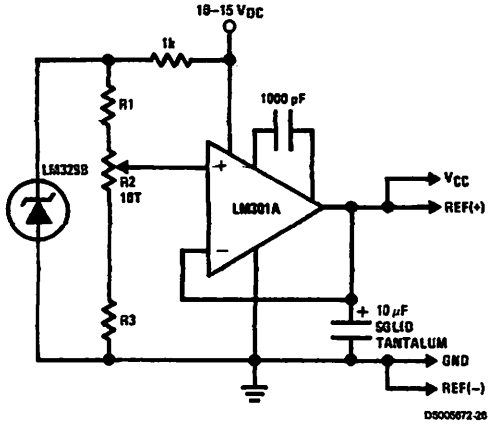
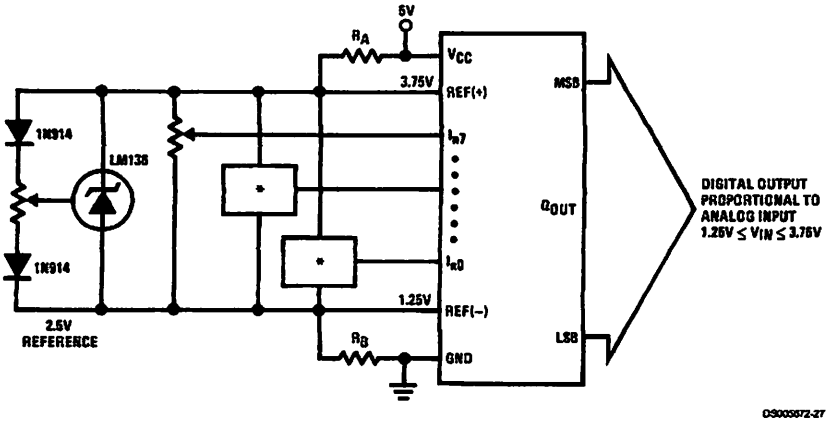


FIGURE 12. Typical Reference and Supply Circuit



$R_A = R_B$

*Ratiometric transducers

FIGURE 13. Symmetrically Centered Reference

3.0 CONVERTER EQUATIONS

The transition between adjacent codes N and N+1 is given by:

$$V_{IN} = \left\{ (V_{REF(+)} - V_{REF(-)}) \left[\frac{N}{256} + \frac{1}{512} \right] \pm V_{TUE} \right\} + V_{REF(-)} \quad (2)$$

The center of an output code N is given by:

$$V_{IN} \left\{ (V_{REF(+)} - V_{REF(-)}) \left[\frac{N}{256} \right] \pm V_{TUE} \right\} + V_{REF(-)} \quad (3)$$

The output code N for an arbitrary input are the integers within the range:

$$N = \frac{V_{IN} - V_{REF(-)}}{V_{REF(+)} - V_{REF(-)}} \times 256 \pm \text{Absolute Accuracy} \quad (4)$$

Where: V_{IN}=Voltage at comparator input
$V_{REF(+)}$=Voltage at Ref(+)
$V_{REF(-)}$=Voltage at Ref(−)
V_{TUE}=Total unadjusted error voltage (typically $V_{REF(+)}$+512)

Applications Information (Continued)

4.0 ANALOG COMPARATOR INPUTS

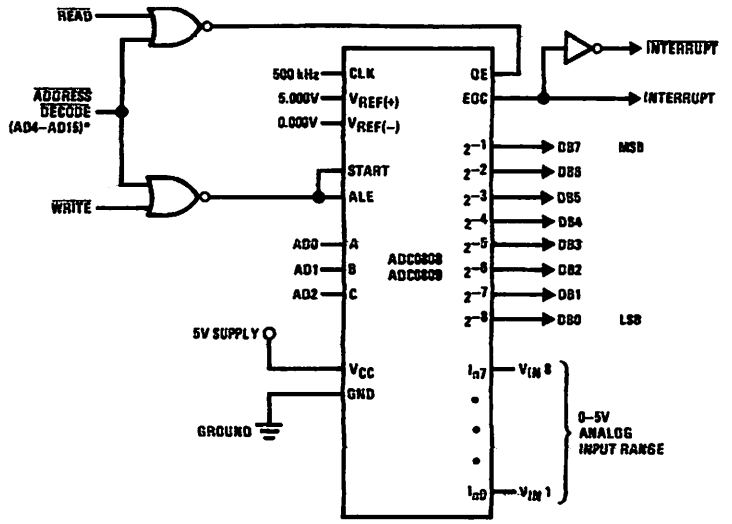
The dynamic comparator input current is caused by the periodic switching of on-chip stray capacitances. These are connected alternately to the output of the resistor ladder/switch tree network and to the comparator input as part of the operation of the chopper stabilized comparator.

The average value of the comparator input current varies directly with clock frequency and with V_{IN} as shown in *Figure 6*.

If no filter capacitors are used at the analog inputs and the signal source impedances are low, the comparator input current should not introduce converter errors, as the transient created by the capacitance discharge will die out before the comparator output is strobed.

If input filter capacitors are desired for noise reduction and signal conditioning they will tend to average out the dynamic comparator input current. It will then take on the characteristics of a DC bias current whose effect can be predicted conventionally.

Typical Application

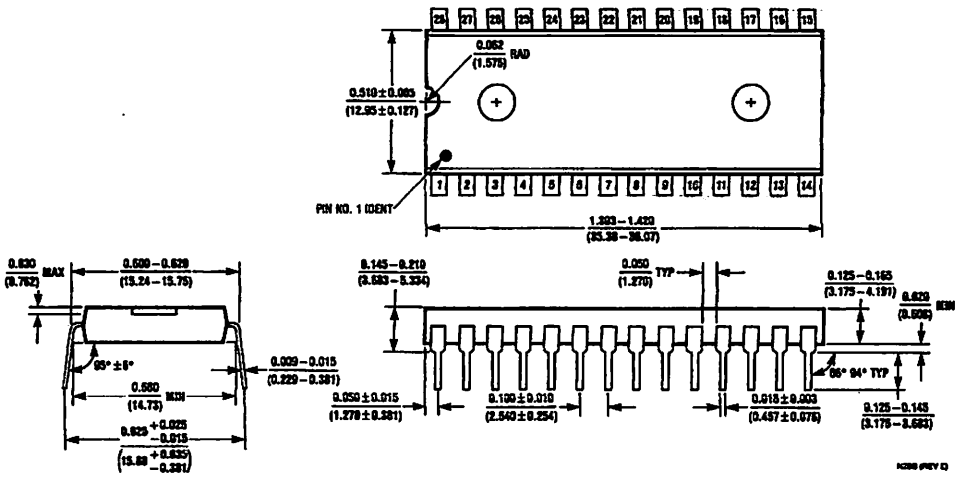


*Address latches needed for 8085 and SC/MP interfacing the ADC0808 to a microprocessor

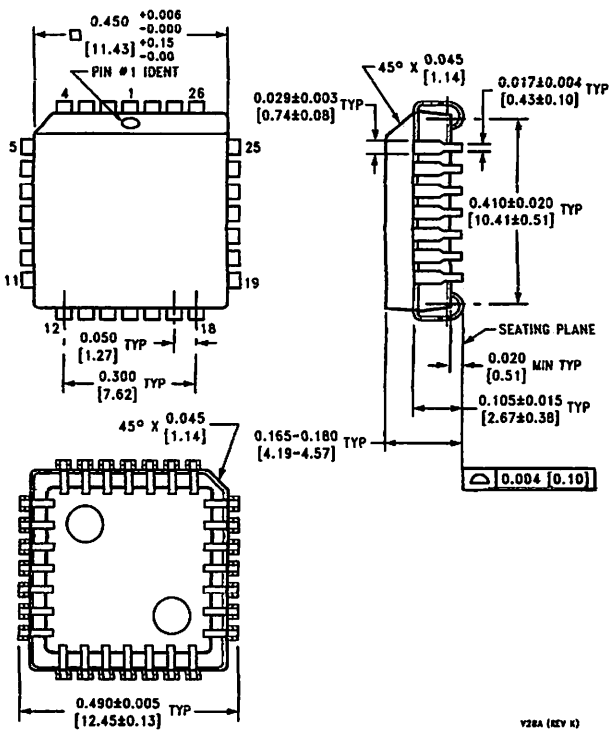
TABLE 2. Microprocessor Interface Table

PROCESSOR	READ	WRITE	INTERRUPT (COMMENT)
8080	MEMR	MEMW	INTR (Thru RST Circuit)
8085	RD	WR	INTR (Thru RST Circuit)
Z-80	RD	WR	INT (Thru RST Circuit, Mode 0)
SC/MP	NRDS	NWDS	SA (Thru Sense A)
6800	VMA•φ2•R/W	VMA•φ•R/W	IRQA or IRQB (Thru PIA)

Physical Dimensions inches (millimeters) unless otherwise noted



Molded Dual-In-Line Package (N)
Order Number ADC0808CCN or ADC0809CCN
NS Package Number N28B



Molded Chip Carrier (V)
Order Number ADC0808CCV or ADC0809CCV
NS Package Number V28A

Notes

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor
Corporation
Americas
Tel: 1-800-272-9959
Fax: 1-800-737-7018
Email: support@nsc.com
www.national.com

National Semiconductor
Europe
Fax: +49 (0) 1 80-530 85 86
Email: europe.support@nsc.com
Deutsch Tel: +49 (0) 1 80-530 85 85
English Tel: +49 (0) 1 80-532 78 32
Français Tel: +49 (0) 1 80-532 93 58
Italiano Tel: +49 (0) 1 80-534 16 80

National Semiconductor
Asia Pacific Customer
Response Group
Tel: 65-2544466
Fax: 65-2504466
Email: sea.support@nsc.com

National Semiconductor
Japan Ltd.
Tel: 81-3-5639-7560
Fax: 81-3-5639-7507

eatures

- Compatible with MCS®-51 Products
- 8K Bytes of In-System Programmable (ISP) Flash Memory
  - Endurance: 10,000 Write/Erase Cycles
- 4.0V to 5.5V Operating Range
- Fully Static Operation: 0 Hz to 33 MHz
- Three-level Program Memory Lock
- 256 x 8-bit Internal RAM
- 32 Programmable I/O Lines
- Three 16-bit Timer/Counters
- Eight Interrupt Sources
- Full Duplex UART Serial Channel
- Low-power Idle and Power-down Modes
- Interrupt Recovery from Power-down Mode
- Watchdog Timer
- Dual Data Pointer
- Power-off Flag
- Fast Programming Time
- Flexible ISP Programming (Byte and Page Mode)
- Green (Pb/Halide-free) Packaging Option

Description

e AT89S52 is a low-power, high-performance CMOS 8-bit microcontroller with 8K tes of in-system programmable Flash memory. The device is manufactured using mel's high-density nonvolatile memory technology and is compatible with the indus--standard 80C51 instruction set and pinout. The on-chip Flash allows the program emory to be reprogrammed in-system or by a conventional nonvolatile memory pro-ammer. By combining a versatile 8-bit CPU with in-system programmable Flash on nonolithic chip, the Atmel AT89S52 is a powerful microcontroller which provides a ghly-flexible and cost-effective solution to many embedded control applications.

e AT89S52 provides the following standard features: 8K bytes of Flash, 256 bytes RAM, 32 I/O lines, Watchdog timer, two data pointers, three 16-bit timer/counters, a -vector two-level interrupt architecture, a full duplex serial port, on-chip oscillator, d clock circuitry. In addition, the AT89S52 is designed with static logic for operation wn to zero frequency and supports two software selectable power saving modes. e Idle Mode stops the CPU while allowing the RAM, timer/counters, serial port, and errupt system to continue functioning. The Power-down mode saves the RAM con-nts but freezes the oscillator, disabling all other chip functions until the next interrupt hardware reset.

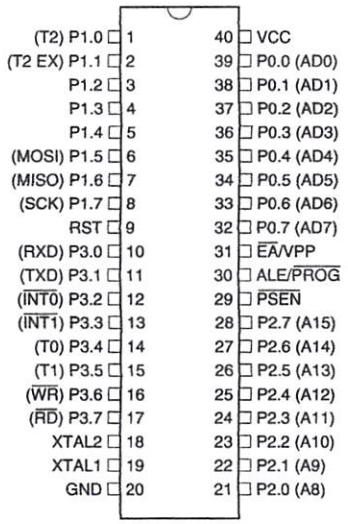
8-bit Microcontroller with 8K Bytes In-System Programmable Flash

