SKRIPSI

DOWNLOADER MIKROKONTROLER AT89S51 MENGGUNAKAN TEKNOLOGI BLUETOOTH

DISUSUN OLEH:

I MADE YUDI ARIYANTO

06.12.905

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

Terakhir , sekali lagi dengan mengucapkan terima kasih kepada semua pihak, dan selanjutnya penulis berharap semoga laporan skripsi ini dapat memberikan mamfaat bagi perkembangan ilmu pengetahuan.

Om Santi, Santi, Santi Om.

Malang, Oktober 2008

Penyusun

ABSTRAKSI

I Made Yudi Ariyanto, 0612905, Jurusan Teknik Elektro S-1, Program Studi Teknik Elektronika, Fakultas Teknologi Industri, Institut Teknologi Nasional Malang, Dosen Pembimbing I : Ir. F. Yudi Limpraptono, MT, Dosen Pembimbing II : Yoseph Dedy Irawan, ST, MT.

Kata Kunci : *Downloader Mikrokontroler, Bluetooth, EB500.*

Saat ini telah banyak downloader-downloader mikrokontroler yang telah beredar dipasaran, baik yang menggunakan interface paralel DB25 maupun yang menggunakan interface serial DB9. Downloader mikrokontroler digunakan untuk melakukan pemrograman terhadap ic mikrokontroler agar ic mikrokontroler tersebut siap digunakan untuk melakukan tugas otomasi yang kita inginkan. Pada skripsi ini, dibuat suatu perangkat downloader mikrokontroler AT89S51 yang pengiriman datanya dilakukan secara bluetooth. Agar pengiriman data secara bluetooth ini dapat terlaksana maka digunakan suatu alat embedded bluetooth eb500 produksi parallax, yang dihubungkan dengan modul downloader ic mikrokontroler AT89S51, sedangkan untuk menentukan dan mengirimkan file-file assembler berextension hex digunakan pc desktop atau laptop yang telah mendukung teknologi bluetooth.

DAFTAR ISI

Halaman

DAFTAR TABEL

DAFTAR GAMBAR

BAB I
PENDAHULUAN

1.1. LATAR BELAKANG

Di era teknologi yang semakin maju ini terus mendorong kita untuk meningkatkan teknologi-teknologi konvensional dan beralih ke teknologi yang simpel tepat guna dan praktis. Perangkat pemrograman mikrokontroler AT89S51 berbasiskan Bluetooth PC merupakan salah satu penerapan dari teknologi modern dan terkini. Saat ini kita telah mengenal beberapa perangkat pemrograman mikrokontroler yang masih menggunakan kabel, perangkat pemrograman tersebut sebenarnya sudah cukup praktis dalam penggunaannya akan tetapi penulis berpikir, kenapa tidak mencoba menggunakan teknologi wireless semacam Bluetooth, misalnya untuk melakukan pemrograman? Tentu bisa lebih praktis, kita tidak perlu lagi repot-repot mencolokkan kabel.

Perangkat pemrograman atau sering disebut dengan downloader berteknologi Bluetooth ini sebenarnya tidak jauh beda dengan perangkat downloader lainnya, hanya saja untuk komunikasi datanya dilakukan secara wireless berteknologi Bluetooth. Pada perangkat downloader ini ditambahkan suatu modul embedded Bluetooth produksi parallax untuk mendukung komunikasi Bluetooth-nya, sedangkan untuk pemroses datanya digunakan mikrokontroler AT89S51.

Selain praktis, downloader ini juga sangat cocok untuk digunakan oleh siapapun, baik yang baru belajar mikrokontroler maupun yang sudah

professional tanpa harus direpotkan oleh pemasangan kabel lagi. Dengan alat ini pengguna bisa melakukan pemrograman dengan mudah dan menyenankan.

1.2. TUJUAN

Tujuan dari perancangan ini adalah untuk membuat suatu alat pemrograman mikrokontroler AT89S51, yang mana untuk komunikasi datanya dapat dikirim secara wireless menggunakan teknologi bluetooth.

1.3. RUMUSAN PERMASALAHAN

Rumusan permasalahan yang terdiri dari perancangan sistem tersebut adalah bagaimana membuat suatu alat pemrograman mikrokontroler AT89S51 yang praktis dan berteknologi bluetooth.

Sehubungan dengan itu maka skripsi ini diberi judul :

"DOWNLOADER MIKROKONTROLER AT89S51 MENGGUNAKAN TEKNOLOGI BLUETOOTH"

1.4. BATASAN MASALAH

Dalam pembahasan ini, ada beberapa batasan yaitu sebagai berikut :

- Menggunakan mikrokontroler AT89S51 sebagai pemroses datanya.
- Untuk komunikasi data secara Bluetooth digunakan modul embedded Bluetooth (eb500) produksi parallax.
- Pemrograman hanya dapat dilakukan menggunakan PC desktop maupun laptop yang telah mendukung teknologi bluetooth.
- Bahasa pemrograman yang digunakan adalah Bahasa Assembler untuk program di mikrokontroler master dan Borland Delphi 7 untuk program di komputer.

1.5. METODOLOGI

Untuk menyelesaikan masalah di atas, akan di tempuh langkah-langkah sebagai berikut :

a. Mempelajari teori-teori dasar dan karakteristik mikrokontroler AT89S51, beserta struktur pemrograman AT89S51.
b. Mempelajari cara kerja embedded Bluetooth eb500 buatan parallax, beserta komunikasi data-nya.
c. Merancang bangun sistem downloader IC mikrokontroler AT89S51 menggunakan teknologi buetooth.
d. Melakukan pengujian terhadap sistem yang telah dibuat.

1.6. SISTEMATIKA PENULISAN

Sistematika penulisan terdiri dari:

a. Bab I. Bab ini berisi tentang pendahuluan. Bab ini terdiri dari Latar Belakang, Tujuan, Rumusan Permasalahan, Batasan Masalah, Metodologi, Sistematika Penulisan.
b. Bab II. Bab ini Berisi tentang teori dasar tentang mikrokontroler AT89S51, Detail dan fungsi port-port pada mikrokontroler AT89S51, Organisasi Memori mikrokontroler AT89S51, Fasilitas Serial Peripheral Interface (SPI) pada mikrokontroler AT89S51, Teori tentang teknologi Bluetooth dan modul embedded Bluetooth EB500 buatan paralax Teori tentang Program bahasa pemrograman DELPHI, dan membahas terori-teori dasar lainnya yang behubungan dengan alat yang dibuat.

c. Bab III. Bab ini berisi tentang Perencanaan alat yang dibuat.

d. Bab IV. Bab ini berisi tentang Analisa dan pengujian alat yang dibuat.

e. Bab V. Bab ini berisi tentang kesimpulan-kesimpulan yang didapat selama perencanaaan dan pembuatan alat.

f. Lampiran-Lampiran. Lampiran berisi tentang Data-data Sheet komponen-komponen yang digunakan untuk pembuatan alat.

BAB II

TEORI DASAR

2.1.Mikrokontroller AT89S51

Mikrokontroller AT89S51 merupakan mikrokontroler yang dapat beroprasi pada tegangan yang rendah dan merupakan mikrokontroller CMOS 8-bit yang memiliki performansi yang tinggi dengan fasilitas flash memori sebesar 4 Kilobyte pada in-system programmable flash memorinya. Mikrokontroller ini diproduksi menggunakan teknologi kepadatan (high-density) memori nonvolatile ATMEL yang kompatibel dengan set instruksi dan pinout standar industri yang diterapkan pada 80C51. Fasilitas On chip flash pada AT89S51 memungkinkan program memory untuk diprogram ulang pada system atau dengan pemrogram memori nonvolatile konvensional. Dengan mengkombinasikan kemampuan CPU 8-bit dengan in-system programmable flash dalam sebuah chip monolitik, mikrokontroller AT89S51 merupakan mikrokontroller yang powerful yang menyediakan flexibilitas yang tinggi dengan harga yang murah untuk beragam aplikasi control embedded.

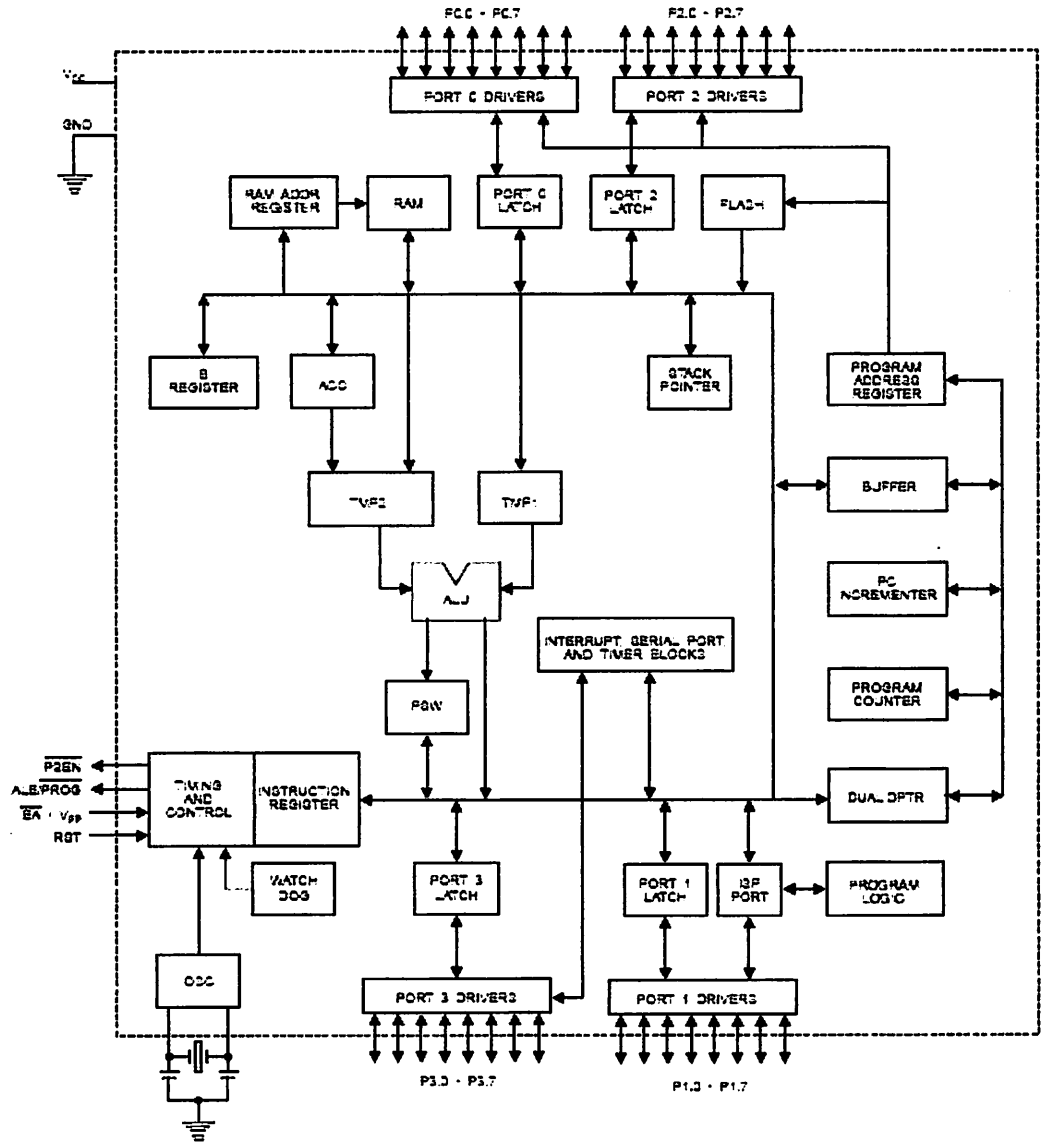
2.1.1. Fasilitas dan Pinout pada AT89S51