AT89S52

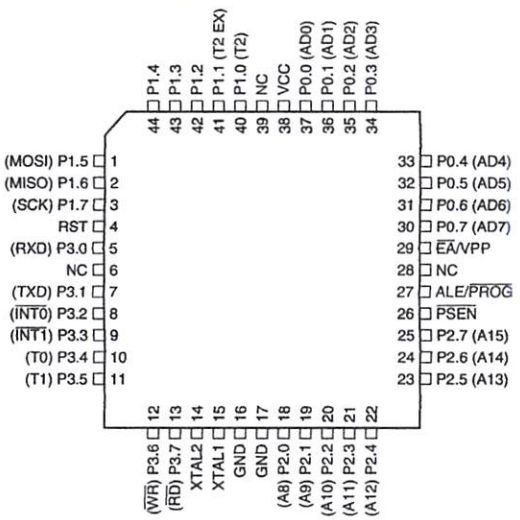
1919D–MICRO–6/08

Pin Configurations

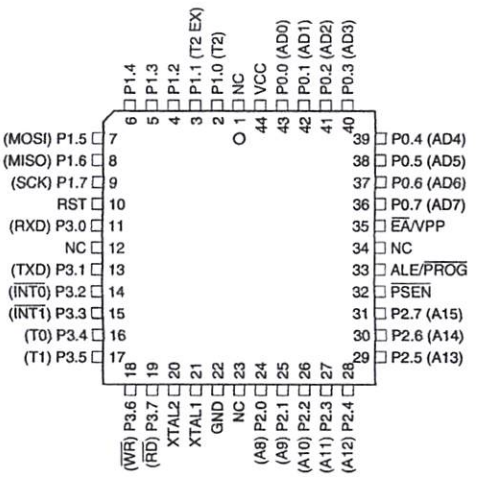
1 40-lead PDIP



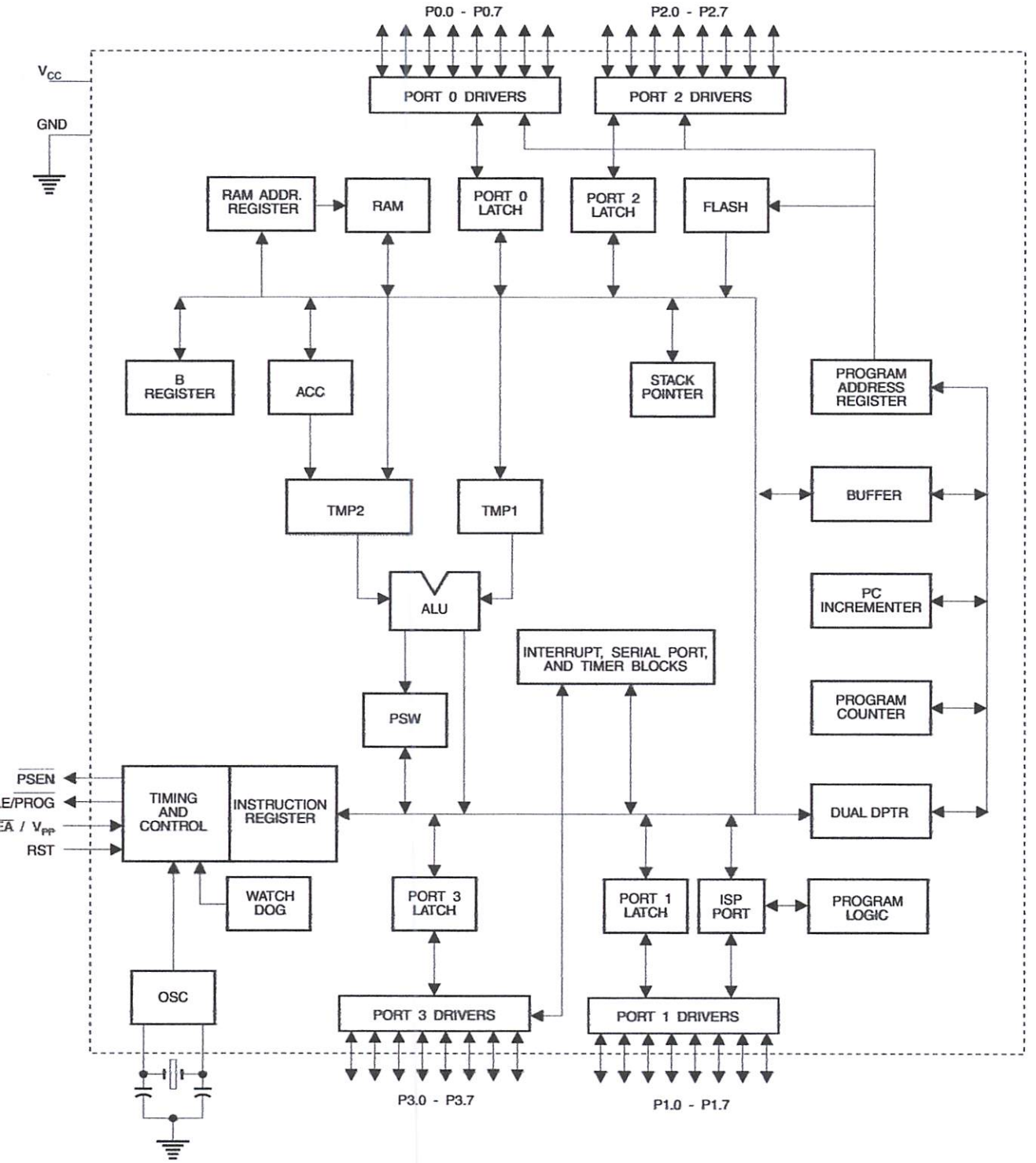
2 44-lead TQFP



2.3 44-lead PLCC



Block Diagram



Pin Description

VCC

Supply voltage.

GND

Ground.

Port 0

Port 0 is an 8-bit open drain bidirectional I/O port. As an output port, each pin can sink eight TTL inputs. When 1s are written to port 0 pins, the pins can be used as high-impedance inputs.

Port 0 can also be configured to be the multiplexed low-order address/data bus during accesses to external program and data memory. In this mode, P0 has internal pull-ups.

Port 0 also receives the code bytes during Flash programming and outputs the code bytes during program verification. **External pull-ups are required during program verification.**

Port 1

Port 1 is an 8-bit bidirectional I/O port with internal pull-ups. The Port 1 output buffers can sink/source four TTL inputs. When 1s are written to Port 1 pins, they are pulled high by the internal pull-ups and can be used as inputs. As inputs, Port 1 pins that are externally being pulled low will source current (I_{IL}) because of the internal pull-ups.

In addition, P1.0 and P1.1 can be configured to be the timer/counter 2 external count input (P1.0/T2) and the timer/counter 2 trigger input (P1.1/T2EX), respectively, as shown in the following table.

Port 1 also receives the low-order address bytes during Flash programming and verification.

Port Pin	Alternate Functions
P1.0	T2 (external count input to Timer/Counter 2), clock-out
P1.1	T2EX (Timer/Counter 2 capture/reload trigger and direction control)
P1.5	MOSI (used for In-System Programming)
P1.6	MISO (used for In-System Programming)
P1.7	SCK (used for In-System Programming)

Port 2

Port 2 is an 8-bit bidirectional I/O port with internal pull-ups. The Port 2 output buffers can sink/source four TTL inputs. When 1s are written to Port 2 pins, they are pulled high by the internal pull-ups and can be used as inputs. As inputs, Port 2 pins that are externally being pulled low will source current (I_{IL}) because of the internal pull-ups.

Port 2 emits the high-order address byte during fetches from external program memory and during accesses to external data memory that use 16-bit addresses (MOVX @ DPTR). In this application, Port 2 uses strong internal pull-ups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOVX @ RI), Port 2 emits the contents of the P2 Special Function Register.

Port 2 also receives the high-order address bits and some control signals during Flash programming and verification.

Port 3

Port 3 is an 8-bit bidirectional I/O port with internal pull-ups. The Port 3 output buffers can sink/source four TTL inputs. When 1s are written to Port 3 pins, they are pulled high by the internal pull-ups and can be used as inputs. As inputs, Port 3 pins that are externally being pulled low will source current (I_{IL}) because of the pull-ups.

Port 3 receives some control signals for Flash programming and verification.

Port 3 also serves the functions of various special features of the AT89S52, as shown in the following table.

Port Pin	Alternate Functions
P3.0	RXD (serial input port)
P3.1	TXD (serial output port)
P3.2	$\overline{INT0}$ (external interrupt 0)
P3.3	$\overline{INT1}$ (external interrupt 1)
P3.4	T0 (timer 0 external input)
P3.5	T1 (timer 1 external input)
P3.6	$\overline{WR}$ (external data memory write strobe)
P3.7	$\overline{RD}$ (external data memory read strobe)

RST

Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device. This pin drives high for 98 oscillator periods after the Watchdog times out. The DISRTO bit in SFR AUXR (address 8EH) can be used to disable this feature. In the default state of bit DISRTO, the RESET HIGH out feature is enabled.

ALE/$\overline{PROG}$

Address Latch Enable (ALE) is an output pulse for latching the low byte of the address during accesses to external memory. This pin is also the program pulse input ($\overline{PROG}$) during Flash programming.

In normal operation, ALE is emitted at a constant rate of 1/6 the oscillator frequency and may be used for external timing or clocking purposes. Note, however, that one ALE pulse is skipped during each access to external data memory.

If desired, ALE operation can be disabled by setting bit 0 of SFR location 8EH. With the bit set, ALE is active only during a MOVX or MOVC instruction. Otherwise, the pin is weakly pulled high. Setting the ALE-disable bit has no effect if the microcontroller is in external execution mode.

9 PSEN

Program Store Enable ($\overline{PSEN}$) is the read strobe to external program memory.

When the AT89S52 is executing code from external program memory, $\overline{PSEN}$ is activated twice each machine cycle, except that two $\overline{PSEN}$ activations are skipped during each access to external data memory.

10 EA/VPP

External Access Enable. $\overline{EA}$ must be strapped to GND in order to enable the device to fetch code from external program memory locations starting at 0000H up to FFFFH. Note, however, that if lock bit 1 is programmed, $\overline{EA}$ will be internally latched on reset.

$\overline{EA}$ should be strapped to V_{CC} for internal program executions.

This pin also receives the 12-volt programming enable voltage (V_{PP}) during Flash programming.

11 XTAL1

Input to the inverting oscillator amplifier and input to the internal clock operating circuit.

12 XTAL2

Output from the inverting oscillator amplifier.

Special Function Registers

A map of the on-chip memory area called the Special Function Register (SFR) space is shown in Table 5-1.

Note that not all of the addresses are occupied, and unoccupied addresses may not be implemented on the chip. Read accesses to these addresses will in general return random data, and write accesses will have an indeterminate effect.

User software should not write 1s to these unlisted locations, since they may be used in future products to invoke new features. In that case, the reset or inactive values of the new bits will always be 0.

Timer 2 Registers: Control and status bits are contained in registers T2CON (shown in Table 5-2) and T2MOD (shown in Table 10-2) for Timer 2. The register pair (RCAP2H, RCAP2L) are the Capture/Reload registers for Timer 2 in 16-bit capture mode or 16-bit auto-reload mode.

Interrupt Registers: The individual interrupt enable bits are in the IE register. Two priorities can be set for each of the six interrupt sources in the IP register.

ble 5-1. AT89S52 SFR Map and Reset Values

0F8H								0FFH
0F0H	B 00000000							0F7H
0E8H								0EFH
0E0H	ACC 00000000							0E7H
0D8H								0DFH
0D0H	PSW 00000000							0D7H
0C8H	T2CON 00000000	T2MOD XXXXXX00	RCAP2L 00000000	RCAP2H 00000000	TL2 00000000	TH2 00000000		0CFH
0C0H								0C7H
0B8H	IP XX000000							0BFH
0B0H	P3 11111111							0B7H
0A8H	IE 0X000000							0AFH
0A0H	P2 11111111		AUXR1 XXXXXXX0				WDTRST XXXXXXXX	0A7H
98H	SCON 00000000	SBUF XXXXXXXX						9FH
90H	P1 11111111							97H
88H	TCON 00000000	TMOD 00000000	TL0 00000000	TL1 00000000	TH0 00000000	TH1 00000000	AUXR XXX00XX0	8FH
80H	P0 11111111	SP 00000111	DP0L 00000000	DP0H 00000000	DP1L 00000000	DP1H 00000000		PCON 0XXX0000 87H

ble 5-2. T2CON – Timer/Counter 2 Control Register

T2CON Address = 0C8H — Reset Value = 0000 0000B

Bit Addressable

Bit	TF2	EXF2	RCLK	TCLK	EXEN2	TR2	C/$\overline{T2}$	CP/$\overline{RL2}$
	7	6	5	4	3	2	1	0

ymbol	Function
F2	Timer 2 overflow flag set by a Timer 2 overflow and must be cleared by software. TF2 will not be set when either RCLK = 1 or TCLK = 1.
XF2	Timer 2 external flag set when either a capture or reload is caused by a negative transition on T2EX and EXEN2 = 1. When Timer 2 interrupt is enabled, EXF2 = 1 will cause the CPU to vector to the Timer 2 interrupt routine. EXF2 must be cleared by software. EXF2 does not cause an interrupt in up/down counter mode (DCEN = 1).
CLK	Receive clock enable. When set, causes the serial port to use Timer 2 overflow pulses for its receive clock in serial port Modes 1 and 3. RCLK = 0 causes Timer 1 overflow to be used for the receive clock.
CLK	Transmit clock enable. When set, causes the serial port to use Timer 2 overflow pulses for its transmit clock in serial port Modes 1 and 3. TCLK = 0 causes Timer 1 overflows to be used for the transmit clock.
XEN2	Timer 2 external enable. When set, allows a capture or reload to occur as a result of a negative transition on T2EX if Timer 2 is not being used to clock the serial port. EXEN2 = 0 causes Timer 2 to ignore events at T2EX.
R2	Start/Stop control for Timer 2. TR2 = 1 starts the timer.
/$\overline{T2}$	Timer or counter select for Timer 2. C/$\overline{T2}$ = 0 for timer function. C/$\overline{T2}$ = 1 for external event counter (falling edge triggered).
P/$\overline{RL2}$	Capture/Reload select. CP/$\overline{RL2}$ = 1 causes captures to occur on negative transitions at T2EX if EXEN2 = 1. CP/$\overline{RL2}$ = 0 causes automatic reloads to occur when Timer 2 overflows or negative transitions occur at T2EX when EXEN2 = 1. When either RCLK or TCLK = 1, this bit is ignored and the timer is forced to auto-reload on Timer 2 overflow.

le 5-3. AUXR: Auxiliary Register

XR Address = 8EH Reset Value = XXX00XX0B

Not Bit Addressable

	–	–	–	WDIDLE	DISRTO	–	–	DISALE
Bit	7	6	5	4	3	2	1	0

Reserved for future expansion

SALE Disable/Enable ALE

DISALE	Operating Mode
0	ALE is emitted at a constant rate of 1/6 the oscillator frequency
1	ALE is active only during a MOVX or MOVC instruction

SRTO Disable/Enable Reset out

DISRTO	
0	Reset pin is driven High after WDT times out
1	Reset pin is input only

DIDLE Disable/Enable WDT in IDLE mode

WDIDLE	
0	WDT continues to count in IDLE mode
1	WDT halts counting in IDLE mode

al Data Pointer Registers: To facilitate accessing both internal and external data memory, two banks of 16-bit Data nter Registers are provided: DP0 at SFR address locations 82H-83H and DP1 at 84H-85H. Bit DPS = 0 in SFR AUXR1 ects DP0 and DPS = 1 selects DP1. The user should **ALWAYS** initialize the DPS bit to the appropriate value before essing the respective Data Pointer Register.

wer Off Flag: The Power Off Flag (POF) is located at bit 4 (PCON.4) in the PCON SFR. POF is set to "1" during power It can be set and rest under software control and is not affected by reset.

le 5-4. AUXR1: Auxiliary Register 1

UXR1 Address = A2H Reset Value = XXXXXXX0B

Not Bit Addressable

	–	–	–	–	–	–	–	DPS
Bit	7	6	5	4	3	2	1	0

Reserved for future expansion

PS Data Pointer Register Select

DPS	
0	Selects DPTR Registers DP0L, DP0H
1	Selects DPTR Registers DP1L, DP1H

Memory Organization

MCS-51 devices have a separate address space for Program and Data Memory. Up to 64K bytes each of external Program and Data Memory can be addressed.

Program Memory

If the $\overline{EA}$ pin is connected to GND, all program fetches are directed to external memory.

On the AT89S52, if $\overline{EA}$ is connected to V_{CC}, program fetches to addresses 0000H through 1FFFH are directed to internal memory and fetches to addresses 2000H through FFFFH are to external memory.

Data Memory

The AT89S52 implements 256 bytes of on-chip RAM. The upper 128 bytes occupy a parallel address space to the Special Function Registers. This means that the upper 128 bytes have the same addresses as the SFR space but are physically separate from SFR space.

When an instruction accesses an internal location above address 7FH, the address mode used in the instruction specifies whether the CPU accesses the upper 128 bytes of RAM or the SFR space. Instructions which use direct addressing access the SFR space.

For example, the following direct addressing instruction accesses the SFR at location 0A0H (which is P2).

```
MOV 0A0H, #data
```

Instructions that use indirect addressing access the upper 128 bytes of RAM. For example, the following indirect addressing instruction, where R0 contains 0A0H, accesses the data byte at address 0A0H, rather than P2 (whose address is 0A0H).

```
MOV @R0, #data
```

Note that stack operations are examples of indirect addressing, so the upper 128 bytes of data RAM are available as stack space.

Watchdog Timer (One-time Enabled with Reset-out)

The WDT is intended as a recovery method in situations where the CPU may be subjected to software upsets. The WDT consists of a 14-bit counter and the Watchdog Timer Reset (WDTRST) SFR. The WDT is defaulted to disable from exiting reset. To enable the WDT, a user must write 01EH and 0E1H in sequence to the WDTRST register (SFR location 0A6H). When the WDT is enabled, it will increment every machine cycle while the oscillator is running. The WDT timeout period is dependent on the external clock frequency. There is no way to disable the WDT except through reset (either hardware reset or WDT overflow reset). When WDT overflows, it will drive an output RESET HIGH pulse at the RST pin.

Using the WDT

To enable the WDT, a user must write 01EH and 0E1H in sequence to the WDTRST register (SFR location 0A6H). When the WDT is enabled, the user needs to service it by writing 01EH and 0E1H to WDTRST to avoid a WDT overflow. The 14-bit counter overflows when it reaches 16383 (3FFFH), and this will reset the device. When the WDT is enabled, it will increment every machine cycle while the oscillator is running. This means the user must reset the WDT at least every 16383 machine cycles. To reset the WDT the user must write 01EH and 0E1H to WDTRST. WDTRST is a write-only register. The WDT counter cannot be read or written. When

WDT overflows, it will generate an output RESET pulse at the RST pin. The RESET pulse duration is 98xTOSC, where TOSC = 1/FOSC. To make the best use of the WDT, it should be serviced in those sections of code that will periodically be executed within the time required to prevent a WDT reset.

WDT During Power-down and Idle

In Power-down mode the oscillator stops, which means the WDT also stops. While in Power-down mode, the user does not need to service the WDT. There are two methods of exiting Power-down mode: by a hardware reset or via a level-activated external interrupt which is enabled prior to entering Power-down mode. When Power-down is exited with hardware reset, servicing the WDT should occur as it normally does whenever the AT89S52 is reset. Exiting Power-down with an interrupt is significantly different. The interrupt is held low long enough for the oscillator to stabilize. When the interrupt is brought high, the interrupt is serviced. To prevent the WDT from resetting the device while the interrupt pin is held low, the WDT is not started until the interrupt is pulled high. It is suggested that the WDT be reset during the interrupt service for the interrupt used to exit Power-down mode.

To ensure that the WDT does not overflow within a few states of exiting Power-down, it is best to reset the WDT just before entering Power-down mode.

Before going into the IDLE mode, the WDIDLE bit in SFR AUXR is used to determine whether the WDT continues to count if enabled. The WDT keeps counting during IDLE (WDIDLE bit = 0) as the default state. To prevent the WDT from resetting the AT89S52 while in IDLE mode, the user should always set up a timer that will periodically exit IDLE, service the WDT, and reenter IDLE mode.

With WDIDLE bit enabled, the WDT will stop to count in IDLE mode and resumes the count upon exit from IDLE.

UART

The UART in the AT89S52 operates the same way as the UART in the AT89C51 and AT89C52. For further information on the UART operation, please click on the document link below:

http://www.atmel.com/dyn/resources/prod_documents/DOC4316.PDF

Timer 0 and 1

Timer 0 and Timer 1 in the AT89S52 operate the same way as Timer 0 and Timer 1 in the AT89C51 and AT89C52. For further information on the timers' operation, please click on the document link below:

http://www.atmel.com/dyn/resources/prod_documents/DOC4316.PDF

0. Timer 2

Timer 2 is a 16-bit Timer/Counter that can operate as either a timer or an event counter. The type of operation is selected by bit C/$\overline{T2}$ in the SFR T2CON (shown in Table 5-2). Timer 2 has three operating modes: capture, auto-reload (up or down counting), and baud rate generator. The modes are selected by bits in T2CON, as shown in Table 10-1. Timer 2 consists of two 8-bit registers, TH2 and TL2. In the Timer function, the TL2 register is incremented every machine cycle. Since a machine cycle consists of 12 oscillator periods, the count rate is 1/12 of the oscillator frequency.

Table 10-1. Timer 2 Operating Modes

RCLK +TCLK	CP/$\overline{RL2}$	TR2	MODE
0	0	1	16-bit Auto-reload
0	1	1	16-bit Capture
1	X	1	Baud Rate Generator
X	X	0	(Off)

In the Counter function, the register is incremented in response to a 1-to-0 transition at its corresponding external input pin, T2. In this function, the external input is sampled during S5P2 of every machine cycle. When the samples show a high in one cycle and a low in the next cycle, the count is incremented. The new count value appears in the register during S3P1 of the cycle following the one in which the transition was detected. Since two machine cycles (24 oscillator periods) are required to recognize a 1-to-0 transition, the maximum count rate is 1/24 of the oscillator frequency. To ensure that a given level is sampled at least once before it changes, the level should be held for at least one full machine cycle.

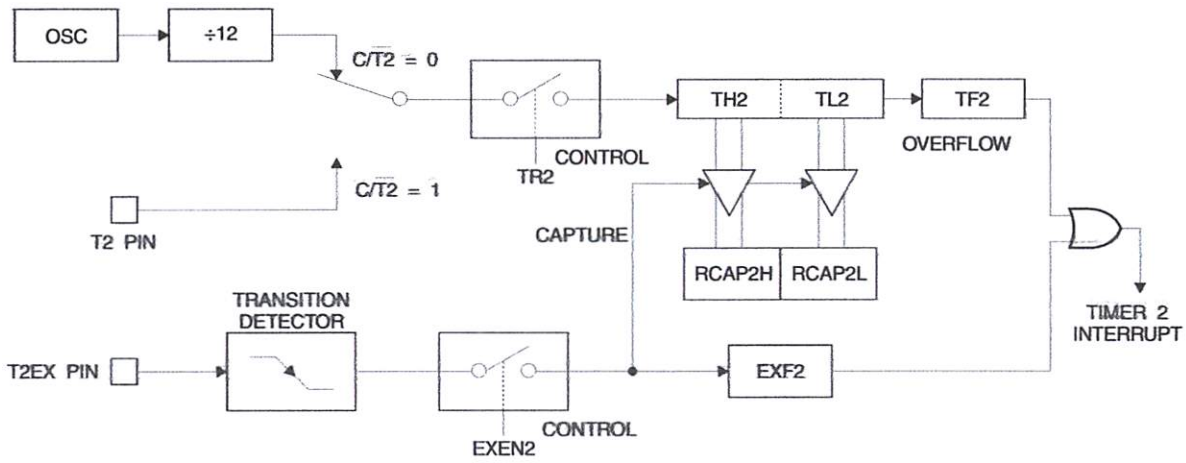
0.1 Capture Mode

In the capture mode, two options are selected by bit EXEN2 in T2CON. If EXEN2 = 0, Timer 2 is a 16-bit timer or counter which upon overflow sets bit TF2 in T2CON. This bit can then be used to generate an interrupt. If EXEN2 = 1, Timer 2 performs the same operation, but a 1-to-0 transition at external input T2EX also causes the current value in TH2 and TL2 to be captured into RCAP2H and RCAP2L, respectively. In addition, the transition at T2EX causes bit EXF2 in T2CON to be set. The EXF2 bit, like TF2, can generate an interrupt. The capture mode is illustrated in Figure 10-1.

0.2 Auto-reload (Up or Down Counter)

Timer 2 can be programmed to count up or down when configured in its 16-bit auto-reload mode. This feature is invoked by the DCEN (Down Counter Enable) bit located in the SFR T2MOD (see Table 10-2). Upon reset, the DCEN bit is set to 0 so that timer 2 will default to count up. When DCEN is set, Timer 2 can count up or down, depending on the value of the T2EX pin.

ure 10-1. Timer in Capture Mode



le 10-2. T2MOD – Timer 2 Mode Control Register

T2MOD Address = 0C9H — Reset Value = XXXX XX00B

Not Bit Addressable

	–	–	–	–	–	–	T2OE	DCEN
Bit	7	6	5	4	3	2	1	0

mbol	**Function**
	Not implemented, reserved for future
OE	Timer 2 Output Enable bit
CEN	When set, this bit allows Timer 2 to be configured as an up/down counter

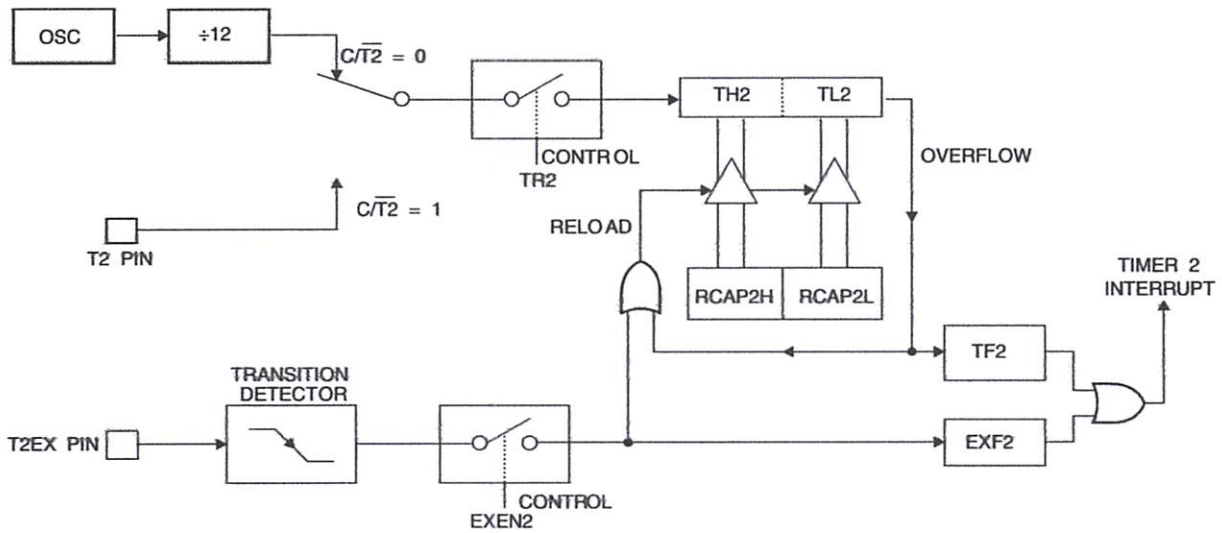
Figure 10-2 shows Timer 2 automatically counting up when DCEN = 0. In this mode, two options are selected by bit EXEN2 in T2CON. If EXEN2 = 0, Timer 2 counts up to 0FFFFH and then sets the TF2 bit upon overflow. The overflow also causes the timer registers to be reloaded with the 16-bit value in RCAP2H and RCAP2L. The values in Timer in Capture ModeRCAP2H and RCAP2L are preset by software. If EXEN2 = 1, a 16-bit reload can be triggered either by an overflow or by a 1-to-0 transition at external input T2EX. This transition also sets the EXF2 bit. Both the TF2 and EXF2 bits can generate an interrupt if enabled.

Setting the DCEN bit enables Timer 2 to count up or down, as shown in Figure 10-2. In this mode, the T2EX pin controls the direction of the count. A logic 1 at T2EX makes Timer 2 count up. The timer will overflow at 0FFFFH and set the TF2 bit. This overflow also causes the 16-bit value in RCAP2H and RCAP2L to be reloaded into the timer registers, TH2 and TL2, respectively.

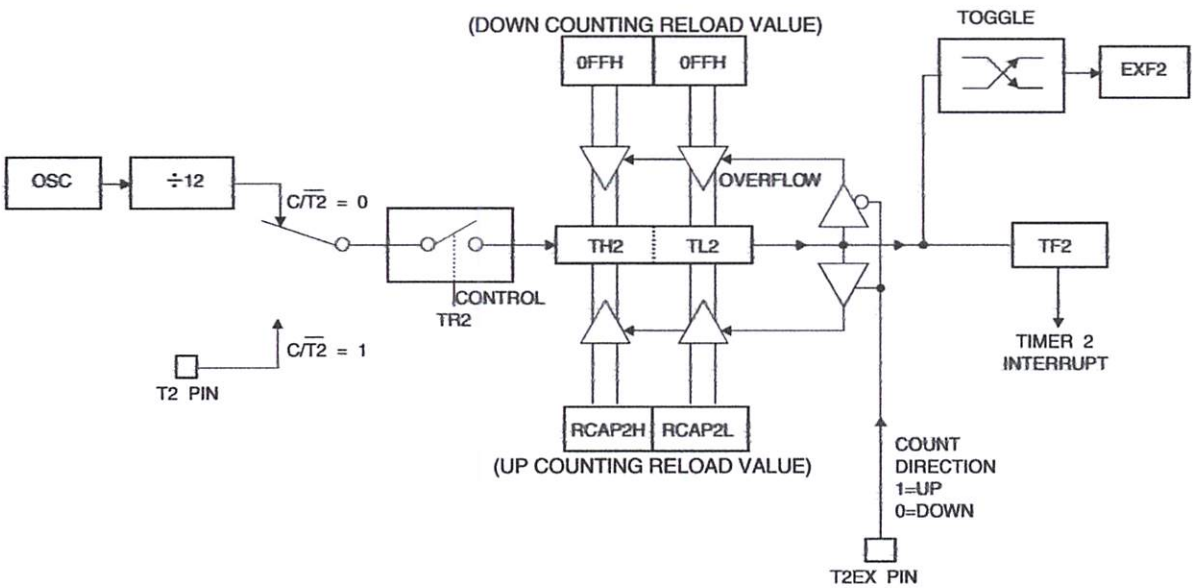
A logic 0 at T2EX makes Timer 2 count down. The timer underflows when TH2 and TL2 equal the values stored in RCAP2H and RCAP2L. The underflow sets the TF2 bit and causes 0FFFFH to be reloaded into the timer registers.

The EXF2 bit toggles whenever Timer 2 overflows or underflows and can be used as a 17th bit of resolution. In this operating mode, EXF2 does not flag an interrupt.

ure 10-2. Timer 2 Auto Reload Mode (DCEN = 0)



ure 10-3. Timer 2 Auto Reload Mode (DCEN = 1)



. Baud Rate Generator

Timer 2 is selected as the baud rate generator by setting TCLK and/or RCLK in T2CON (Table 5-2). Note that the baud rates for transmit and receive can be different if Timer 2 is used for the receiver or transmitter and Timer 1 is used for the other function. Setting RCLK and/or TCLK puts Timer 2 into its baud rate generator mode, as shown in Figure 11-1.

The baud rate generator mode is similar to the auto-reload mode, in that a rollover in TH2 causes the Timer 2 registers to be reloaded with the 16-bit value in registers RCAP2H and RCAP2L, which are preset by software.

The baud rates in Modes 1 and 3 are determined by Timer 2's overflow rate according to the following equation.

$$\text{Modes 1 and 3 Baud Rates} = \frac{\text{Timer 2 Overflow Rate}}{16}$$

The Timer can be configured for either timer or counter operation. In most applications, it is configured for timer operation ($CP/\overline{T2} = 0$). The timer operation is different for Timer 2 when it is used as a baud rate generator. Normally, as a timer, it increments every machine cycle (at 1/12 the oscillator frequency). As a baud rate generator, however, it increments every state time (at 1/2 the oscillator frequency). The baud rate formula is given below.