Mikrokontroler AT89S51 memiliki berbagai fasilitas penting yang dimiliki oleh AT89S51 sebagai sebuah pengendali, antara lain adalah sebagai berikut:

- Memiliki 4 kilobyte *Flash PEROM* (programmable and Erasable Read only memory) yang dapat diprogram ulang dengan fasilitas SPI (*Serial Programming Interface*). Memori flash PEROM ini dapat bertahan untuk dihapus dan ditulis ulang kira-kira sebanyak 1000 kali.
- Dapat beroprasi pada kisaran tegangan 4-5 VDC.
- Mendukung pewaktuan 0 Hz sampai 24 MHz clock speed.
- Fasilitas pengamanan program memori sebanyak 3 level pengamanan data (program memory lock).
- Memiliki RAM internal sebanyak 128 byte.
- Memiliki 32 input dan output (port pins) yang dapat deprogram sebagai jalur masukan dan keluaran. Terbagi dalam 4 port parallel dan 1 port sereial yang dapat berjalan dalam mode 2 arah (full duplex).
- Memiliki fasilitas timer dan counter sebanyak 3 buah, masing-masing dapat difungsikan dengan perhitungan 16 bit.
- Memiliki fasilitas watchdog timer yang dapat diprogram sesuai dengan keinginan.
- Memiliki 2 buah DPTR (Data Pointer). Data pointer berguna untuk pengalamatan memori dengan alamat 16 bit.

Seperti halnya jenis mikrokontroler lainnya, mikrokontroler AT89S51 memiliki dimensi yang cukup kecil. Dimensinya yang cukup

kecil membuat mikrokontroler sangat berguna dalam perancangan embedded control applications.



Gambar 2-1, Blok Diagram AT89S51[1]

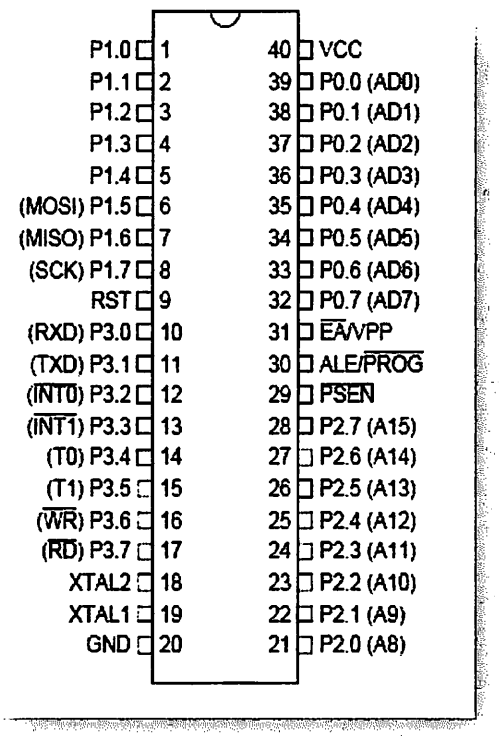
Mikrokontroler AT89S51 dibangun dalam sebuah chip yang memiliki 40 buah pin. Penjelasan dari masing-masing pin adalah sebagai berikut:

- ***Power pin :*** AT89S51 berjalan dengan baik pada kisaran tegangan 4 - 5.5 VDC , *Pin 40* berfungsi untuk supplay tegangan **Vcc**, dan *Pin 20* berfungsi untuk system groundingnya.
- ***Input/Output (I/O) pin:*** AT89S51 memiliki 4 buah port parallel dan 1 buah port serial. Total dari pin input/output adalah 32 buah jalur. Port 0 menempati 8 buah pin dari nomor pin 32 sampai 39. Port 1 menempati 8 buah pin dari nomor pin 1 sampai 8. Port 2 menempati 8 buah pin dari nomor pin 21 sampai 28. Port 3 menempati 8 buah pin dari nomor pin 10 sampai 17. Perlu diketahui juga bahwa beberapa pin mempunyai fungsi khusus, termasuk pin-pin serial port yang sebenarnya juga menempati port 3.
- ***Reset pin:*** Input dari tombol reset mikrokontroler. Untuk melakukan reset terhadap proses yang sedang dilakukan mikrokontroler, maka diperlukan pemberian nilai 1 (high) pada pin ini selama 2 siklus mesin. Ada beberapa teknik melakukan reset pada mikrokontroller.
- ***Memory addressing pin:*** Pin-pin ini berfungsi apabila mikrokontroler melakukan pengaksesan terhadap alamat memori eksternal. Port yang dilibatkan dalam hal ini adalah port 2, sedangkan pin-pin fungsional yang terlibat adalah pin WR (pin nomor 36) dan RD (pin Nomor 37).

- ***ALE/PROG (pin nomor30):*** *ALE (Address Latch Enable)* adalah pin output untuk melakukan proses *Latching* terhadap pengaksesan byte rendah (low byte) memori eksternal. Latching adalah proses pergantian fungsi antara pengiriman dan penerimaan alamat dan data memori eksternal. Selain itu, pin ini juga memegang peranan dalam proses pengisian program pada flash PEROM. ALE hanya bereaksi dengan perintah MOVX dan MOVC saja. ALE dapat diaktifkan dan dinonaktifkan 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.
- ***EA/VPP (pin nomor 31):*** *EA (External Access Enable)* harus dihubungkan ke Ground (pin 20) apabila ingin melakukan akses ke program memori pada memori external. Namun apabila program memori ada di dalam memori internal mikrokontroler, dan tidak diperlukan akses ke memori eksternal maka EA harus diberi nilai HIGH (1) dengan cara dihubungkan ke VCC (pin 40). Pin EA berfungsi ganda sebagai penerima tegangan 12 volt DC (VPP) dalam proses pengisian program pada flash PEROM.
- ***PSEN (pin nomor 29):*** PSEN (Programmable Store Enable) berfungsi dalam pemberian sinyal strobe apabila mikrokontroler melakukan pengaksesan memori program

yang berada pada memori eksternal. PSEN diaktifkan sebanyak 2 kali setiap satu siklus mesin pada saat mikrokontroler mengakses program memori dari memori eksternal.

- ***CPU Clock pin:*** berfungsi dalam pewaktuan prosesor dari mikrokontroler. CPU Clock pin ini terdiri dari XTAL 1 (pin nomer 18) yang merupakan masukan clock dari kaki osilatot Kristal ke untai penguat pembalik (inverting) dan untai pewaktu internal mikrokontroler, dan yang kedua adalah XTAL2 yang berada pada pin nomor 19 yang merupakan keluaran dari untai penguat pembalik internal menuju osilator Kristal.



Gambar 2-2, Skema PIN OUT Mikrokontroler AT89S51[1]

2.1.2. Detail dan Fungsi Port parallel AT89S51

Berikut adalah penjelasan yang lebih detail mengenai struktur dan fungsi empat buah port parallel pada AT89S51:

- ***Port 0:*** Port 0 terdiri dari 8 buah pin yang berurutan dari pin nomor 32 sampai 39. Port 0 adalah salah satu jalur input/output dua arah (*bidirectional*) yang dapat diprogram sesuai kebutuhan. Selain berfungsi sebagai input/output, port 0 juga memiliki fungsi khusus dalam pengalamatan 8 bit atau 16 bit alamat memori eksternal. Dalam hal pengalamatan memori eksternal, port 0 berfungsi memberikan 8 bit pertama *(low byte)*. Port 0 memiliki pull-up internal. Port 0 juga berfungsi untuk menerima kode program pada saat flash PEROM mikrokontroler diisi, dan juga memberikan verifikasi terhadap kode program tersebut. Dalam proses pemberian verifikasi, port 0 memerlukan bantuan pull-up eksternal.
- ***Port 1:*** Port 1 juga merupakan 8 bit jalur input/output dua arah seperti halnya port 0. Port 1 terletak pada pin nomer 1 sampai 8. Port 1 memiliki pull-up internal. Beberapa pin dari port 1 memiliki fungsi khusus. Port 1 juga berperan dalam proses pengisian program flash PEROM dan proses verifikasi, yaitu sebagai penerima byte alamat pertama (low

order address byte). Fungsi khusus pin-pin dalam port 1 tersebut dapat dilihat pada tabel berikut:

Tabel 2-1, Fungsi Port 1

Nama Pin	Nomor pin	Kegunaan Khusus
P1.0	1	T2 (input atau counter 2 dari sumber clock eksternal)
P1.1	2	T2EX (Timer/Counter 2 capture/reload trigger dan diection control
P1.4	5	SS (untuk memilih slave port)
P1.5	6	MOSI (Master Data output, untuk fasilitas SPI)
P1.6	7	MISO (Master Data Input, untuk fasilitas SPI)
P1.7	8	SCK (master Clock Output, masukan untuk pin slave clock untuk fasilitas SPI channel)

- ***Port 2:*** Port 2 terdiri dari 8 buah pin yang berurutan dari pin nomor 21 sampai 27. Port 2 adalah salah satu jalur input/output dua arah yang dapat diprogram sesuai kebutuhan. Port 2 memiliki internal pull-up. Dalam pengaksesan memori eksternal, port 2 berperan dalam pengalamatan byte terakhir terhadap memori eksternal (high Order Address Byte) yang memiliki alamat sampai 16 bit. Sedangkan pada saat pengaksesan memori eksternal 8 bit, port 2 berfungsi untuk mengeluarkan isi dari SFR P2. Pada saat proses pengisian program flash PEROM dan proses verifikasi, port 2 berfungsi menerima byte alamat kedua (High Order Address Byte) dan beberapa sinyal control lainya.

- ***Port 3:*** Port 3 terdiri dari 8 buah pin yang berurutan dari pin nomor 10 sampai 17. Port 3 adalah salah satu jalur input/output dua arah yang dapat diprogram sesuai kebutuhan. Port 3 memiliki internal pull-up. Dalam proses pemrograman flash PEROM dan verifikasi, port 3 berfungsi untuk menerima beberapa sinyal kontrol. Port 3 adalah port paling spesial dari AT89S51 karena memiliki berbagai fungsi sebagai berikut:

Tabel 2-2, Fungsi Port 3

Nama Pin	Nomor pin	Kegunaan Khusus
P3.0	10	RXD (port penerima komunikasi serial)
P3.1	11	TXD (port pengirim komunikasi serial)
P3.2	12	INT0 (masukan untuk fasilitas intrupsi eksternal 0)
P3.3	13	INT1 (masukan untuk fasilitas intrupsi eksternal 1)
P3.4	14	T0 (masukan untuk fasilitas timer eksternal 0)
P3.5	15	T1 (masukan untuk fasilitas timer eksternal 1)
P3.6	16	WR (external data memory write strobe)
P3.7	17	RD (external data memory read strobe)

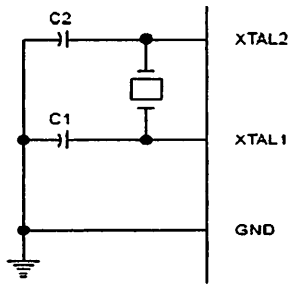
2.1.3. Port Serial AT89S51

Port serial pada mikrokontroler dapat digunakan dalam mode *full-duplex,* artinya dapat menerima dan mengirim data serial secara bersamaan. Penerimaan dan pengiriman data port serial melalui sebuah register yang disebut SBUF pada penerima dan pengirim data serial *(Serial Data Buffer).* Dengan adanya SBUF, maka dimungkinkan juga untuk

melakukan pembacaan atau pengiriman data pada lebih dari satu byte data yang datang atau terkirim secara terpisah dan berurutan. Port serial dapat bekerja pada empat mode yang berbeda, 1 mode diantaranya bersifat sinkron, dan 3 mode lainya bersifat asinkron. Perbedaan antara mode komunikasi sinkron dan asinkron adalah bahwa pada pengiriman data sinkron, pengiriman data akan dikirim secara bersamaan dengan clock, sedangkan pada data asinkron tidak dikirim bersama dengan clock. Rangkaian receiver harus membangkitkan sendiri clock yang diperlukan dalam pembacaan data serial tersebut. Keempat data tersebut di set dengan menggunakan SFR (Special Data Register) bernama ***SCON***.

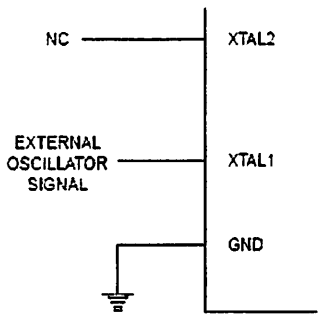
2.1.4. Sistem Pewaktuan (Clock) mikrokontroler

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



Gambar 2-3, Internal Clock Mikrokontroler[1]

Selain dengan memamfaatkan osilator internal (on-chip oscillator) seperti diatas, anda juga dapat menggunakan sumber pewaktuan (clock) eksternal. Cara pemakaian dapat dilihat pada gambar berikut:



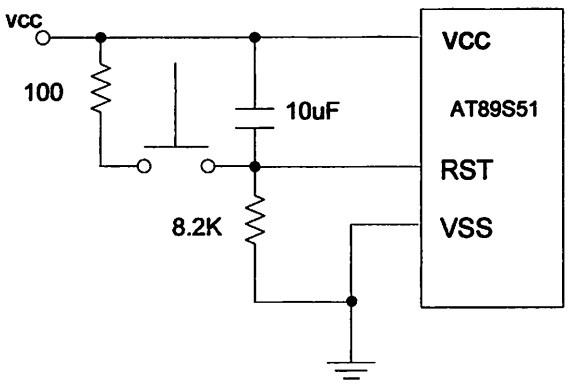
Gambar 2-4, External Clock Mikrokontroler[1]

Perlu diperhatikan bahwa semakin cepat clock yang diberikan, maka akan semakin cepat pula pemrosesan data atau oprasi yang dapat dilakukan oleh CPU mikrokontroler.

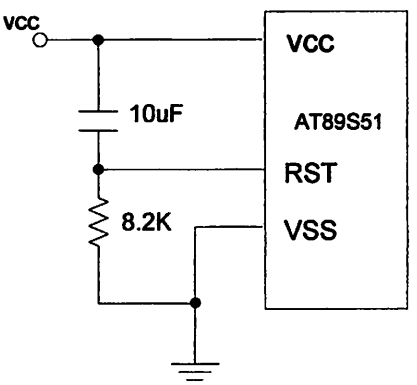
2.1.5. Fasilitas Reset pada Mikrokontroler

Fasilitas *RESET* mikrokontroler yang berada pada pin nomer 9 pada skema hardware mikrokontroler adalah satu-satunya fasilitas untuk melakukan reset pada program mikrokontroler, fungsinya menyerupai tombol reset pada PC, yakni menghentikan semua aktivitas yang dilakukan oleh CPU dan melakukan restart terhadap rangkaian kode program kembali lagi dari baris awal program. Cara melakukan reset adalah dengan memberikan sinyal 1 (high) terhadap pin 9 mikrokontroler selama 2 siklus mesin, dan kemudian mengembalikannya ke kondisi semula (low). Ada

dua macam system reset yang dapat dihubungkan ke pin 9 (RST) mikrokontroler. Dua system tersebut adalah manual reset dan power-ON reset. Perbedaan antara keduanya adalah jika pada manual reset kita dapat menggunakan reset dengan menggunakan tombol atau saklar yang tersendiri, sedangkan apabila kita menggunakan mode power-on reset, reset dilakukan dengan mematikan sementara suplai power (daya) ke mikrokontroler, dan kemudian dihidupkan kembali. Dilakukan pada saat pertama kali program dijalankan. Rangkaian elektronik untuk kedua mode diatas dapat dilihat sebagai berikut:



Gambar 2-5, Manual Reset



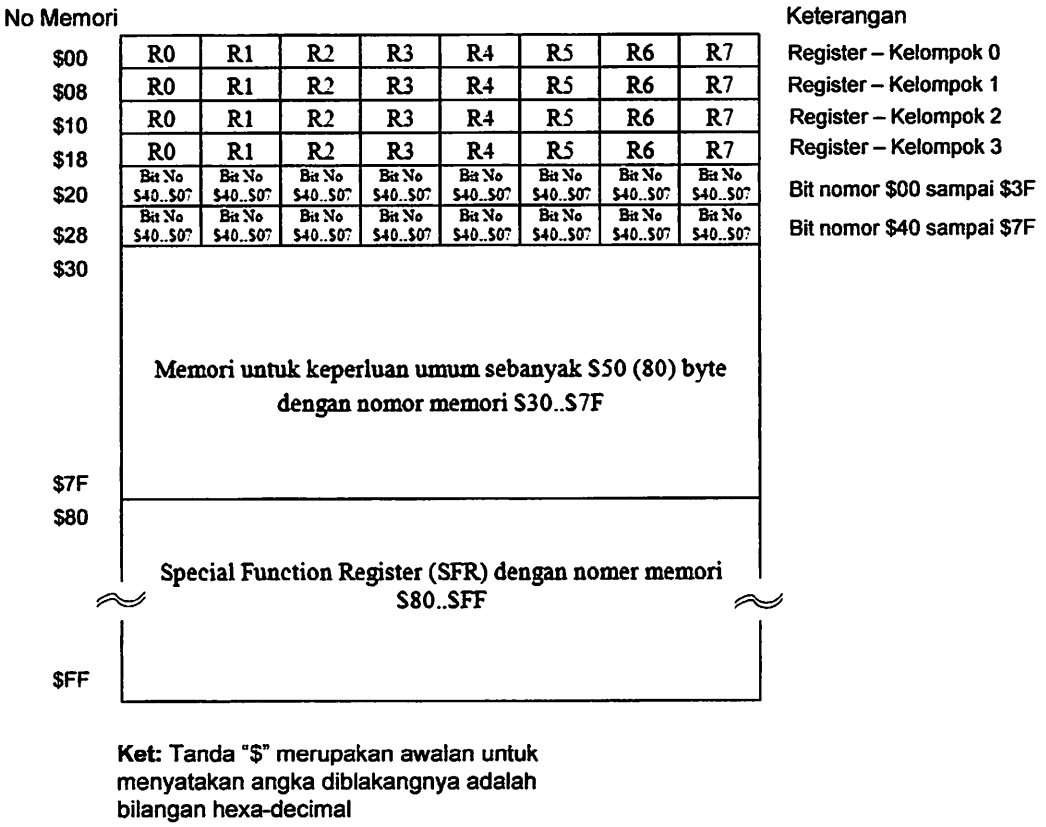
Gambar 2-6, Power-ON Reset

2.1.6. Organisasi Memori

Seperti yang kita ketahui bahwa ada 2 tipe memori, RAM *(Random Access Memory)* dan ROM *(Read Only Memory)*. Secara fungsi, pemisahan memori ini juga dapat digolongkan sebagai *memori data* (RAM) dan *memori program* (ROM). Adapula RAM eksternal yang bersifat optional, artinya bisa dipasang dan dipakai atau bisa juga tidak. Memori program hanya bisa dibaca saja, sedangkan memori data bisa dibaca bisa juga ditulisi.