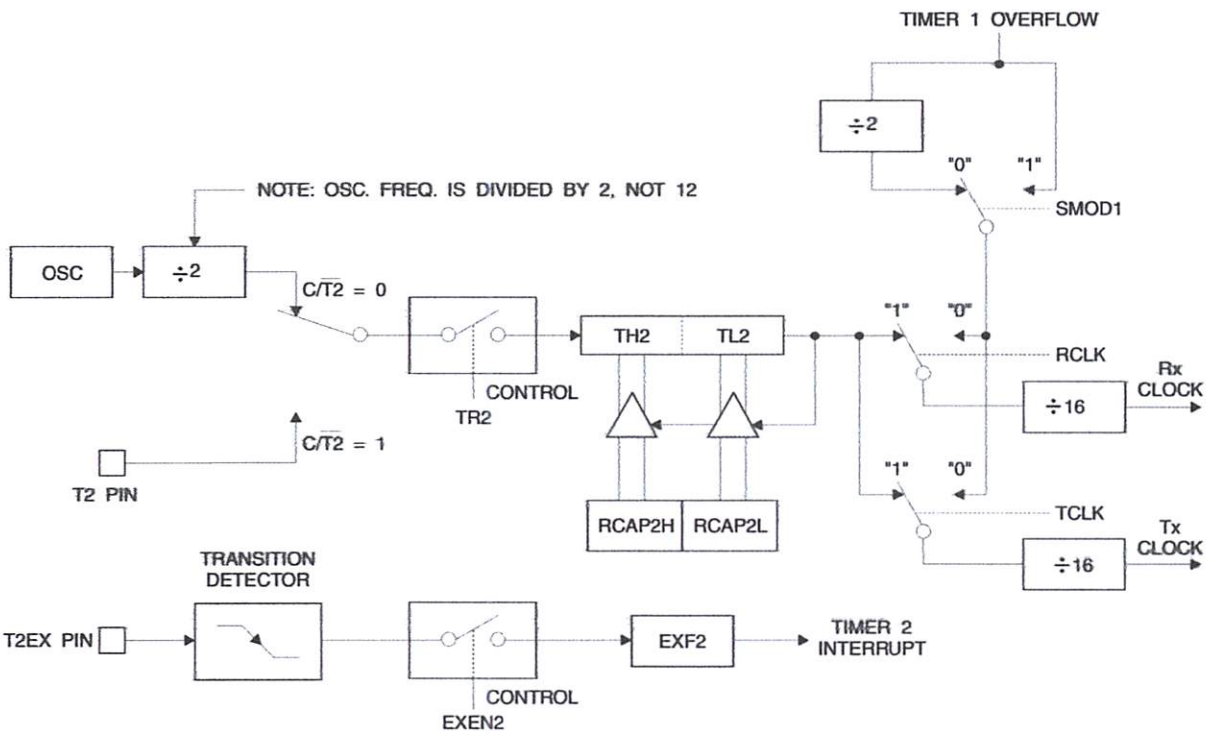
$$\frac{\text{Modes 1 and 3}}{\text{Baud Rate}} = \frac{\text{Oscillator Frequency}}{32 \times [65536\text{-RCAP2H,RCAP2L})]}$$

where (RCAP2H, RCAP2L) is the content of RCAP2H and RCAP2L taken as a 16-bit unsigned integer.

Timer 2 as a baud rate generator is shown in Figure 11-1. This figure is valid only if RCLK or TCLK = 1 in T2CON. Note that a rollover in TH2 does not set TF2 and will not generate an interrupt. Note too, that if EXEN2 is set, a 1-to-0 transition in T2EX will set EXF2 but will not cause a reload from (RCAP2H, RCAP2L) to (TH2, TL2). Thus, when Timer 2 is in use as a baud rate generator, T2EX can be used as an extra external interrupt.

Note that when Timer 2 is running (TR2 = 1) as a timer in the baud rate generator mode, TH2 or TL2 should not be read from or written to. Under these conditions, the Timer is incremented every state time, and the results of a read or write may not be accurate. The RCAP2 registers may be read but should not be written to, because a write might overlap a reload and cause write and/or reload errors. The timer should be turned off (clear TR2) before accessing the Timer 2 or RCAP2 registers.

ure 11-1. Timer 2 in Baud Rate Generator Mode



. Programmable Clock Out

A 50% duty cycle clock can be programmed to come out on P1.0, as shown in Figure 12-1. This pin, besides being a regular I/O pin, has two alternate functions. It can be programmed to input the external clock for Timer/Counter 2 or to output a 50% duty cycle clock ranging from 61 Hz to 4 MHz (for a 16-MHz operating frequency).

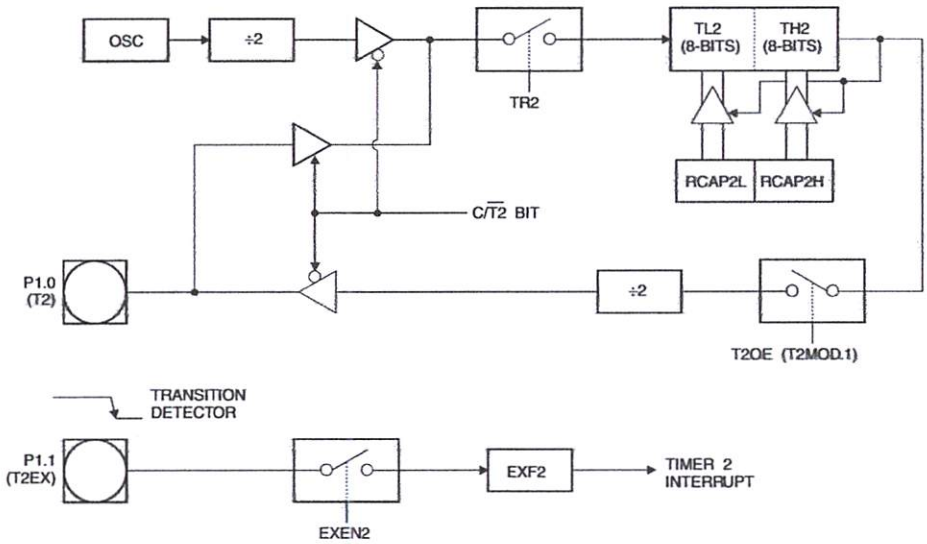
To configure the Timer/Counter 2 as a clock generator, bit C/$\overline{T2}$ (T2CON.1) must be cleared and bit T2OE (T2MOD.1) must be set. Bit TR2 (T2CON.2) starts and stops the timer.

The clock-out frequency depends on the oscillator frequency and the reload value of Timer 2 capture registers (RCAP2H, RCAP2L), as shown in the following equation.

$$\text{Clock-Out Frequency} = \frac{\text{Oscillator Frequency}}{4 \times [65536-(\text{RCAP2H,RCAP2L})]}$$

In the clock-out mode, Timer 2 roll-overs will not generate an interrupt. This behavior is similar to when Timer 2 is used as a baud-rate generator. It is possible to use Timer 2 as a baud-rate generator and a clock generator simultaneously. Note, however, that the baud-rate and clock-out frequencies cannot be determined independently from one another since they both use RCAP2H and RCAP2L.

ure 12-1. Timer 2 in Clock-Out Mode



. Interrupts

The AT89S52 has a total of six interrupt vectors: two external interrupts ($\overline{INT0}$ and $\overline{INT1}$), three timer interrupts (Timers 0, 1, and 2), and the serial port interrupt. These interrupts are all shown in Figure 13-1.

Each of these interrupt sources can be individually enabled or disabled by setting or clearing a bit in Special Function Register IE. IE also contains a global disable bit, EA, which disables all interrupts at once.

Note that Table 13-1 shows that bit position IE.6 is unimplemented. User software should not write a 1 to this bit position, since it may be used in future AT89 products.

Timer 2 interrupt is generated by the logical OR of bits TF2 and EXF2 in register T2CON. Neither of these flags is cleared by hardware when the service routine is vectored to. In fact, the service routine may have to determine whether it was TF2 or EXF2 that generated the interrupt, and that bit will have to be cleared in software.

The Timer 0 and Timer 1 flags, TF0 and TF1, are set at S5P2 of the cycle in which the timers overflow. The values are then polled by the circuitry in the next cycle. However, the Timer 2 flag, TF2, is set at S2P2 and is polled in the same cycle in which the timer overflows.

ble 13-1. Interrupt Enable (IE) Register

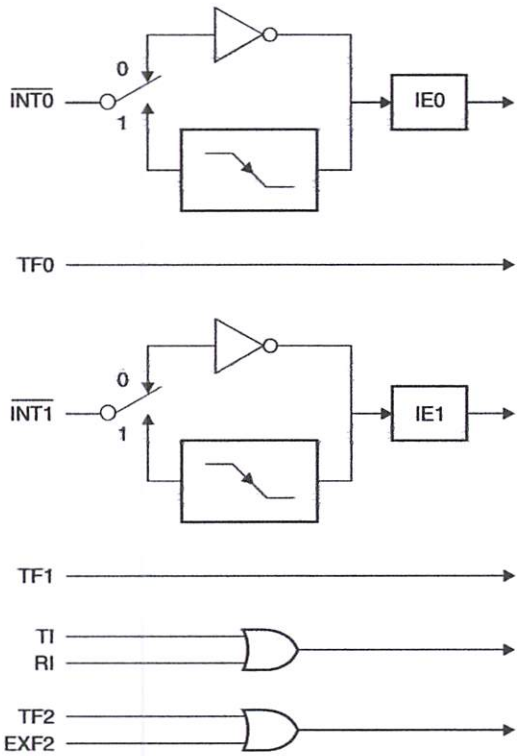
(MSB)							(LSB)
EA	–	ET2	ES	ET1	EX1	ET0	EX0

Enable Bit = 1 enables the interrupt.

Enable Bit = 0 disables the interrupt.

mbol	Position	Function
	IE.7	Disables all interrupts. If EA = 0, no interrupt is acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
	IE.6	Reserved.
2	IE.5	Timer 2 interrupt enable bit.
	IE.4	Serial Port interrupt enable bit.
1	IE.3	Timer 1 interrupt enable bit.
1	IE.2	External interrupt 1 enable bit.
0	IE.1	Timer 0 interrupt enable bit.
0	IE.0	External interrupt 0 enable bit.
er software should never write 1s to reserved bits, because they may be used in future AT89 products.		

ure 13-1. Interrupt Sources



. Oscillator Characteristics

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier that can be configured for use as an on-chip oscillator, as shown in Figure 16-1. Either a quartz crystal or ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected while XTAL1 is driven, as shown in Figure 16-2. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum voltage high and low time specifications must be observed.

. Idle Mode

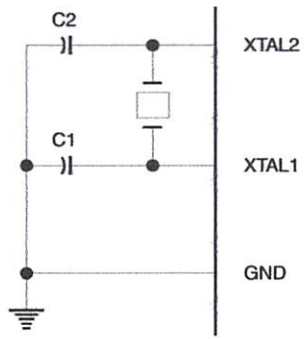
In idle mode, the CPU puts itself to sleep while all the on-chip peripherals remain active. The mode is invoked by software. The content of the on-chip RAM and all the special functions registers remain unchanged during this mode. The idle mode can be terminated by any enabled interrupt or by a hardware reset.

Note that when idle mode is terminated by a hardware reset, the device normally resumes program execution from where it left off, up to two machine cycles before the internal reset algorithm takes control. On-chip hardware inhibits access to internal RAM in this event, but access to the port pins is not inhibited. To eliminate the possibility of an unexpected write to a port pin when idle mode is terminated by a reset, the instruction following the one that invokes idle mode should not write to a port pin or to external memory.

. Power-down Mode

In the Power-down mode, the oscillator is stopped, and the instruction that invokes Power-down is the last instruction executed. The on-chip RAM and Special Function Registers retain their values until the Power-down mode is terminated. Exit from Power-down mode can be initiated either by a hardware reset or by an enabled external interrupt. Reset redefines the SFRs but does not change the on-chip RAM. The reset should not be activated before V_{CC} is restored to its normal operating level and must be held active long enough to allow the oscillator to restart and stabilize.

Figure 16-1. Oscillator Connections



Note: 1. C1, C2 = 30 pF ±10 pF for Crystals
= 40 pF ±10 pF for Ceramic Resonators

Figure 16-2. External Clock Drive Configuration

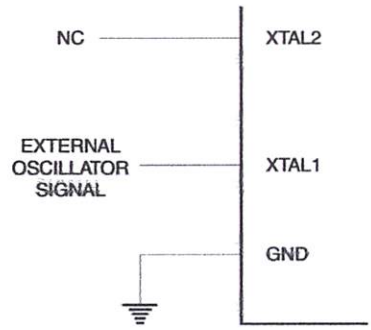


Table 16-1. Status of External Pins During Idle and Power-down Modes

Mode	Program Memory	ALE	$\overline{PSEN}$	PORT0	PORT1	PORT2	PORT3
Idle	Internal	1	1	Data	Data	Data	Data
Idle	External	1	1	Float	Data	Address	Data
Power-down	Internal	0	0	Data	Data	Data	Data
Power-down	External	0	0	Float	Data	Data	Data

. Program Memory Lock Bits

The AT89S52 has three lock bits that can be left unprogrammed (U) or can be programmed (P) to obtain the additional features listed in Table 17-1.

Table 17-1. Lock Bit Protection Modes

Program Lock Bits				
	LB1	LB2	LB3	Protection Type
1	U	U	U	No program lock features
2	P	U	U	MOVC instructions executed from external program memory are disabled from fetching code bytes from internal memory, $\overline{EA}$ is sampled and latched on reset, and further programming of the Flash memory is disabled
3	P	P	U	Same as mode 2, but verify is also disabled
4	P	P	P	Same as mode 3, but external execution is also disabled

When lock bit 1 is programmed, the logic level at the $\overline{EA}$ pin is sampled and latched during reset. If the device is powered up without a reset, the latch initializes to a random value and holds that value until reset is activated. The latched value of $\overline{EA}$ must agree with the current logic level at that pin in order for the device to function properly.

. Programming the Flash – Parallel Mode

The AT89S52 is shipped with the on-chip Flash memory array ready to be programmed. The programming interface needs a high-voltage (12-volt) program enable signal and is compatible with conventional third-party Flash or EPROM programmers.

The AT89S52 code memory array is programmed byte-by-byte.

Programming Algorithm: Before programming the AT89S52, the address, data, and control signals should be set up according to the "Flash Programming Modes" (Table 22-1) and Figure 22-1 and Figure 22-2. To program the AT89S52, take the following steps:

1. Input the desired memory location on the address lines.
2. Input the appropriate data byte on the data lines.
3. Activate the correct combination of control signals.
4. Raise $\overline{EA}/V_{PP}$ to 12V.
5. Pulse ALE/$\overline{PROG}$ once to program a byte in the Flash array or the lock bits. The byte-write cycle is self-timed and typically takes no more than 50 µs. Repeat steps 1 through 5, changing the address and data for the entire array or until the end of the object file is reached.

$\overline{Data}$ Polling: The AT89S52 features $\overline{Data}$ Polling to indicate the end of a byte write cycle. During a write cycle, an attempted read of the last byte written will result in the complement of the written data on P0.7. Once the write cycle has been completed, true data is valid on all outputs, and the next cycle may begin. $\overline{Data}$ Polling may begin any time after a write cycle has been initiated.