2.1.6.1. Memori Internal

Memori RAM internal pada AT89S51 sebanyak 128 byte dapat dipetakan sebagai berikut :



Ket: Tanda "$" merupakan awalan untuk menyatakan angka diblakangnya adalah bilangan hexa-decimal

Gambar 2-7, Peta memori RAM Internal

Memori data dibagi menjadi dua bagian, memori nomor $00 sampai $7F (Gambar 2-7 sebelah atas) merupakan memori seperti RAM selayaknya walaupun beberapa mempunyai kegunaan yang spesifik, sedangkan memori nomor 80H sampai FFH (Gambar 2-7 sebelah bawah) dipakai sangat khusus yang dinamakan sebagai *Special Funcion Register*.

Momori data nomor 00H sampai 7FH bisa dipakai sebagai memori penyimpanan data biasa, dibagi menjadi 3 bagian, yaitu:

- ✓ Memori nomor 00H sampai 18H selain sebagai memori data biasa, bisa pula dipakai sebagai *register serba guna* (General Purpose Register).
- ✓ Memori nomor 20H sampai 2FH selain sebagai memori data biasa, bisa dipakai untuk menyimpan informasi dalam level bit (bit-addressable). Setiap byte memori di daerah ini bisa dipakai menampung 8 bit informasi yang masing-masing di nomori tersendiri, dengan demikian dari 8 byte memori yang ada bisa dipakai untuk menyimpan 64 bit (8 x 8 bit).
- ✓ Memori nomor 30H sampai 7FH(sebnyak 80byte) merupakan memori data atau dipakai untuk stack.

2.1.6.2. Memori Eksternal

Memori eksternal bersifat opsional atau bisa dipakai bila diperlukan dan tidak apabila dirasa memori internal sudah mencukupi. Memori eksternal dapat di akses maksimal bernilai 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 adalah printah MOVX.

2.1.7. Special Function Register (SFR)

Register – register internal dalam sebuah mikrokontroler pada umumnya dapat diakses dengan menggunakan set instruksi pemrograman. 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 di ujung dari atas dari alamat memori RAM, yaitu pada antara 80H sampai FFH. Ada beberapa alamat SFR yang tidak ada isinya. Meski letaknya pada RAM, namun ada beberapa alamat yang tidal dapat digunakan. Apabila dilakukan proses pembacaan maka akan dikembalikan ke nilai acak, sedangkan apabila ditulisi akan tidak tersimpan dengan benar pada alamat – alamat tersebut. Berikut adalah gambar peta memori SFR.

	8 Bytes								
F8									FF
F0	B								F7
E8									EF
E0	ACC								E7
D8									DF
D0	PSW								D7
C8	(T2CON)	(T2MOD)	(RCAP2L)	(RCAP2H)	(TL2)	(TH2)			CF
C0									C7
B8	IP								BF
B0	P3								B7
A8	IE								AF
A0	P2								A7
98	SCON	SBUF							9F
90	P1								97
88	TCON	TMOD	TL0	TL1	TH0	TH1			8F
80	P0	SP	DPL	DPH				PCON	87

Gambar 2-8, Peta Memori Special Function Register (SFR)[1]

Beberapa bagian-bagian penting dari SFR adalah sebagai berikut:

- ✓ **Accumulator** (ACC): *Accumulator* merupakan sebuah register yang berfungsi untuk menampung hasil pengolahan data dari banyak instruksi ASM51. Pada program ditulis dengan A.
- ✓ **Program Status Word (PSW):** *PSW* terletak pada alamat 0DH pada segmen memori berisi SFR. Byte PSW terdiri dari sederetan status bit, seperti terlihat pada tabel berikut:

Tabel 2-3, Peta Bit Register PSW

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CY	AC	F0	RS1	RS0	OV	-	P

Keterangan:

- **Carry Flag (CY) :** *Carry flag* memiliki dua buah fungsi. Fungsi pertama adalah sebagai penanda oprasi aritmatika. Disini CY dapat bernilai logika 1 apabila terjadi carry (kelebihan) bit pada proses penjumlahan dengan instruksi ADD, atau borrow (kekurangan) bit pada saat oprasi pengurangan dengan instruksi SUBB. Fungsi kedua adalah bit CY dapat berguna sebagai "akumulator" Boolean. Artinya dapat difungsikan untuk instruksi bit dengan peran mirip dengan akumulator.
- **Auxilliary Carry Flag (AC) :** *Auxlilliary Carry Flag* akan bernilai 1 (high) apabila terjadi penambahan bilangan berbasis BCD dengan instruksi DA, dan terjadi kelebihan bit pada nibble bawah (bit 0-3) dalam range 0AH sampai dengan 0FH.
- **Flag 0 (F0) :** *Flag 0* adalah flag kosong yang dapat digunakan dalam aplikasi apa saja, tergantung pada keinginan programmer.
- **Register Bank Select (RS) :** *Register Bank Select* terdiri dari 2 bit, yakni RS0 dan RS1, fungsinya adalah untuk menentukan bank register mana diantara Ri (R0 sampai

R7) yang aktif pada waktu tertentu. Untuk masing-masing register R0 sampai R7, ada 4 bank yang dikendalikan oleh RS. Bit RS0 dan RS1 di reset menjadi 0saat dilakukan reset pada mikrokontroller.

- **Overflow Flag (OV) :** *Flag OV* diset menjadi bernilai 1 apabila terjadi overflow pada saat dilakukan instruksi aritmatika. Namun operasi aritmatika yang berpenganruh terhadap flag OV hanyalah yang menggunakan bilangan bertanda *(signed number)* saja, dan tidak berlaku untuk bilangan tidak bertanda *(unsigned number)*. Dikatakan terjadi overflow apabila hasil dari operasi aritmatika tersebut bernilai lebih besar dari (+127) atau kurang dari (-128).
- **Parity Bit (P) :** *Parity Bit (P)* secara otomatis akan diset bernilai satu atau direset bernilai 0 pada tiap siklus mesin untuk melakukan pengecekan parity ganap *(even parity checking)* terhadap akumulator. Artinya jumlah dari angka 1 dari nilai akumulator ditambah dengan 1 dari P haruslah berjumlah genap. Parity bit umumnya dimanfaatkan dalam antamuka komunikasi serial mikrokontroler.

- ✓ **Register B:** *Register B* digunakan pada operasi perkalian dan pembagian. Pada instruksi-instruksi yang lain berfungsi seperti register umumnya.
- ✓ **Stack Pointer (SP) :** *Stack Pointer (SP)* adalah register 8 bit yang memiliki lokasi pada alamt 81H dalam segmen memori SFR. Fungsi utamanya adalah berperan dalam oprasi *stack* terkait dengan instruksi PUSH dan POP dalam pemrograman mikrokontroler, instruksi push akan mengakibatkan penambahan SP, sedangkan POP menyebabkan pengurangan nilai SP. Selain kedua instruksi diatas, SP juga berperan dalam instruksi CALL dan RET atau RETI, yakni untuk menyimpan dan mengembalikan nilai PC (*Progam Counter*).
- ✓ **Dual Data Pointer (DPTR) :** *DPTR* atau data pointer adalah satu-satunya register pada mikrokontroler yang dapat memproses nilai sebesar 16 bit. Umumnya DPTR digunakan untuk mengakses alamat memori eksternal. Pada segmen memori SFR letaknya terpisah menjadi 2, yaitu DPH *(high byte)* dan DPL *(low byte).*
- ✓ **Register Port Paralel :** register ini adalah sarana antarmuka internal yang vital bagi sebuah mikrokontroler dalam hubungannya dengan dunia luar. Register port 0 terletak pada alamat 80H, register port 1 terletak pada alamat 90H, register port2 terletak pada alamat A0H, dan register port 3 terletak pada alamat B0H. Semua register port parallel berfungsi

sebagai register input atau register output (register I/O). Namun register port 0, port 2, port 3 memiliki fungsi lain. Ketiga register ini tidak dapat digunakan sebagai register I/O apabila mikrokontroler sedang melakukan akses terhadap memori eksternal, melakukan oprasi intrupsi, dan komunikasi serial.

✓ **Register Port Serial :** AT89S51 memiliki fasilitas On-Chip serial port untuk berkomunikasi dengan dunia luar. SBUF (alamat 99H) dan SCON (alamat 9AH) memegang peranan terpenting dalam komunikasi serial mikrokontroler, masing-masing terdiri atas 1 byte Register, register SBUF berkaitan dengan data, sedangkan register SCON lebih kearah kontrol. Selain 2 buah register diatas adapula 1 bit yang terletak pada register PCON yang berperan dalam komunikasi serial, bit tersebut disebut dengan SMOD (serial mode).

	SCON.7	SCON.6	SCON.5	SCON.4	SCON.3	SCON.2	SCON.1	SCON.0
98H	SM0	SM1	SM2	REN	TB8	RB8	TI	RI

Gambar 2-9, Peta Bit Register SCON

Keterangan: SM0: *Serial Port Mode bit 0*, bit Pengatur Mode Serial

SM1: *Serial Port Mode bit 1*, bit Pengatur Mode Serial

SM2: *Serial Port Mode* bit 2, bit untuk mengaktifkan komunikasi multiprosesor pada kondisi set.

REN: *Receive Enable*, bit untuk mengaktifkan penerimaan data dari Port Serial pada kondisi set. Bit ini di set dan clear oleh perangkat lunak.

TB8: *Transmit bit 8*, bit ke 9 yang akan dikirimkan pada mode 2 atau 3. Bit ini di set dan clear oleh perangkat lunak

RB8: *Receive bit 8*, bit ke 9 yang diterima pada mode 2 atau 3. Pada Mode 1 bit ini berfungsi sebagai stop bit.

TI: *Transmit Interrupt Flag*, bit yang akan set pada akhir pengiriman karakter. Bit ini diset oleh perangkat keras dan di clear oleh perangkat lunak

RI: *Receive Interrupt Flag*, bit yang akan set pada akhir penerimaan karakter. Bit ini diset oleh perangkat keras dan di clear oleh perangkat lunak.

Tabel 2-4, Mode Komunikasi Serial Mikrokontroler

SM0	SM1	Mode	Deskripsi
0	0	0	Shift Register 8 bit
0	1	1	UART 8 bit dengan baud rate yang dapat diatur
1	0	2	UART 9 bit dengan baud rate permanen
1	1	3	UART 9 bit dengan baud rate yang dapat diatur

Pada mode ini komunikasi data dilakukan secara 8 bit data asinkron yang terdiri 10 bit yaitu 1 bit start, 8 bit data dan 1 bit stop. Baud Rate pada mode ini dapat diatur dengan menggunakan Timer 1. Tidak seperti pada mode 0, pada mode ini yang merupakan mode UART, fungsi-fungsi alternatif dari P3.0/RXD dan P3.1/TXD digunakan. P3.0 berfungsi sebagai RXD yaitu kaki untuk

penerimaan data serial dan P3.1 berfungsi sebagai TXD yaitu kaki untuk pengiriman data serial. Hal ini juga berlaku pada mode-mode UART yang lain seperti mode 2 dan mode 3.

Pengiriman data dilakukan dengan menuliskan data yang akan dikirim ke Register SBUF. Data serial akan digeser keluar diawali dengan bit start dan diakhiri dengan bit stop dimulai dari bit yang berbobot terendah (LSB) hingga bit berbobot tertinggi (MSB). Bit TI akan set setelah bit stop keluar melalui kaki TXD yang menandakan bahwa proses pengiriman data telah selesai. Bit ini harus di-clear oleh perangkat lunak setelah pengiriman data selesai.

Penerimaan data dilakukan oleh mikrokontroler dengan mendeteksi adanya perubahan kondisi dari logika high ke logika low pada kaki RXD di mana perubahan kondisi tersebut adalah merupakan bit start. Selanjutnya data serial akan digeser masuk ke dalam SBUF dan bit stop ke dalam bit RB8. Bit RI akan set setelah 1 byte data diterima ke dalam SBUF kecuali bila bit stop = 0 pada komunikasi multiprosesor (SM2 = 1). Untuk menentukan baudrate bisa dilihat pada ilustrasi dibawah:

$$\text{Baud rate} = \frac{\text{Freq. Oscilator}}{32 \times 12 \times (256 - TH)} \times K$$

Dimana K adalah konstanta bernilai 1 bila SMOD = 0 dan bernilai 2 bila SMOD=1, contoh:

- menentukan isi Timer TH1 untuk baud rate=9600, x-tal yang dipasang = 11059200Hz, SMOD=0, maka:

$$TH1=256 - \frac{K \times F.Osc}{384 \times baud\ rate} = 256 - \frac{1 \times 11059200}{384 \times 9600} = 256 - 3 = 253 = FDH$$

SMOD berada pada register PCON bit ke 7. Karena register PCON tidak bisa dialamati per bit maka untuk menjadikan SMOD menjadi logika '1' dapat dilakukan dengan cara menuliskan instruksi : ***ORL PCON,#80H.***

Seperti yang telah kita ketahui bahwa ada 4 macam tipe oprasi komunikasi serial yang didukung oleh mikrokontroler. Tiga diantaranya menganut sistem komunikasi asinkron, dan satunya lagi menganut sistem sinkron. Berikut adalah penjelasan dari masing-masing mode tersebut:

- **Mode 0 :** data serial masuk dan keluar melalui RXD. TXD mengeluarkan sinyal clock. 8 bit data dikirim/diterima dengan bit LSB (Least Significant Bit) yang pertama. Baud rate tetap pada 1/12 frekuensi osilator.
- **Mode 1 :** 10 bit dikirim melalui TXD atau diterima melalui RXD yang terdiri dari sebuah start bit (0), 8 bit data (LSB pertama), dan sebuah stop bit (1). Pada penerimaan, stop bit menuju RB8 pada SFR SCON. Baudrate variabel.
- **Mode 2 :** 11 bit dikirim melalui TXD atau diterima melalui RXD, sebuah start bit (0), 8 bit data (LSB pertama), bit data ke 9 yang terprogram, dan sebuah stop bit (1). Pada saat pengiriman, bit data ke 9 (TB8 pada SCON) dapat diberi nilai 0 atau 1. Sebagai contoh bit parity (P pada PSW) dapat dipindahkan ke TB8. Pada penerimaan, bit data ke 9 masuk ke

RB8 pada SCON sedangkan stop bit diabaikan. Baud rate dapat diprogram 1/32 atau 1/64 frekuensi osilator.

- **Mode 3 :** 11 bit dikirim melalui TXD atau diterima melalui RXD, sebuah start bit (0), 8 bit data (pertama LSB), bit data ke 9 yang terprogram dan sebuah stop bit (1). Sebenarnya Mode 3 sama seperti Mode 2, namun baud rate Mode 3 variabel.

✓ **Register Interupsi :** AT89S51 memiliki enam buah sumber interupsi yang esensial, masing-masing adalah 2 buah intrupsi eksternal, 3 buah instrupsi timer, dan 1 buah instrupsi serial. Instrupsi akan dinonaktifkan setelah dilakukan reset terhadap operasi mikrokontroler, dan akan dibangkitkan lagi melalui software yaitu dengan register IE *(Interupt Enable)* pada SCON. Sedangkan kontrol dan prioritas intrupsi akan ditentukan dengan register IP *(Interrupt Priority)* pada SCON. Register IE terletak pada alamat dan register IP diletakan pada alamat pada segmen memori SFR. Kedua register ini dapat dialamati secara byte (8 bit) ataupun secara bit.

Tabel 2-5, Peta Bit Register IE

IE.7	IE.6	IE.5	IE.4	IE.3	IE.2	IE.1	IE.0
EA	-	ET2	ES	ET1	EX1	ET0	EX0

Keterangan: -> **EA** pada bit IE.7 berfungsi untuk mengaktifkan atau menonaktifkan semua sumber intrupsi. Semua sumber intrupsi mikrokontroler dibuat menjadi

aktif dengan cara memberikan nilai 1 pada bit EA. Sedangkan semua sumber intrupsi dapat di-nonaktifkan dengan cara menuliskan nilai 0 pada bit EA.

-> **Bit IE.6** tidak digunakan dan sebaiknya tidak menuliskan nilai apapun ke dalamnya. Bit ini akan digunakan dalam pengembangan produk mikrokontroler.

-> **Bit ET2** (IE.5, ET1 (IE.3), dan ET0 (IE1) merupakan bit intrupsi timer. Masing-masing adalah intrupsi timer 2, timer 1, dan timer 0. Intrupsi timer 2 dihasilkan melalui logika OR antara bit TF2 dan EXF2 yang terletak pada register T2CON.

-> **ES (IE.4)** adalah bit yang menangani sumber intrupsi dari port serial mikrokontroler. Memberikan nilai 1 pada bit ini berarti mengaktifkan intrupsi serial; mikrokontroler.

-> EXO (IE.2) dan EX1 (IE.0) masing-masing adalah interrupsi eksternal dari yang dibangkitkan melalui pemberian nilai 0 pada P3.2 (INT0) dan pin (INT1) pada mikrokontroler.

Dengan demikian diketahui bahwa untuk mengektifkan sumber-sember intrupsi mikrokontroler AT89S51 dapat dilakukan dengan

memberikan nilai 1 pada tiap-tiap bit IE yang mewakilinya atau dapat juga memberikan nilai 1 pada EA untuk mengaktifkan semua sumber intrupsi. Sedangkan untuk menonaktifkan ialah dengan memberikan nilai 0.

Tabel 2-6, Peta Bit Register IP

IP.7	IP.6	IP.5	IP.4	IP.3	IP.2	IP.1	IP.0
-	-	PT2	PS	PT1	PX1	PT0	PX0

Keterangan: -> **PT2 (IP.5)** Adalah bit yang berfungsi untuk menentukan nilai prioritas sumber intrupsi Timer2.

-> **PS (IP.4)** Adalah bit yuang berfungsi untuk menentukan level prioritas sumber intrupsi port serial.

-> **PT1 (IP.3)** adalah bit yang berfungsi untuk menentukan level prioritas sumber intrupsi timer 1.

-> **PX1 (IP.2)** adalah bit yng berfungsi untuk menentukan level prioritas sumber intrupsi eksternal 1.

-> **PT0 (IP.1)** adalah bit yng berfungsi untuk menentukan level prioritas sumber intrupsi Timer 0.

-> **PX0 (IP.0)** adalah bit yng berfungsi untuk menentukan level prioritas sumber intrupsi eksternal 0.

-> Ada dua lokasi bit yang belum dialokasikan (IP.6 dan IP.7), kedua lokasi ini akan dipakai untuk pengembangan produk mikrokontroler selanjutnya.