Ready/$\overline{Busy}$: The progress of byte programming can also be monitored by the RDY/$\overline{BSY}$ output signal. P3.0 is pulled low after ALE goes high during programming to indicate $\overline{BUSY}$. P3.0 is pulled high again when programming is done to indicate READY.

Program Verify: If lock bits LB1 and LB2 have not been programmed, the programmed code data can be read back via the address and data lines for verification. **The status of the individual lock bits can be verified directly by reading them back.**

Reading the Signature Bytes: The signature bytes are read by the same procedure as a normal verification of locations 000H, 100H, and 200H, except that P3.6 and P3.7 must be pulled to a logic low. The values returned are as follows.

(000H) = 1EH indicates manufactured by Atmel
(100H) = 52H indicates AT89S52
(200H) = 06H

Chip Erase: In the parallel programming mode, a chip erase operation is initiated by using the proper combination of control signals and by pulsing ALE/$\overline{PROG}$ low for a duration of 200 ns - 500 ns.

In the serial programming mode, a chip erase operation is initiated by issuing the Chip Erase instruction. In this mode, chip erase is self-timed and takes about 500 ms.

During chip erase, a serial read from any address location will return 00H at the data output.

. Programming the Flash – Serial Mode

The Code memory array can be programmed using the serial ISP interface while RST is pulled to V_{CC}. The serial interface consists of pins SCK, MOSI (input) and MISO (output). After RST is set high, the Programming Enable instruction needs to be executed first before other operations can be executed. Before a reprogramming sequence can occur, a Chip Erase operation is required.

The Chip Erase operation turns the content of every memory location in the Code array into FFH.

Either an external system clock can be supplied at pin XTAL1 or a crystal needs to be connected across pins XTAL1 and XTAL2. The maximum serial clock (SCK) frequency should be less than 1/16 of the crystal frequency. With a 33 MHz oscillator clock, the maximum SCK frequency is 2 MHz.

. Serial Programming Algorithm

To program and verify the AT89S52 in the serial programming mode, the following sequence is recommended:

1. Power-up sequence:
   a. Apply power between VCC and GND pins.
   b. Set RST pin to "H".

If a crystal is not connected across pins XTAL1 and XTAL2, apply a 3 MHz to 33 MHz clock to XTAL1 pin and wait for at least 10 milliseconds.

2. Enable serial programming by sending the Programming Enable serial instruction to pin MOSI/P1.5. The frequency of the shift clock supplied at pin SCK/P1.7 needs to be less than the CPU clock at XTAL1 divided by 16.
3. The Code array is programmed one byte at a time in either the Byte or Page mode. The write cycle is self-timed and typically takes less than 0.5 ms at 5V.
4. Any memory location can be verified by using the Read instruction which returns the content at the selected address at serial output MISO/P1.6.
5. At the end of a programming session, RST can be set low to commence normal device operation.

Power-off sequence (if needed):

1. Set XTAL1 to "L" (if a crystal is not used).
2. Set RST to "L".
3. Turn V_{CC} power off.

Data Polling: The $\overline{\text{Data}}$ Polling feature is also available in the serial mode. In this mode, during a write cycle an attempted read of the last byte written will result in the complement of the MSB of the serial output byte on MISO.

. Serial Programming Instruction Set

The Instruction Set for Serial Programming follows a 4-byte protocol and is shown in Table 24-1.

. Programming Interface – Parallel Mode

Every code byte in the Flash array can be programmed by using the appropriate combination of control signals. The write operation cycle is self-timed and once initiated, will automatically time itself to completion.

Most major worldwide programming vendors offer support for the Atmel AT89 microcontroller series. Please contact your local programming vendor for the appropriate software revision.

ble 22-1. Flash Programming Modes

ode	V_{CC}	RST	$\overline{PSEN}$	ALE/ $\overline{PROG}$	$\overline{EA}$/ V_{PP}	P2.6	P2.7	P3.3	P3.6	P3.7	P0.7-0 Data	P2.4-0	P1.7-0
												Address	
rite Code Data	5V	H	L	(2)	12V	L	H	H	H	H	D_{IN}	A12-8	A7-0
ead Code Data	5V	H	L	H	H	L	L	L	H	H	D_{OUT}	A12-8	A7-0
rite Lock Bit 1	5V	H	L	(3)	12V	H	H	H	H	H	X	X	X
rite Lock Bit 2	5V	H	L	(3)	12V	H	H	H	L	L	X	X	X
rite Lock Bit 3	5V	H	L	(3)	12V	H	L	H	H	L	X	X	X
ead Lock Bits 2, 3	5V	H	L	H	H	H	H	L	H	L	P0.2, P0.3, P0.4	X	X
hip Erase	5V	H	L	(1)	12V	H	L	H	L	L	X	X	X
ead Atmel ID	5V	H	L	H	H	L	L	L	L	L	1EH	X 0000	00H
ead Device ID	5V	H	L	H	H	L	L	L	L	L	52H	X 0001	00H
ead Device ID	5V	H	L	H	H	L	L	L	L	L	06H	X 0010	00H

es:
1. Each $\overline{PROG}$ pulse is 200 ns - 500 ns for Chip Erase.
2. Each $\overline{PROG}$ pulse is 200 ns - 500 ns for Write Code Data.
3. Each $\overline{PROG}$ pulse is 200 ns - 500 ns for Write Lock Bits.
4. RDY/$\overline{BSY}$ signal is output on P3.0 during programming.
5. X = don't care.

Figure 22-1. Programming the Flash Memory (Parallel Mode)

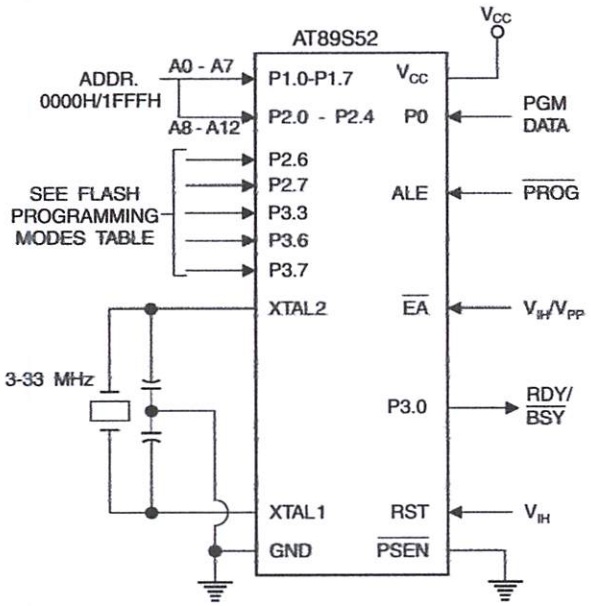
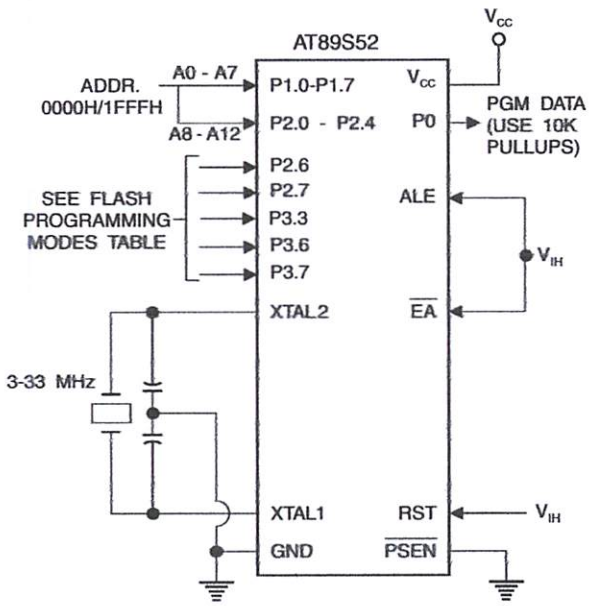


Figure 22-2. Verifying the Flash Memory (Parallel Mode)



. Flash Programming and Verification Characteristics (Parallel Mode)

= 20°C to 30°C, V_{CC} = 4.5 to 5.5V

ymbol	Parameter	Min	Max	Units
PP	Programming Supply Voltage	11.5	12.5	V
P	Programming Supply Current		10	mA
C	V_{CC} Supply Current		30	mA
CLCL	Oscillator Frequency	3	33	MHz
VGL	Address Setup to $\overline{PROG}$ Low	48 t_{CLCL}		
HAX	Address Hold After $\overline{PROG}$	48 t_{CLCL}		
VGL	Data Setup to $\overline{PROG}$ Low	48 t_{CLCL}		
HDX	Data Hold After $\overline{PROG}$	48 t_{CLCL}		
HSH	P2.7 ($\overline{ENABLE}$) High to V_{PP}	48 t_{CLCL}		
HGL	V_{PP} Setup to $\overline{PROG}$ Low	10		µs
HSL	V_{PP} Hold After $\overline{PROG}$	10		µs
LGH	$\overline{PROG}$ Width	0.2	1	µs
VQV	Address to Data Valid		48 t_{CLCL}	
LQV	$\overline{ENABLE}$ Low to Data Valid		48 t_{CLCL}	
HQZ	Data Float After $\overline{ENABLE}$	0	48 t_{CLCL}	
HBL	$\overline{PROG}$ High to $\overline{BUSY}$ Low		1.0	µs
C	Byte Write Cycle Time		50	µs

ure 23-1. Flash Programming and Verification Waveforms – Parallel Mode

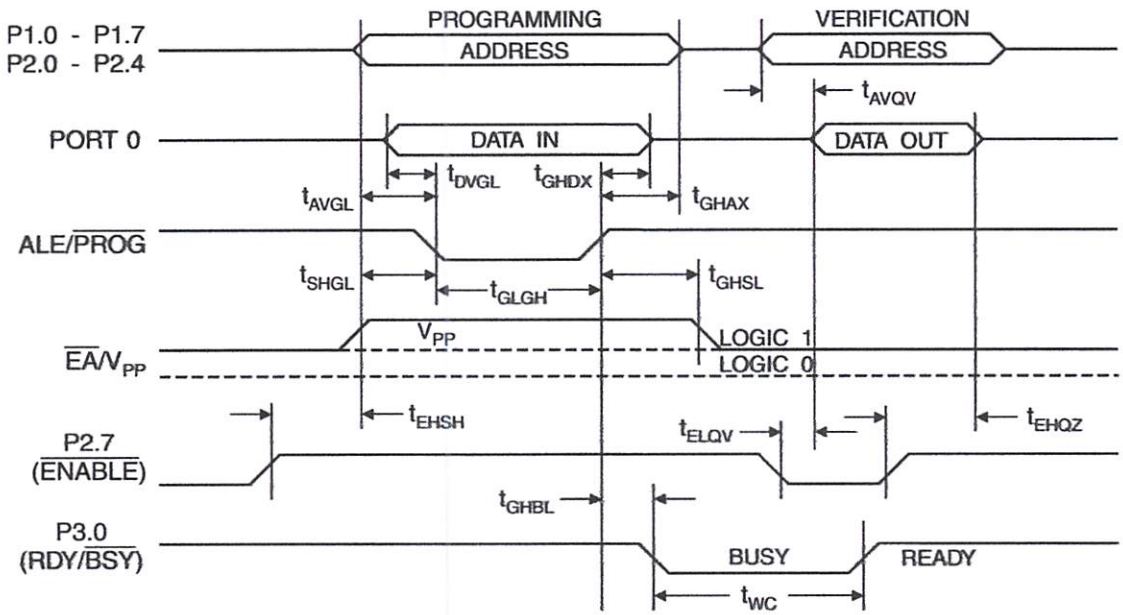
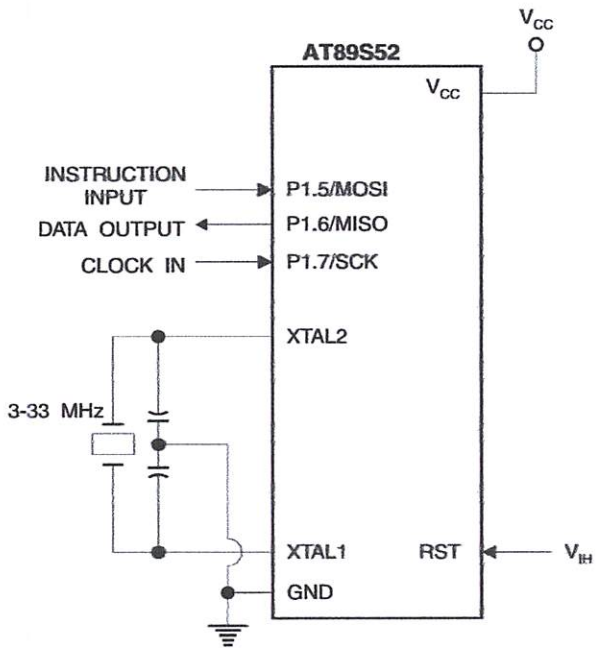
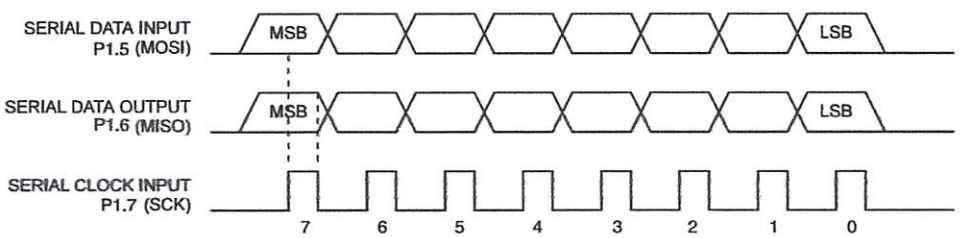


Figure 23-2. Flash Memory Serial Downloading



4. Flash Programming and Verification Waveforms – Serial Mode

Figure 24-1. Serial Programming Waveforms



ble 24-1. Serial Programming Instruction Set

	Instruction Format				
struction	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Operation**
rogramming Enable	1010 1100	0101 0011	xxxx xxxx	xxxx xxxx 0110 1001 (Output on MISO)	Enable Serial Programming while RST is high
hip Erase	1010 1100	100x xxxx	xxxx xxxx	xxxx xxxx	Chip Erase Flash memory array
ead Program Memory Byte Mode)	0010 0000	xxx A12 A11 A10 A9 A8	A7 A6 A5 A4 A3 A2 A1 A0	D7 D6 D5 D4 D3 D2 D1 D0	Read data from Program memory in the byte mode
Vrite Program Memory Byte Mode)	0100 0000	xxx A12 A11 A10 A9 A8	A7 A6 A5 A4 A3 A2 A1 A0	D7 D6 D5 D4 D3 D2 D1 D0	Write data to Program memory in the byte mode
Vrite Lock Bits[1]	1010 1100	1110 00 B1 B2	xxxx xxxx	xxxx xxxx	Write Lock bits. See Note (1).
ead Lock Bits	0010 0100	xxxx xxxx	xxxx xxxx	xxx LB3 LB2 LB1 xx	Read back current status of the lock bits (a programmed lock bit reads back as a "1")
ead Signature Bytes	0010 1000	xxx A12 A11 A10 A9 A8	A7 xxx xxx0	Signature Byte	Read Signature Byte
ead Program Memory Page Mode)	0011 0000	xxx A12 A11 A10 A9 A8	Byte 0	Byte 1... Byte 255	Read data from Program memory in the Page Mode (256 bytes)
Vrite Program Memory Page Mode)	0101 0000	xxx A12 A11 A10 A9 A8	Byte 0	Byte 1... Byte 255	Write data to Program memory in the Page Mode (256 bytes)

te: 1. B1 = 0, B2 = 0 ---> Mode 1, no lock protection
B1 = 0, B2 = 1 ---> Mode 2, lock bit 1 activated
B1 = 1, B2 = 0 ---> Mode 3, lock bit 2 activated
B1 = 1, B2 = 1 ---> Mode 4, lock bit 3 activated

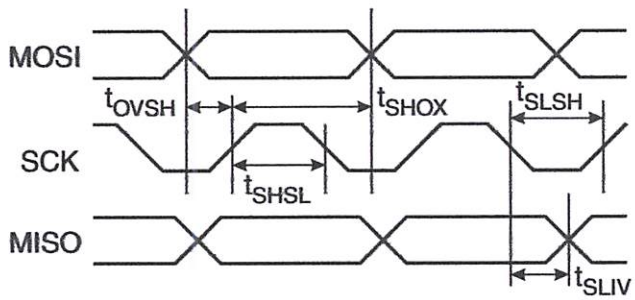
} Each of the lock bit modes needs to be activated sequentially before Mode 4 can be executed.