Register IP berfungsi untuk mengatur peringkat atau prioritas intrupsi dari keenam sumber intrupsi mikrokontroler. Sama dengan mengaktifkan dan membuat non aktif sumber intrupsi pada IE, untuk menentukan peringkat yang lebih tinggi adalah dengan memberikan nilai 1 pada peringkat yang lebih tinggi, dengan nilai 0 untuk yang lebih rendah. Apabila muncul dua buah intrupsi dengan prioritas yang sama, maka akan diadakan polling untuk menentukan yang mana yang akan dieksekusi terlebih dahulu. Urutan polling adalah sebagai berikut: Intrupsi Eksternal 0, Intrupsi Timer 0, Intrupsi Eksternal 1, Intrupsi Timer 1, Intrupsi Port serial, dan terakhir adalah intrupsi Timer 2. Perlu juga diketahui bahwa ada sebuah sumber intrupsi yang lebih tinggi prioritasnya dari keenam sumber intrupsi dengan urutan prioritas seperti diatas, yaitu RESET system mikrokontroler.

- ✓ **Register Power Control:** Pada register power control bit ke 4, 5, dan 6 ini masih tidak terdefinisi. Sedangkan bit 2 dan bit 3 adalah bit flag yang dapat digunakan secara bebas oleh programmer dalam aplikasinya. SMOD berperan dalam komunikasi serial mikrokontroler, yaitu penggunaan mode 1, 2, dan 3 dalam komunikasi serial. Untuk lebih jelasnya kita bisa memperhatikan tabel dibawah:

Tabel 2-7, Peta Bit Register PCON

Bit.7	Bit .6	Bit .5	Bit .4	Bit .3	Bit .2	Bit .1	Bit .0
SMOD	-	-	-	GF1	GF0	PD	IDL

Keterangan: -> ***Idle Mode (IDL),*** intruksi yang menyebabkan bit IDL bernilai 1 adalah intruksi terakhir yang diproses sebelum mikrokontroller memasuki kondisi idle. Pada kondisi idle, clock menuju CPU mikrokontroler akan diputus, namun tidak pada peripheral lainya seperti serial port, intrupsi, timer, dan lain sebagainya. Mode idle dapat diakhiri dengan picuan intrupsi atau dengan melakukan reset terhadap system mikrokontroler. ***Power-down Mode (PD),*** apabila bit PD pada PCON diset sehingga bernilai 1, maka mikrokontroler akan memasuki kondisi power down. Pada mode powerdown yang dilakukan mikrokontroler adalah mematikan on-chip oscillator, mematikan semua fungsi dari mikrokontroler, dan memberikan nilai low terhadap ALE dan PSEN. Satu-satunya jalan keluar dari kondisi power down adalah dengan melakukan reset terhadap sistem.

✓ **Register WMCON:** *Register WMCON* adalah *Special Function Register (SFR)* yang terletak pada alamat 96H. Register ini berperan dalam penggunaan Timer *watchdog* dan memori (EEPROM) tambahan pada AT89S51. Pada register WMCON bit PS2, PS1, dan PS0 (bit 5, 6, dan 7) berguna untuk menentukan nilai periode dari watchdog timer, memberikan nilai 0 pada ketiga bit tersebut akan memberikan nilai periode 16 ms (milliseconds). Sedangkan menuliskan nilai 1 pada ketiga bit tersebut akan

memberikan nilai periode 2048 ms. Pada register WMCON bit 4 dan 3 (EEMWE dan EEMEN) berperan dalam penggunaan memori EEPROM tambahan. Pada register WMCON bit 2 (DPS) atau sering disebut *Data Point Select*, berguna untuk melakukan pemilihan satu diantara dua register DPTR yang akan digunakan. Apabila bit DPS bernilai 0, maka yang dipilih adalah DP0, sedangkan apabila bernilai 1 maka yang dipilih adalah DP1. Pada register WMCON bit 1 (WDTRST) atau sering disebut *Watchdog Timer Reset and EEPROM Ready busy Flag)* memiliki dua fungsi, yaitu sebagai reset untuk watchdog timer (dengan menuliskan nilai 1), dan sebagai tanda apabila EEPROM siap diisi program oleh program. Nilai 1 pada bit ini menyatakan bahwa EEPROM siap untuk ditulisi. Pada register WMCON bit 0 (WDTEN) atau sering disebut *Watchdog Timer Enable* berfungsi sebagai saklar on/off untuk mengaktifkan watchdog timer. Berikut adalah detail semua bit WMCON :

Tabel 2-8, Peta Bit Register WMCON

Bit.7	Bit .6	Bit .5	Bit .4	Bit .3	Bit .2	Bit .1	Bit .0
PS2	PS1	PS0	EEMWE	EEMEN	DPS	WDTRST	WDTEN

- ✓ **Register Timer/Counter:** Mikrokontroler AT89S51 memiliki dua buah timer, masing-masing diberi nama Timer 0 dan Timer 1. Clock input dari kedua Timer ini diperoleh dari frequensi X-TAL yang terpasang dibagi 12. Kedua timer bisa dipakai untuk meng-*interrupt* program dengan selang waktu yang bisa dihitung dengan rumus:

- Bila dipakai sebagai timer 8 bit maka rumusnya :

$$T = (256 - TL_x) * 1/(F.osc/12)$$

- Bila dipakai sebagai Timer 16 bit maka rumusnya :

$$T = (65536 - TH_x\ TL_x) * 1/(F.osc/12)$$

Dimana : TH_x = isi register TH1 atau TH0

TL_x = isi register TL1 atau TL0

Timer 0 dan timer 1 mempunyai 4 mode oprasi yaitu:

- **Timer 0 dan Timer 1**

  Fungsi timer dan counter dipilih dengan bit kontrol C/T pada SFR TMOD. Kedua timer/counter ini mempunyai 4 mode operasi yang dipilih dengan sepasang bit M1 dan M0.

- **Mode 0**

  Kedua timer pada mode ini berfungsi sebagai counter 8 bit dengan *divided-by-32 prescaler*. Menunjukkan operasi mode 0 pada timer 1, sehingga konfigurasi register timer menjadi 13 bit. Ketika perhitungan berubah dari nilai maksimum (semua bit = 1) menjadi 0 maka flag interupt timer TF1 akan aktif. Input akan dihitung oleh timer bila TR1=1 dan salah satu GATE=0 atau INT1=1. Bila GATE diset = 1 maka timer dikontrol oleh input eksternal INT1, dan dapat digunakan untuk mengukur lebar pulsa. TR1 adalah bit kontrol pada

SFR TCON, sedangkan GATE ada pada TMOD. Men-set TR1 Register 13 bit terdiri dari 8 bit TH1 dan 5 bit TL1, 3 bit TL1 bagian atas dapat diabaikan. Men-set TR1 tidak menghapus isi register. Mode 0 untuk Timer 0 sama seperti Timer 1. Substitusi TR1, TF1 dan INT1, dengan TR0, TF0, dan INT0. Ada 2 bit GATE yang berbeda yaitu TMOD.7/TMOD bit ke 7 untuk Timer 1 dan TMOD.3/TMOD bit ke 3 untuk Timer 0.

- **Mode 1**

  Mode 1 sama dengan mode 0, kecuali register timer berjalan dengan 16 bit. Jadi semua bit pada TH1/TL1 (Timer 1) atau TH0/TL0 (Timer 0) berfungsi.

- **Mode 2**

  Pada mode ini register timer berfungsi sebagai counter 8 bit (TL1) dengan isi ulang otomatis. Overflow dari TL1 tidak hanya men-set TF1, tetapi juga mengisi ulang TL1 dengan isi TH1, yang ditentukan dengan software. Proses isi ulang ini tidak mengakibatkan isi TH1 berubah. Mode 2 untuk Timer / Counter 0 sama seperti Timer/Counter 1.

- **Mode 3**

  Timer 1 pada Mode 3 tidak menghitung sama sekali, sama seperti men-set TR1 = 0. Timer 0 pada mode 3 menjadikan TL0 dan TH0 sebagai 2 counter yang terpisah. Cara kerja Timer 0 pada Mode 3 ini ditunjukkan. TL0 menggunakan bit

kontrol Timer 0 : C/T, GATE, TR0, INT0, dan TF0. TH0 berfungsi sebagai timer yang menghitung siklus mesin dan mengambil alih kontrol TR1 dan TF1 dari Timer 1. Jadi TH0 sekarang mengontrol interupt Timer 1. Mode 3 ini digunakan untuk aplikasi yang membutuhkan sebuah timer atau counter 8 bit tambahan. Dengan timer 0 pada Mode 3, 89S51 seolah-olah mempunyai 3 buah timer/counter. Ketika Timer 0 bekerja pada Mode 3, Timer 1 dapat diaktifkan pada mode yang lain. Sebagai contoh Timer 1 dapat digunakan sebagai baud rate generator atau aplikasi apapun yang tidak memerlukan interupt.

Tabel 2-9, Peta Bit Register TMOD

Bit.7	Bit .6	Bit .5	Bit .4	Bit .3	Bit .2	Bit .1	Bit .0
GATE	C/T	M1	M0	GATE	C/T	M1	M0

Keterangan: -> **Gate**, bila bit ini berlogika satu, maka Timer akan berjalan bila kaki INT1 berlogika '1'/High. **C/T,** bit pemilih timer atau counter yaitu: '1'= counter dan '0'= timer. **M1** adalah mode bit 1 dan **M0** adalah mode bit 0.

Sedangkan register yang digunakan untuk mengontrol timer adalah register TCON:

Tabel 2-10, Peta Bit Register TCON

Bit.7	Bit .6	Bit .5	Bit .4	Bit .3	Bit .2	Bit .1	Bit .0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0

Keterangan:

TF1 : Timer 1 overflow flag (diset oleh hardware)

TR1 : Timer 1 on/off (diset / clear oleh software)

TF0 : Timer 0 overflow flag

TR0 : Timer 0 ON / OFF

IE1 : External interrupt 1 edge flag

IT1 : Interrupt 1 type control bit

IE0 : External interrupt 0 edge flag

IT0 : Interrupt 0 type control bit

2.1.8. Program Memori Lock Bits

Mikrokontroler AT89S51 mempunyai tiga buah lock bits yang tidak dapat deprogram (U) atau bisa deprogram (P), untuk lebih jelasnya dapat dilihat pada tabel berikut:

Tabel 2-11, Mode Proteksi Lock Bits

	Program Lock Bits			
	LB1	LB2	LB3	Tipe Proteksi
1	U	U	U	Tidak Memiliki fitur program lock
2	P	U	U	Instruksi MOVC dijalankan dari eksternal program memory yang di non-aktifkan dari byte fetching code yang berasal dari internal memory, EA dicontohkan terpasang pada reset dan selanjutnya pemrograman flash memori di non aktifkan
3	P	P	U	Sama seperti mode 2, tetapi verify di off-kan
4	P	P	P	Sama seperti mode 3, tetapi eksekusi eksternal di off-kan

Ketika lock bit berlogika 1, logic level pada pin EA disampel-kan terpasang selama proses reset berlangsung. Jika perangkat di- power-up tanpa direset terlebih dahulu maka latch diinisialisasikan pada nilai yang acak dan nilai tersebut ditahan sampai reset diaktifkan kembali. Nilai yang telah terpasang pada pin EA harus disetujui dengan level logic arus pada pin tersebut agar device tersebut berfungsi sebagaimana mestinya.

2.1.9. Flash Pemrograman Pada Mode Paralel

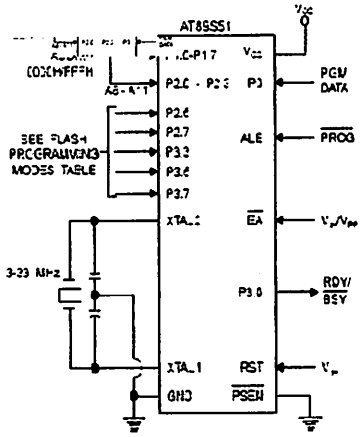
Mikrokontroler AT89S51 memiliki fitur ON-Chip flash memory yang dapat diprogram. Perangkat pemrogram butuh tegangan sebesar 12 volt dc. Barisan kode-kode memori mikrokontroler AT89S51 diprogram secara byte demi byte.

Tiap-tiap code byte pada flash array memori dapat diprogram menggunakan kombinasi kontrol sinyal yang tepat. Sebuah cycle operasi penulisan mempunyai pewaktuan sendiri dan sesekali dilakukan inisialisasi, dan akan memberikan waktu pada dirinya sendiri untuk menyelesaikan proses pemrograman. Berikut ini adalah flash pemrograman pada mode pemrograman parallel:

Tabel 2-12, Flash Pemrograman pada Mode Parallel[1]

Mode	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	P2.3-0 Address	P1.7-0 Address
Write Code Data	5V	H	L	(2)	12V	L	H	H	H	H	D_{IN}	A11-8	A7-0
Read Code Data	5V	H	L	H	H	L	L	L	H	H	D_{OUT}	A11-8	A7-0
Write Lock Bit 1	5V	H	L	(3)	12V	H	H	H	H	H	X	X	X
Write Lock Bit 2	5V	H	L	(3)	12V	H	H	H	L	L	X	X	X
Write Lock Bit 3	5V	H	L	(3)	12V	H	L	H	H	L	X	X	X
Read Lock Bits 1, 2, 3	5V	H	L	H	H	H	H	L	H	L	P0.2, P0.3, P0.4	X	X
Chip Erase	5V	H	L	(1)	12V	H	L	H	L	L	X	X	X
Read Atmel ID	5V	H	L	H	H	L	L	L	L	L	1EH	0000	00H
Read Device ID	5V	H	L	H	H	L	L	L	L	L	51H	0001	00H
Read Device ID	5V	H	L	H	H	L	L	L	L	L	06H	0010	00H

Notes: 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.

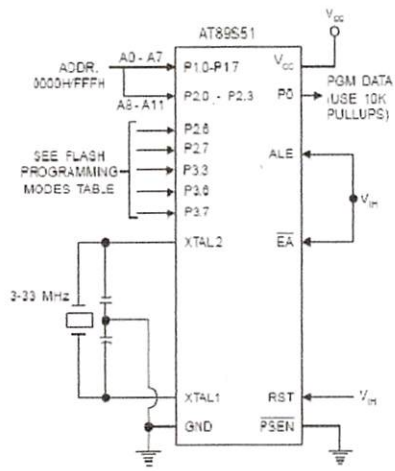


Gambar 2-10, Flash Pemrograman pada Mode Parallel[1]

Berikut adalah karakteristik flash pemrograman dan verifikasi pada mode parallel :

Tabel 2-13, Karakteristik Pemrograman dan verifikasi pada Mode Parallel[1]

Symbol	Parameter	Min	Max	Units
V_{PP}	Programming Supply Voltage	11.5	12.5	V
I_{PP}	Programming Supply Current		10	mA
I_{CC}	V_{CC} Supply Current		30	mA
$1/t_{CLCL}$	Oscillator Frequency	3	33	MHz
t_{AVGL}	Address Setup to $\overline{PROG}$ Low	$48t_{CLCL}$		
t_{GHAX}	Address Hold After $\overline{PROG}$	$48t_{CLCL}$		
t_{DVGL}	Data Setup to $\overline{PROG}$ Low	$48t_{CLCL}$		
t_{GHDX}	Data Hold After $\overline{PROG}$	$48t_{CLCL}$		
t_{EHSH}	P2.7 ($\overline{ENABLE}$) High to V_{PP}	$48t_{CLCL}$		
t_{SHGL}	V_{PP} Setup to $\overline{PROG}$ Low	10		µs
t_{GHSL}	V_{PP} Hold After $\overline{PROG}$	10		µs
t_{GLGH}	$\overline{PROG}$ Width	0.2	1	µs
t_{AVQV}	Address to Data Valid		$48t_{CLCL}$	
t_{ELQV}	$\overline{ENABLE}$ Low to Data Valid		$48t_{CLCL}$	
t_{EHQZ}	Data Float After $\overline{ENABLE}$	0	$48t_{CLCL}$	
t_{GHBL}	$\overline{PROG}$ High to $\overline{BUSY}$ Low		1.0	µs
t_{WC}	Byte Write Cycle Time		50	µs



Gambar 2-11, Verifikasi Flash Memory pada Mode Paralel[1]

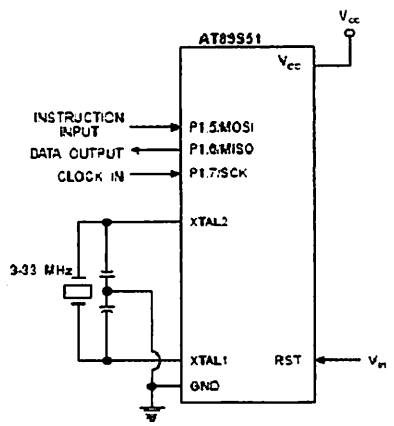
2.1.10. Flash Pemrograman Pada Mode Serial

Pemrogram dapat dilakukan dengan menggunakan ISP serial interface yang mana pin VCC dipengaruhi oleh pin RST. Interface-interface serial tersebut terdiri dari pin-pin, SCK (Serial Clock), MOSI (input), dan

MISO (output). Setelah RST diset pada posisi high, instruksi pemrograman harus dijalankan terlebih dahulu sebelum operasi-operasi lainya dapat dijalankan. Dan jika ingin melakukan pemrograman ulang maka kita harus melakukan oprasi penghapusan *(chip erase)* terlebih dahulu. Berikut adalah flash programming pada mode serial:

Tabel 2-14, Flash Pemrograman pada Mode Serial[1]

Instruction	Instruction Format				Operation
	Byte 1	Byte 2	Byte 3	Byte 4	
Programming Enable	1010 1100	0101 0011	xxxx xxxx	xxxx xxxx 0110 1001 (Output)	Enable Serial Programming while RST is high
Chip Erase	1010 1100	100x xxxx	xxxx xxxx	xxxx xxxx	Chip Erase Flash memory array
Read Program Memory (Byte Mode)	0010 0000	xxxx A11 A10 A9 A8	A7 A6 A5 A4 A3 A2 A1 A0	O7 O6 O5 O4 O3 O2 O1 O0	Read data from Program memory in the byte mode
Write Program Memory (Byte Mode)	0100 0000	xxxx A11 A10 A9 A8	A7 A6 A5 A4 A3 A2 A1 A0	I7 I6 I5 I4 I3 I2 I1 I0	Write data to Program memory in the byte mode
Write Lock Bits[2]	1010 1100	1110 00 BB	xxxx xxxx	xxxx xxxx	Write Lock bits. See Note (2).
Read Lock Bits	0010 0100	xxxx xxxx	xxxx xxxx	xx LB3 LB2 xx	Read back current status of the lock bits (a programmed lock bit reads back as a "1")
Read Signature Bytes[1]	0010 1000	xxx A2 A1 A0	xxx xxxx	Signature Byte	Read Signature Byte
Read Program Memory (Page Mode)	0011 0000	xxxx A11 A10 A9 A8	Byte 0	Byte 1... Byte 255	Read data from Program memory in the Page Mode (256 bytes)
Write Program Memory (Page Mode)	0101 0000	xxxx A11 A10 A9 A8	Byte 0	Byte 1... Byte 255	Write data to Program memory in the Page Mode (256 bytes)



Gambar 2-12, Flash Pemrograman pada Mode Serial [1]

Berikut adalah karakteristik flash pemrograman dan verifikasi pada mode parallel :

Tabel 2-15, Karakteristik Pemrograman pada Mode Serial [1]

Symbol	Parameter	Min	Typ	Max	Units
$1/t_{CLCL}$	Oscillator Frequency	0		33	MHz
t_{CLCL}	Oscillator Period	30			ns
t_{SHSL}	SCK Pulse Width High	$8\ t_{CLCL}$			ns
t_{SLSH}	SCK Pulse Width Low	$8\ t_{CLCL}$			ns
t_{OVSH}	MOSI Setup to SCK High	t_{CLCL}			ns
t_{SHOX}	MOSI Hold after SCK High	$2\ t_{CLCL}$			ns
t_{SLIV}	SCK Low to MISO Valid	10	16	32	ns
t_{ERASE}	Chip Erase Instruction Cycle Time			500	ms
t_{SWC}	Serial Byte Write Cycle Time			$64\ t_{CLCL} + 400$	µs