After Reset signal is high, SCK should be low for at least 64 system clocks before it goes high to clock in the enable data bytes. No pulsing of Reset signal is necessary. SCK should be no faster than 1/16 of the system clock at XTAL1.

For Page Read/Write, the data always starts from byte 0 to 255. After the command byte and upper address byte are latched, each byte thereafter is treated as data until all 256 bytes are shifted in/out. Then the next instruction will be ready to be decoded.

5. Serial Programming Characteristics

gure 25-1. Serial Programming Timing



ble 25-1. Serial Programming Characteristics, T_A = -40· C to 85· C, V_{CC} = 4.0 - 5.5V (Unless Otherwise Noted)

ymbol	Parameter	Min	Typ	Max	Units
/t_{CLCL}	Oscillator Frequency	3		33	MHz
CLCL	Oscillator Period	30			ns
SHSL	SCK Pulse Width High	8 t_{CLCL}			ns
SLSH	SCK Pulse Width Low	8 t_{CLCL}			ns
OVSH	MOSI Setup to SCK High	t_{CLCL}			ns
SHOX	MOSI Hold after SCK High	2 t_{CLCL}			ns
SLIV	SCK Low to MISO Valid	10	16	32	ns
ERASE	Chip Erase Instruction Cycle Time			500	ms
SWC	Serial Byte Write Cycle Time			64 t_{CLCL} + 400	µs

6. Absolute Maximum Ratings*

perating Temperature	-55°C to +125°C
torage Temperature	-65°C to +150°C
oltage on Any Pin ith Respect to Ground	-1.0V to +7.0V
laximum Operating Voltage	6.6V
C Output Current	15.0 mA

*NOTICE: Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

7. DC Characteristics

e values shown in this table are valid for T_A = -40°C to 85°C and V_{CC} = 4.0V to 5.5V, unless otherwise noted.

ymbol	Parameter	Condition	Min	Max	Units
IL	Input Low Voltage	(Except $\overline{EA}$)	-0.5	0.2 V_{CC}-0.1	V
IL1	Input Low Voltage ($\overline{EA}$)		-0.5	0.2 V_{CC}-0.3	V
IH	Input High Voltage	(Except XTAL1, RST)	0.2 V_{CC}+0.9	V_{CC}+0.5	V
IH1	Input High Voltage	(XTAL1, RST)	0.7 V_{CC}	V_{CC}+0.5	V
OL	Output Low Voltage[1] (Ports 1,2,3)	I_{OL} = 1.6 mA		0.45	V
OL1	Output Low Voltage[1] (Port 0, ALE, $\overline{PSEN}$)	I_{OL} = 3.2 mA		0.45	V
OH	Output High Voltage (Ports 1,2,3, ALE, $\overline{PSEN}$)	I_{OH} = -60 µA, V_{CC} = 5V ±10%	2.4		V
		I_{OH} = -25 µA	0.75 V_{CC}		V
		I_{OH} = -10 µA	0.9 V_{CC}		V
OH1	Output High Voltage (Port 0 in External Bus Mode)	I_{OH} = -800 µA, V_{CC} = 5V ±10%	2.4		V
		I_{OH} = -300 µA	0.75 V_{CC}		V
		I_{OH} = -80 µA	0.9 V_{CC}		V
-	Logical 0 Input Current (Ports 1,2,3)	V_{IN} = 0.45V		-50	µA
L	Logical 1 to 0 Transition Current (Ports 1,2,3)	V_{IN} = 2V, V_{CC} = 5V ±10%		-300	µA
I	Input Leakage Current (Port 0, $\overline{EA}$)	0.45 < V_{IN} < V_{CC}		±10	µA
RST	Reset Pulldown Resistor		50	300	KΩ
IO	Pin Capacitance	Test Freq. = 1 MHz, T_A = 25°C		10	pF
CC	Power Supply Current	Active Mode, 12 MHz		25	mA
		Idle Mode, 12 MHz		6.5	mA
	Power-down Mode[1]	V_{CC} = 5.5V		50	µA

tes:
1. Under steady state (non-transient) conditions, I_{OL} must be externally limited as follows:
   Maximum I_{OL} per port pin: 10 mA
   Maximum I_{OL} per 8-bit port:
   Port 0: 26 mA Ports 1, 2, 3: 15 mA
   Maximum total I_{OL} for all output pins: 71 mA
   If I_{OL} exceeds the test condition, V_{OL} may exceed the related specification. Pins are not guaranteed to sink current greater than the listed test conditions.
2. Minimum V_{CC} for Power-down is 2V.

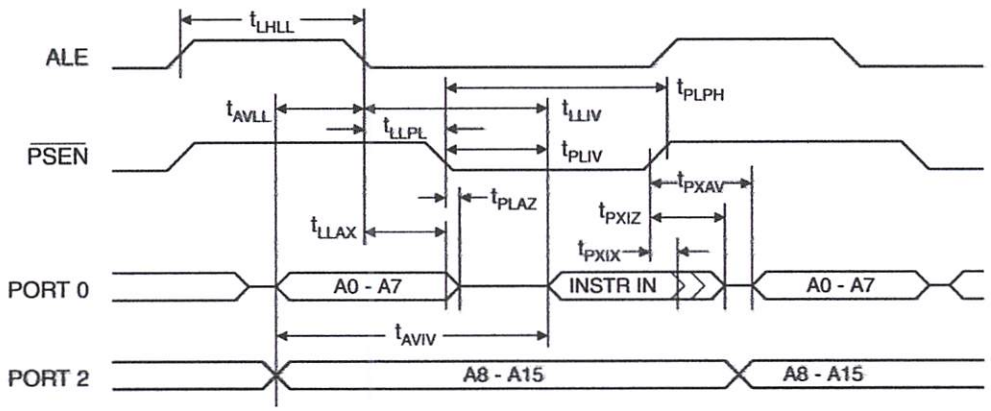
3. AC Characteristics

der operating conditions, load capacitance for Port 0, ALE/$\overline{PROG}$, and $\overline{PSEN}$ = 100 pF; load capacitance for all other tputs = 80 pF.

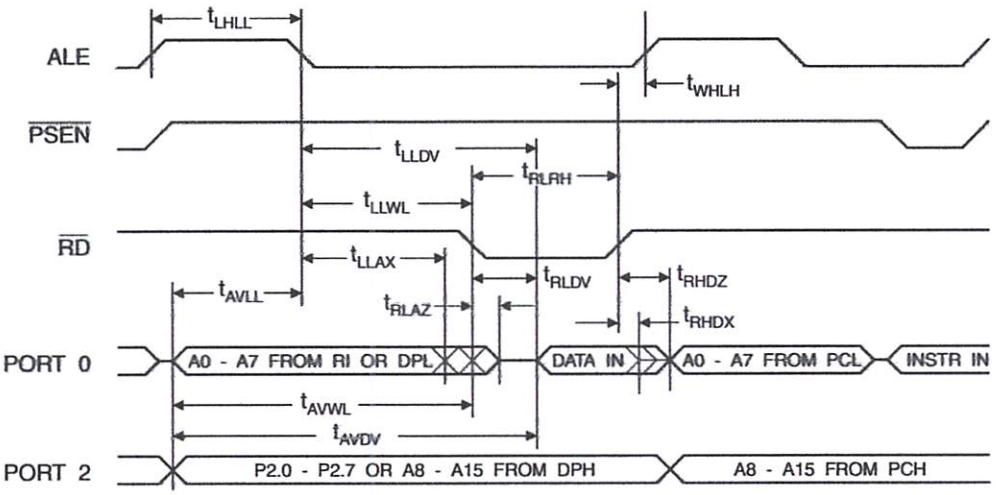
3.1 External Program and Data Memory Characteristics

ymbol	Parameter	12 MHz Oscillator		Variable Oscillator		Units
		Min	Max	Min	Max	
/t_{CLCL}	Oscillator Frequency			0	33	MHz
HLL	ALE Pulse Width	127		$2t_{CLCL}$-40		ns
VLL	Address Valid to ALE Low	43		t_{CLCL}-25		ns
LAX	Address Hold After ALE Low	48		t_{CLCL}-25		ns
LIV	ALE Low to Valid Instruction In		233		$4t_{CLCL}$-65	ns
LPL	ALE Low to $\overline{PSEN}$ Low	43		t_{CLCL}-25		ns
PLPH	$\overline{PSEN}$ Pulse Width	205		$3t_{CLCL}$-45		ns
PLIV	$\overline{PSEN}$ Low to Valid Instruction In		145		$3t_{CLCL}$-60	ns
PXIX	Input Instruction Hold After $\overline{PSEN}$	0		0		ns
PXIZ	Input Instruction Float After $\overline{PSEN}$		59		t_{CLCL}-25	ns
PXAV	$\overline{PSEN}$ to Address Valid	75		t_{CLCL}-8		ns
AVIV	Address to Valid Instruction In		312		$5t_{CLCL}$-80	ns
PLAZ	$\overline{PSEN}$ Low to Address Float		10		10	ns
RLRH	$\overline{RD}$ Pulse Width	400		$6t_{CLCL}$-100		ns
WLWH	$\overline{WR}$ Pulse Width	400		$6t_{CLCL}$-100		ns
RLDV	$\overline{RD}$ Low to Valid Data In		252		$5t_{CLCL}$-90	ns
RHDX	Data Hold After $\overline{RD}$	0		0		ns
RHDZ	Data Float After $\overline{RD}$		97		$2t_{CLCL}$-28	ns
LDV	ALE Low to Valid Data In		517		$8t_{CLCL}$-150	ns
AVDV	Address to Valid Data In		585		$9t_{CLCL}$-165	ns
LWL	ALE Low to $\overline{RD}$ or $\overline{WR}$ Low	200	300	$3t_{CLCL}$-50	$3t_{CLCL}$+50	ns
AVWL	Address to $\overline{RD}$ or $\overline{WR}$ Low	203		$4t_{CLCL}$-75		ns
QVWX	Data Valid to $\overline{WR}$ Transition	23		t_{CLCL}-30		ns
QVWH	Data Valid to $\overline{WR}$ High	433		$7t_{CLCL}$-130		ns
WHQX	Data Hold After $\overline{WR}$	33		t_{CLCL}-25		ns
RLAZ	$\overline{RD}$ Low to Address Float		0		0	ns
WHLH	$\overline{RD}$ or $\overline{WR}$ High to ALE High	43	123	t_{CLCL}-25	t_{CLCL}+25	ns

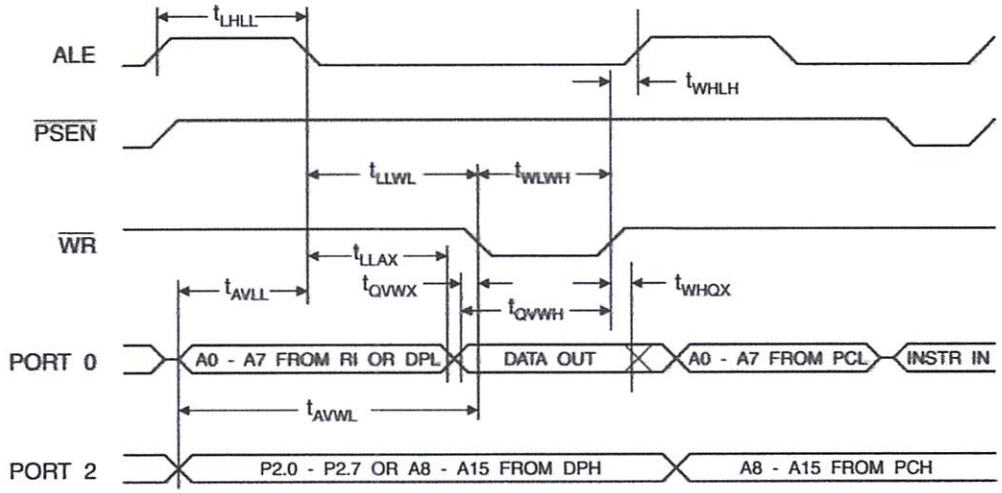
9. External Program Memory Read Cycle



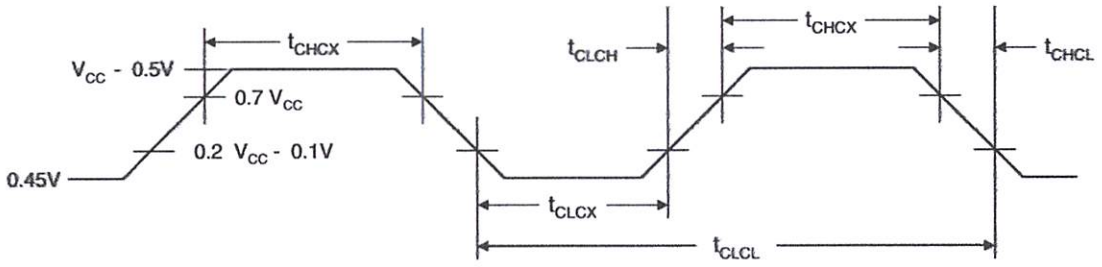
10. External Data Memory Read Cycle



1. External Data Memory Write Cycle



2. External Clock Drive Waveforms



3. External Clock Drive

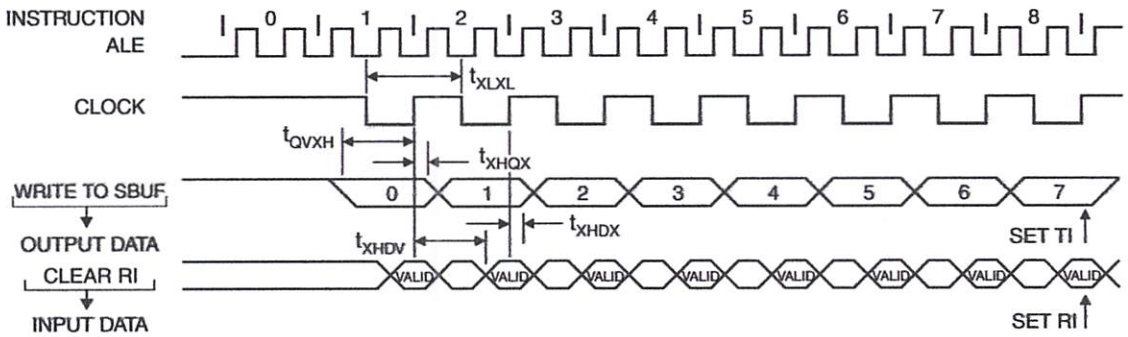
Symbol	Parameter	Min	Max	Units
/t_{CLCL}	Oscillator Frequency	0	33	MHz
CLCL	Clock Period	30		ns
CHCX	High Time	12		ns
CLCX	Low Time	12		ns
CLCH	Rise Time		5	ns
CHCL	Fall Time		5	ns

4. Serial Port Timing: Shift Register Mode Test Conditions

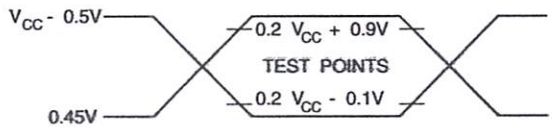
e values in this table are valid for V_{CC} = 4.0V to 5.5V and Load Capacitance = 80 pF.

ymbol	Parameter	12 MHz Osc		Variable Oscillator		Units
		Min	Max	Min	Max	
XLXL	Serial Port Clock Cycle Time	1.0		12 t_{CLCL}		µs
QVXH	Output Data Setup to Clock Rising Edge	700		10 t_{CLCL}-133		ns
XHQX	Output Data Hold After Clock Rising Edge	50		2 t_{CLCL}-80		ns
XHDX	Input Data Hold After Clock Rising Edge	0		0		ns
XHDV	Clock Rising Edge to Input Data Valid		700		10 t_{CLCL}-133	ns

5. Shift Register Mode Timing Waveforms

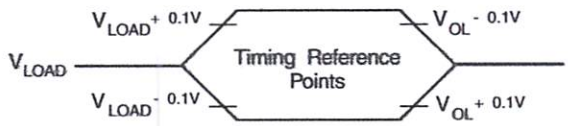


6. AC Testing Input/Output Waveforms[1]



te: 1. AC Inputs during testing are driven at V_{CC} - 0.5V for a logic 1 and 0.45V for a logic 0. Timing measurements are made at V_{IH} min. for a logic 1 and V_{IL} max. for a logic 0.

7. Float Waveforms[1]



te: 1. For timing purposes, a port pin is no longer floating when a 100 mV change from load voltage occurs. A port pin begins to float when a 100 mV change from the loaded V_{OH}/V_{OL} level occurs.

3. Ordering Information

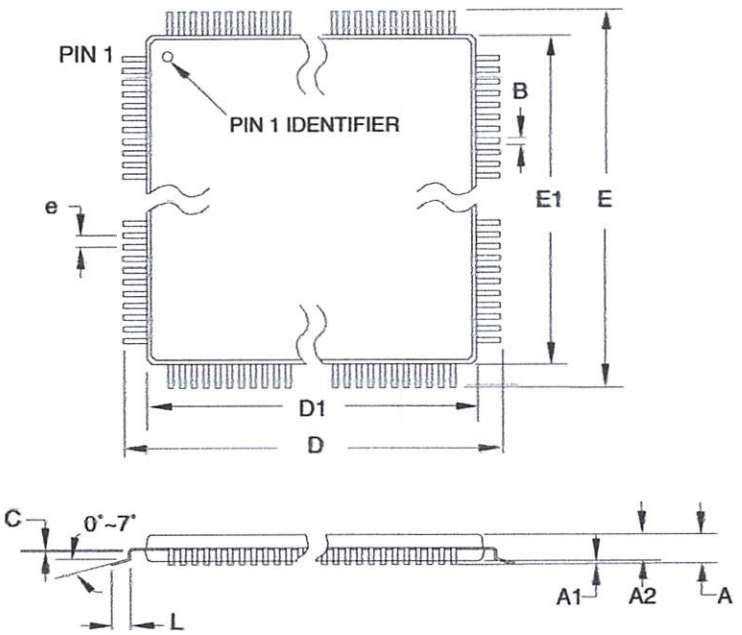
3.1 Green Package Option (Pb/Halide-free)

Speed (MHz)	Power Supply	Ordering Code	Package	Operation Range
24	4.0V to 5.5V	AT89S52-24AU AT89S52-24JU AT89S52-24PU	44A 44J 40P6	Industrial (-40° C to 85° C)
33	4.5V to 5.5V	AT89S52-33AU AT89S52-33JU AT89S52-33PU	44A 44J 40P6	Industrial (-40° C to 85° C)

	Package Type
4A	44-lead, Thin Plastic Gull Wing Quad Flatpack (TQFP)
4J	44-lead, Plastic J-leaded Chip Carrier (PLCC)
0P6	40-pin, 0.600" Wide, Plastic Dual Inline Package (PDIP)

9. Packaging Information

9.1 44A – TQFP



COMMON DIMENSIONS
(Unit of Measure = mm)

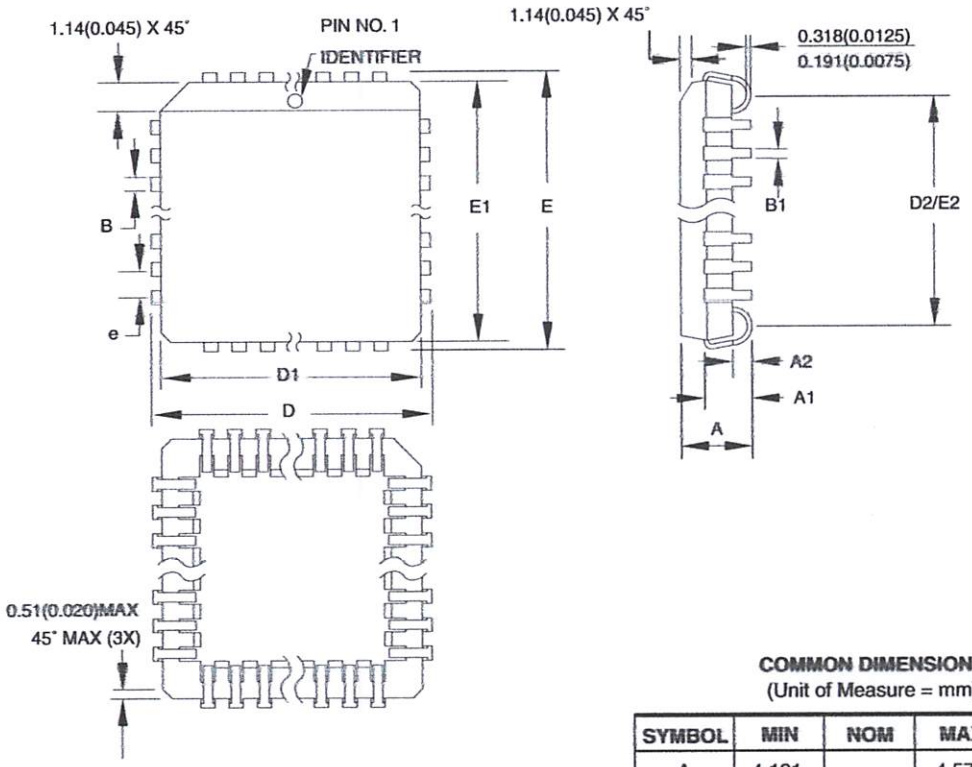
SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	1.20	
A1	0.05	–	0.15	
A2	0.95	1.00	1.05	
D	11.75	12.00	12.25	
D1	9.90	10.00	10.10	Note 2
E	11.75	12.00	12.25	
E1	9.90	10.00	10.10	Note 2
B	0.30	–	0.45	
C	0.09	–	0.20	
L	0.45	–	0.75	
e	0.80 TYP			

Notes:
1. This package conforms to JEDEC reference MS-026, Variation ACB.
2. Dimensions D1 and E1 do not include mold protrusion. Allowable protrusion is 0.25 mm per side. Dimensions D1 and E1 are maximum plastic body size dimensions including mold mismatch.
3. Lead coplanarity is 0.10 mm maximum.

10/5/2001

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**44A**, 44-lead, 10 x 10 mm Body Size, 1.0 mm Body Thickness, 0.8 mm Lead Pitch, Thin Profile Plastic Quad Flat Package (TQFP)	44A	B

0.2 44J – PLCC



COMMON DIMENSIONS
(Unit of Measure = mm)

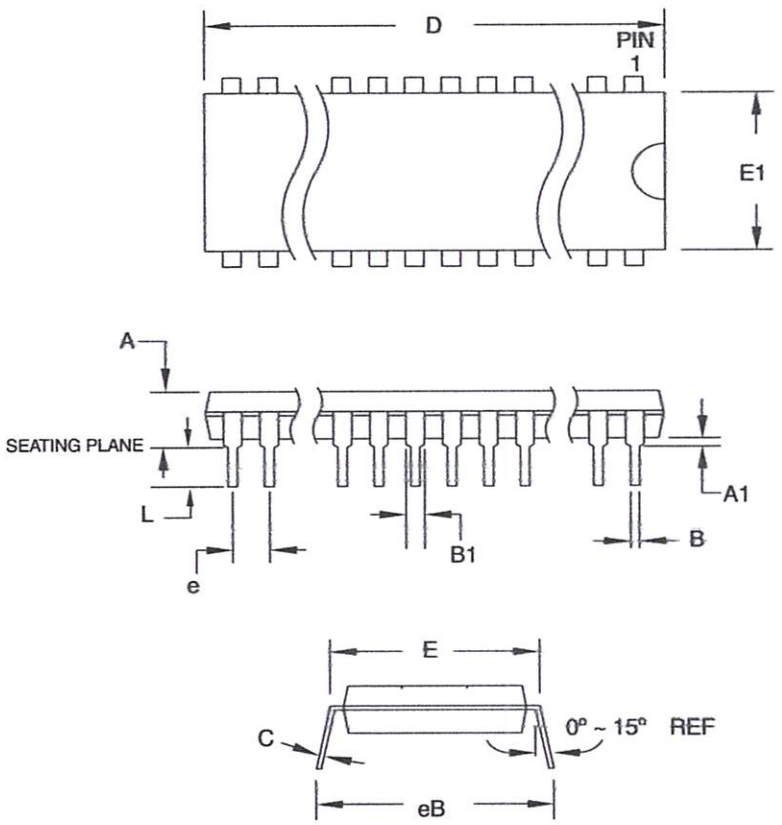
SYMBOL	MIN	NOM	MAX	NOTE
A	4.191	–	4.572	
A1	2.286	–	3.048	
A2	0.508	–	–	
D	17.399	–	17.653	
D1	16.510	–	16.662	Note 2
E	17.399	–	17.653	
E1	16.510	–	16.662	Note 2
D2/E2	14.986	–	16.002	
B	0.660	–	0.813	
B1	0.330	–	0.533	
e	1.270 TYP			

Notes:
1. This package conforms to JEDEC reference MS-018, Variation AC.
2. Dimensions D1 and E1 do not include mold protrusion. Allowable protrusion is .010"(0.254 mm) per side. Dimension D1 and E1 include mold mismatch and are measured at the extreme material condition at the upper or lower parting line.
3. Lead coplanarity is 0.004" (0.102 mm) maximum.

10/04/01

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**44J**, 44-lead, Plastic J-leaded Chip Carrier (PLCC)	44J	B

9.3 40P6 – PDIP



COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	4.826	
A1	0.381	–	–	
D	52.070	–	52.578	Note 2
E	15.240	–	15.875	
E1	13.462	–	13.970	Note 2
B	0.356	–	0.559	
B1	1.041	–	1.651	
L	3.048	–	3.556	
C	0.203	–	0.381	
eB	15.494	–	17.526	
e	2.540 TYP			

Notes:
1. This package conforms to JEDEC reference MS-011, Variation AC.
2. Dimensions D and E1 do not include mold Flash or Protrusion. Mold Flash or Protrusion shall not exceed 0.25 mm (0.010").

09/28/01

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**40P6**, 40-lead (0.600"/15.24 mm Wide) Plastic Dual Inline Package (PDIP)	40P6	B

eadquarters

tmel Corporation
25 Orchard Parkway
an Jose, CA 95131
SA
l: 1(408) 441-0311
ax: 1(408) 487-2600

International

Atmel Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-
Yvelines Cedex
France
Tel: (33) 1-30-60-70-00
Fax: (33) 1-30-60-71-11

Atmel Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Product Contact

Web Site
www.atmel.com

Technical Support
mcu@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Requests
www.atmel.com/literature

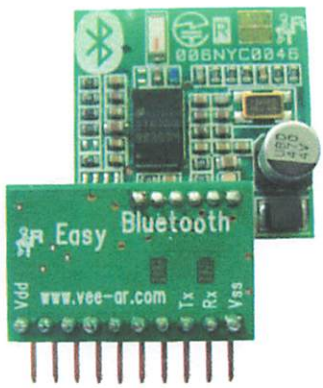
1919D–MICRO–6/08

PARALLAX INC.

Web Site: www.parallax.com
Forums: forums.parallax.com
Sales: sales@parallax.com
Technical: support@parallax.com

Office: (916) 624-8333
Fax: (916) 624-8003
Sales: (888) 512-1024
Tech Support: (888) 997-8267

Easy Bluetooth (#30085)



The Easy Bluetooth is a RoboTech RBT-001 Bluetooth serial module with an adapter specifically designed to be used with the Parallax Board of Education® AppMod Header or breadboard. In addition, the Easy Bluetooth can plug into any 0.1 inch spacing development platform which allows it to be breadboard friendly, and also easy to implement into a soldering application. The module has two parts, the RBT-001 module and the SIP with voltage regulator PCB. With the on-board regulator, the module can be connected to voltages higher than 3.3 VDC, such as the Board of Education regulated supply (+5 VDC) without worry of damaging the unit; while the RX and TX can utilize serial communication at CMOS and TTL levels.

Features

- 1.x & 2.0 Bluetooth Compliant
- Class 2 Operation (nominal range up to 30 meters)
- 10-pin SIP package for breadboard, perfboard, or Board of Education AppMod Header
- On-board regulator for safe operations across various voltages
- CMOS & TTL Compatible

Key Specifications

- Power requirements: 3.3 to 5.5 VDC
- Communication: UART Command/Data Port supports for up to 921.6k baud
- Operating temperature: +32 to +113 °F (0 to +45 °C)
- Dimensions: 1.40 x 1.79 x .49 in (34.41 x 45.65 x 12.51 mm)

Application Ideas

- Control a Boe-Bot via Bluetooth from a PC, Cell Phone, or another Bluetooth module
- Communicate with a device or project wirelessly

Packing List

- RBT-001 Module
- 10-pin SIP with connector Module (pre-assembled with RTB-001)

Quick Start Circuit for the BASIC Stamp 2 and Board of Education

There are a few steps to take to install an Easy Bluetooth module, creating a Bluetooth connection on a PC operating with Windows XP, and finally writing a program for the BASIC Stamp® 2. Once you are done, you will have a working Easy Bluetooth module communicating with the PC via Bluetooth.

Installing the Easy Bluetooth Module

1. Carefully open the Easy Bluetooth package and check that the two small boards are properly plugged into each other as shown in Figure 1.

2. Now plug the Easy Bluetooth module into the AppMod header of the Board of Education: insert the module in the left row of the AppMod Header and notice there are labels on the module that indicate correct pin placement (Rx uses P0, Tx uses P2, Vss with Vss and Vdd with Vdd). Visually confirm the module is inserted correctly using Figure 2 before powering.

3. Connect a communication cable and power supply to the Board of Education and turn the power switch to location 1.

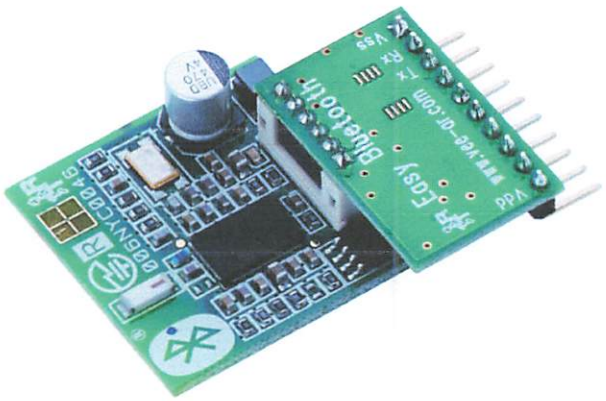


Figure 1

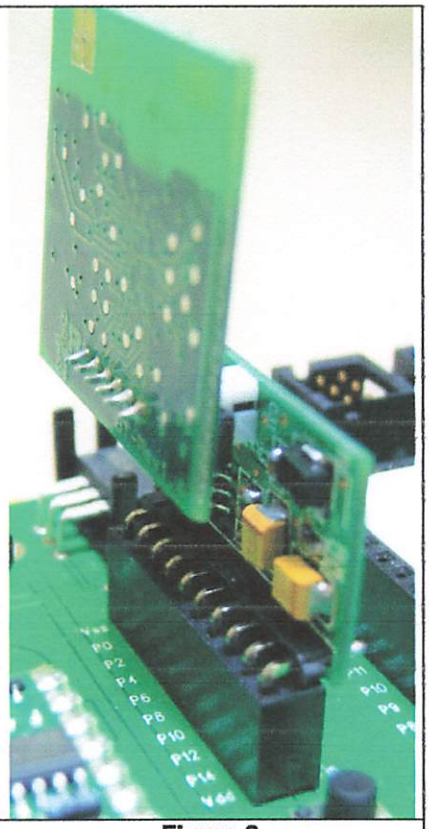


Figure 2

Creating a new Bluetooth Connection

1. Be sure that the Bluetooth dongle that you installed on your PC is working correctly and is turned on (see Bluetooth Dongle user manual for specific instructions to ensure proper operation).

    a. Open Control Panel, click "Printer and Other Hardware" and then click Bluetooth Devices. If in Classic View double click "Bluetooth Devices".

    b. Click "Add".

c. Check "My Device is set up and ready to be found" in the "Add Bluetooth Device Wizard" and click "Next".

d. Select "EasyBT" from the displayed Bluetooth devices and click "Next".

e. Select "Use the passkey found in the documentation" and enter the Passkey Code 0000 (zero four times) and click "Next".

f. Bluetooth Manager should display and designate an Outgoing and Incoming COM port for the Easy Bluetooth device: make a note of the COM ports because these are the ports you will use to communicate with the Easy Bluetooth.

g. Select "Finish" to complete the Bluetooth device configuration.

2. To close the "Bluetooth Devices" window click "OK".

A test for the BASIC Stamp 2 microcontroller

1. The Easy Bluetooth can communicate at numerous baud rates, and with many different microcontrollers. This example uses the AppMod Header on the Board of Education, the BASIC Stamp 2, at 9600 baud.

2. The easiest way to see in your code if a Bluetooth connection has been established is to wait for a byte to be received. You can copy the program below into the BASIC Stamp Editor to ensure a proper connection.

```
' {$STAMP BS2}
' {$PBASIC 2.5}

RX            CON    2      'Receive Pin
TX            CON    0      'Transmit Pin
Baud          CON    84     '9600 Baud

combyte          VAR Byte   'Communication Byte

DEBUG "Use This Screen for Display",CR

DO
  'Wait for a first byte indicating an active Bluetooth connection
  SERIN RX, Baud, [combyte]
  DEBUG combyte
LOOP
```

3. Download the program using Run => Run (ctrl + r) from the BASIC Stamp Editor

4. Keep Current DEBUG Screen open, and open another DEBUG Screen (ctrl +d) from the BASIC Stamp Editor, and select TX COM that the Bluetooth module (refer to COM ports during Bluetooth installation) is using; you should now have 2 DEBUG Terminal windows open. Click in the white area of the DEBUG window of DEBUG Terminal 2 and press any key; if all the steps were correctly done, the same key will be sent back to the DEBUG Terminal 1 window.

Resources and Downloads

You may download the RBT-001 user manual, example source codes, PC applications, and mobile phone application from www.parallax.com.

*Mobile phone must be compatible with the JSR-82 Java API; usually all Nokia and Sony Ericsson phone are compatible, but check users manual for compatibility. Configuration of mobile phone is not supported by Parallax Technical Support.

Device Information

Theory of Operation

Bluetooth is an open wireless protocol for exchanging data over short distances from fixed and mobile devices, creating Personal Area Networks (PANs). It was originally conceived as a wireless alternative to RS232 data cables. It can connect several devices, overcoming problems of synchronization.

There are various versions of Bluetooth communication and the Easy Bluetooth is compatible with versions 1.x and 2.0.

Precautions

The Easy Bluetooth uses the microwave radio frequency spectrum in the 2.4 GHz to 2.4835 GHz range. Maximum power output from this Bluetooth radio 2.5 mW; Class 2 devices are considered less of a potential hazard than mobile phones.

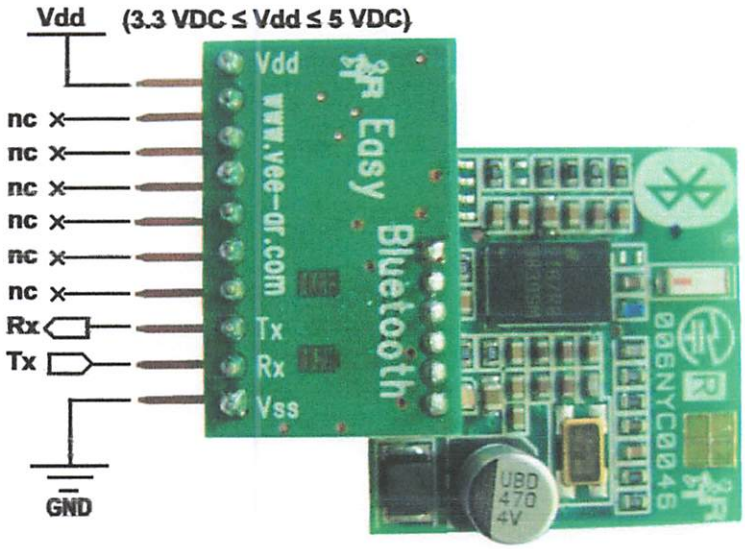
Pin Definitions and Ratings

Pin	Name	Function
1	Vdd	+3.3 to +5 VDC (+5.5 max voltage)
2	RX	Receive Serial Communication (CMOS & TTL Level Compatible)
3	TX	Transmit Serial Communication (CMOS & TTL Level Compatible)
4	NC	Not Connected
5	NC	Not Connected
6	NC	Not Connected
7	NC	Not Connected
8	NC	Not Connected
9	NC	Not Connected
10	Vss	Ground

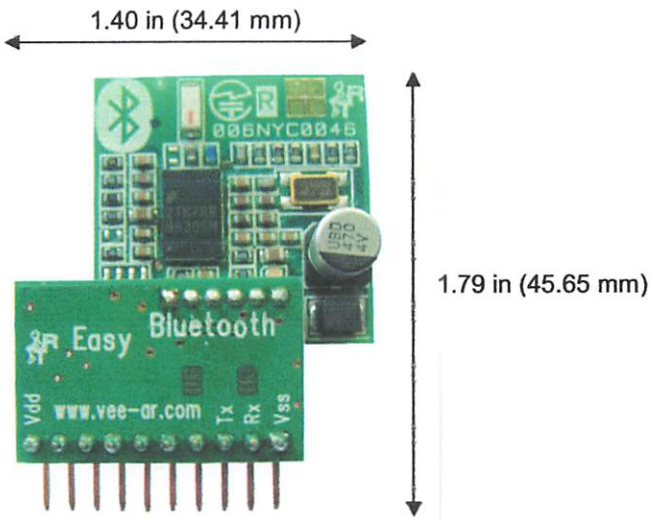
Communication Protocol

UART Command/Data Port supports up to 921.6k non-inverted baud.

Connection Diagram



Module Dimensions



Example Application

This example application uses the RoboTech SRL GUI interface to control a Boe-Bot® robot via Bluetooth.

*A note to mention is that this application uses a standard operational Boe-Bot; so please have an operational Boe-Bot ready while completing this example. For help building a fully operational Boe-Bot, please refer to Robotics with the Boe-Bot Text.

To install the GUI software, complete the following steps:

1. Install the Easy Bluetooth PC software from the "Easy Bluetooth Software.zip" file with all the default and install locations as prompted by the installer. At the end of the installations you will find a new program folder "Easy Bluetooth" under the "Parallax Inc" main folder named with a sub folder named "BoeBot Remote Control", in your Start Menu.

2. Select "BoeBot Remote Control" application from the sub-folder listed in the previous step.

3. Click "Options" and select the COM port that was previously established with the Bluetooth configuration.

4. Download the program below using the BASIC Stamp Editor (ctrl + r) to the BASIC Stamp 2 and then download the program below to the BASIC Stamp 2 using Run => Run or (crtl + r).

5. Disconnect the Boe-Bot from PC

6. Click "Connect" in the BoeBot Remote Control application to gain control of the Boe-Bot.

BASIC Stamp® 2 Program

```
'' =============================================================================
'
'   File..... Easy Bluetooth.bs2
'   Purpose... Control a Boe-Bot with a GUI interface
'   Author.... Technical Support & RoboTech SRL
'   E-mail.... support@parallax.com
'   Started... April 1st 2009
'   Updated... N/A
'
'   {$STAMP BS2}
'   {$PBASIC 2.5}
'
' =============================================================================


' -----[ Program Description ]---------------------------------------

' This program is what is loaded into the BASIC Stamp prior to using the GUI interface
' from RoboTech SLR. It allows a Boe-Bot to be controlled via a GUI interface.
'
' Controls are as followed:
'
'     Up Arrow    = Forward
'     Down Arrow  = Backwards
'     Left Arrow  = Left Turn
'     Right Arrow = Right Turn
```

```
'     Space Bar    = Stop
'
' You can press and hold and release an arrow key for 2 seconds to access a double
' speed of each action. For example, Press Up Arrow, release to make the Boe-Bot move
' faster forward.

' -----[ I/O Definitions ]-------------------------------------------------------

BT_RX           PIN     2               ' RX of the Easy Bluetooth
BT_TX           PIN     0               ' TX of the Easy Bluetooth
LED             PIN     5               ' Indicator LED for Bluetooth Connection

' -----[ Constants ]-------------------------------------------------------------

BT_SPEED        CON     84              ' baud 9600 true UART

' -----[ Variables ]-------------------------------------------------------------

tLeft           VAR     Word            ' Left Servo control pulse durations
tRight          VAR     Word            ' Right Servo control pulse durations
temp            VAR     Word            ' Temp variable

' Buffer array not declared as buffer VAR Word(5) for SERIN functionality.
' It can still be accessed as buffer(0), buffer(1), etc.  However,
' buffer0, buffer1, etc. should be used in SERIN commands with variations
' of WAIT.

  buffer0       VAR     Byte            ' Buffer - Start char = $ff
  buffer1       VAR     Byte            ' Message Index value
  buffer2       VAR     Byte            ' Command
  buffer3       VAR     Byte            ' Argument 1 (return data 1)
  buffer4       VAR     Byte            ' Argument 2 (return data 2)
  buffer        VAR     buffer0         ' For standard array indexing
  msgIndex      VAR     Byte            ' message index
  rxc           VAR     Byte            ' Receive Clear

' -----[ Initialization ]--------------------------------------------------------

Program_Start:
  LOW LED
  DEBUG CLS


  PAUSE 1000                            ' Wait for the RBT-001 radio to be ready.

  msgIndex = 0


  buffer0 = $FF                         ' Connection packet
  buffer1 = msgIndex
  buffer2 = $CC
  buffer3 = 100
  buffer4 = 100
  GOSUB Set_Servo_Speed


  DEBUG CR,"Waiting connection..."  ' wait for connection request
  SERIN BT_RX, BT_SPEED, [WAITSTR buffer \ 3, buffer3, buffer4]

' -----[ Program Code ]-----------------------------------------------------------

DO
```

```
  SELECT buffer2
  CASE $CC                          ' Connect
    HIGH LED
    msgIndex = 0
    DEBUG CR,"Connected"
    GOSUB Reply
  CASE $DD                          ' Disconnect
    LOW LED
    DEBUG CR,"Disconnected"
    GOSUB Reply
    GOTO Program_Start
  CASE $11                          ' Servo
    DEBUG CR,"Servo"
    GOSUB Set_Servo_Speed
    GOSUB Reply
  CASE ELSE                         ' Error
    buffer2 = $EE
    GOSUB Reply
  ENDSELECT

 Resume:                            ' If Message not rcvd, try again

 PULSOUT 13, tLeft                  ' Servo control pulses
 PULSOUT 12, tRight

 SERIN BT_RX, BT_SPEED, 10, Resume, ' Get next command
      [WAITSTR buffer \ 2, buffer2,
      buffer3, buffer4]

 PULSOUT 13, tLeft                  ' Servo control pulses again
 PULSOUT 12, tRight
LOOP

Reply:
  msgIndex = msgIndex + 1           ' Increment message index for reply.
  buffer1 = msgIndex                ' Next message from PC has to use reply's buf[1].
  SEROUT BT_TX, BT_SPEED, [STR buffer \5]


' -----[ Subroutines ]-----------------------------------------------------

Set_Servo_Speed:                    ' Maps to 650 to 850 with 750 stopped.
  tLeft = buffer3 + 650             ' Decode servo speed.
  tRight = buffer4 + 650
RETURN

' -----[ Data ]------------------------------------------------------------

ResetOnOff          DATA 0          ' On/off toggle w/ Reset button
RequestConnect      DATA $FF, 0, 1, 0, 0
ConnectionGranted   DATA $FF, 0, 2, 0, 0
RequestCommand      DATA $FF, 0, 3, 0, 0
ServoSpeeds         DATA $FF, 0, 4, 0, 0
```

louser Electronics

elated Product Links

9-30085 - Parallax 30085