2.2. Teknologi Bluetooth

Bluetooth adalah sebuah teknologi komunikasi wireless *(tanpa kabel)* yang beroperasi dalam pita frekuensi 2,4 GHz *unlicensed ISM (Industrial, Scientific and Medical)* dengan menggunakan sebuah *frequency hopping tranceiver* yang mampu menyediakan layanan komunikasi data dan suara secara *real-time* antara *host-host Bluetooth* dengan jarak jangkauan layanan yang terbatas. *Bluetooth* sendiri dapat berupa *card* yang bentuk dan fungsinya hampir sama dengan *card* yang digunakan untuk *wireless local area network (WLAN)* dimana menggunakan frekuensi radio standar IEEE 802.11, hanya saja pada *bluetooth* mempunyai jangkauan jarak layanan yang lebih pendek dan kemampuan *transfer* data yang lebih rendah. Pada dasarnya *bluetooth* diciptakan bukan hanya menggantikan atau menghilangkan penggunaan kabel didalam melakukan pertukaran informasi, tetapi juga mampu menawarkan fitur yang baik untuk teknologi *mobile wireless* dengan biaya yang relatif

rendah, konsumsi daya yang rendah, *interoperability* yang menjanjikan, mudah dalam pengoperasian dan mampu menyediakan layanan yang bermacam-macam. *Bluetooth* menggunakan salah satu dari dua jenis frekuensi *Spread Specturm Radio* yang digunakan untuk kebutuhan *wireless*. Jenis frekuensi yang digunakan adalah *Frequency Hopping Spread Spedtrum (FHSS),* sedangkan yang satu lagi yaitu *Direct Sequence Spread Spectrum (DSSS)* digunakan oleh IEEE802.11xxx. *Transceive* yang digunakan oleh *Bluetooth* bekerja pada frekuensi 2,4 GHz *unlicensed ISM (Industrial, Scientific, and Medical).*

2.2.1. Perkembangan Sejarah Perangkat Bluetooth

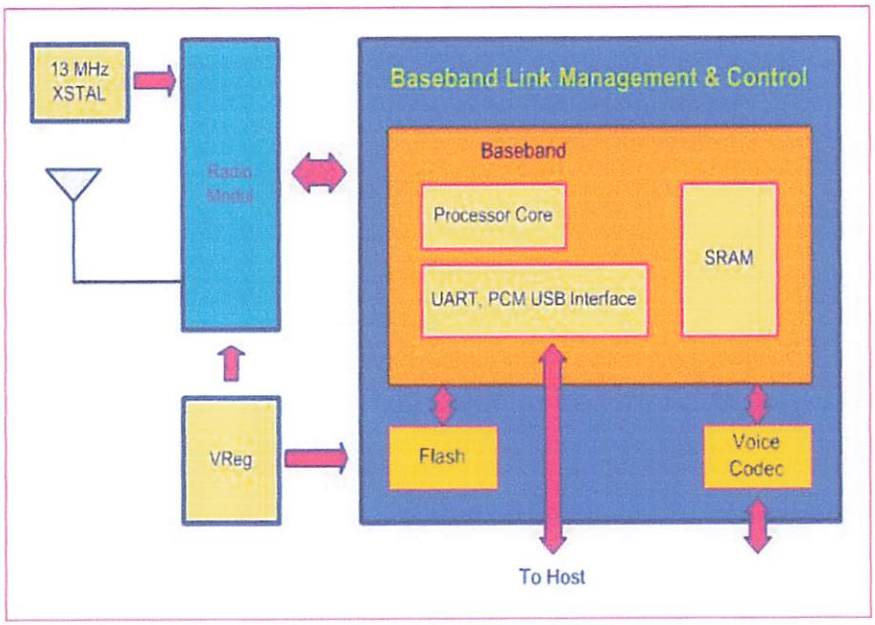
Nama bluetooth berawal dari proyek prestisius yang dipromotori oleh perusahaan-perusahaan raksasa internasional yang bergerak di bidang telekomunikasi dan komputer, *di antaranya Ericsson, IBM, Intel, Nokia, dan Toshiba.* Proyek ini di awal tahun 1998 dengan kode nama bluetooth, karena terinspirasi oleh seorang raja Viking (Denmark) yang bernama Harald Blatand. Raja Harald Blatand ini berkuasa pada abad ke-10 dengan menguasai sebagian besar daerah Denmark dan daerah Skandinavia pada masa itu. *Dikarenakan daerah kekuasaannya yang luas*, raja Harald Blatand ini membiayai para ilmuwan dan insinyur untuk membangun sebuah proyek berteknologi metamorfosis yang bertujuan untuk mengontrol pasukan dari suku-suku di daerah Skandinavia tersebut dari jarak jauh. Maka untuk menghormati ide raja Viking tersebut, yaitu Blatand yang berarti bluetooth (dalam bahasa Inggris) proyek ini diberi nama. Berikut adalah tabel perkembangan teknologi Bluetooth:

Tabel 2-16, Tabel Perkembangan Teknologi Bluetooth

Tahun	Versi	Keterangan
Juli, 1999	1.0 dan 1.0 B	▪ Dibutuhkan perintah manual pada Hardware Device Address (BD-ADDR) transmisi saat proses koneksi di antara dua device dalam satu jaringan (handshaking process). ▪ Keamanan pengguna tidak terjamin ▪ Penggunaan protokol tanpa nama (anonymite mode) tidak dimungkinkan.
Oktober, 1999	1.1 dan 1.2	▪ Digunakannya masks pada perangkat Hardware Device Address (BD-ASSR) untuk melindungi pengguna dari identity snooping (pengintai) maupun tracker. ▪ Penggunaan protokol tanpa nama (anonymite mode) sudah tersedia namun tidak diimplementasikan, sehingga konsumen biasa tidak dapat menggunakannya. ▪ Adaptive Frequency Hopping (AFH), dengan memperbaiki daya tahan dari gangguan frekuensi radio yang digunakan oleh banyak orang di dalam hopping sequence.
	2.0	▪ Diperkenalkannya Non-hopping narrowband channels. Pada channel ini bisa digunakan untuk memperkenalkan layanan profile bluetooth oleh berbagai device dengan volume yang sangat tinggi dari perangkat bluetooth secara simultan. ▪ Tidak dienkripsinya informasi yang bersifat umum secara realtime, sehingga dasar kemacetan trafik informasi dan laju trafik ke tujuan dapat dihindari waktu ditransmisikan oleh perangkat dengan melewati setiap host dengan kecepatan tinggi. ▪ Koneksi berkecepatan tinggi. ▪ Multiple speeds level.

2.2.2. Aplikasi dan Layanan Teknologi Bluetooth

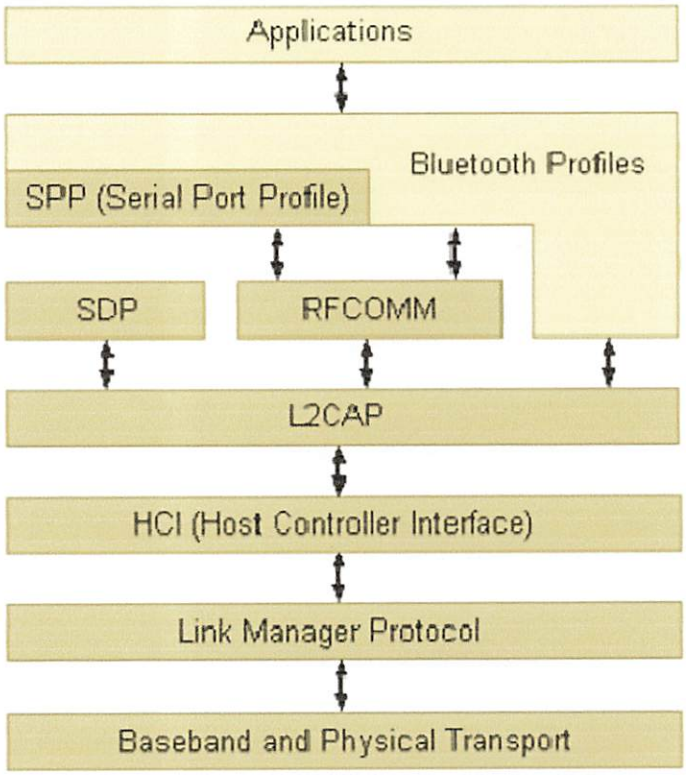
Protokol bluetooth menggunakan *sebuah kombinasi antara circuit switching dan packet switching.* Bluetooth dapat mendukung sebuah kanal data asinkron, tiga kanal suara sinkron simultan atau sebuah kanal dimana secara bersamaan mendukung layanan data asinkron dan suara sinkron. Setiap kanal suara mendukung sebuah kanal suara sinkron 64 kb/s. Kanal asinkron dapat mendukung kecepatan maksimal 723,2 kb/s asimetris, dimana untuk arah sebaliknya dapat mendukung sampai dengan kecepatan 57,6 kb/s. Sedangkan untuk mode simetris dapat mendukung sampai dengan kecepatan 433,9 kb/s. Sebuah perangkat yang memiliki teknologi wireless bluetooth akan mempunyai kemampuan untuk melakukan pertukaran informasi dengan jarak jangkauan sampai dengan 10 meter (~30 feet), bahkan untuk daya kelas 1 bisa sampai pada jarak 100 meter. Sistem bluetooth terdiri dari sebuah radio transceiver, baseband link Management dan Control, Baseband (processor core, SRAM, UART, PCM USB Interface), flash dan voice code. sebuah link manager. Baseband link controller menghubungkan perangkat keras radio ke baseband processing dan layer protokol fisik. Link manager melakukan aktivitas-aktivitas protokol tingkat tinggi seperti melakukan link setup, autentikasi dan konfigurasi. Secara umum blok fungsional pada sistem bluetooth secara umum dapat dilihat pada Gambar dibawah ini.



Gambar 2-13, Blok Fungsional Sistem Bluetooth

2.2.3. Profil Bluetooth

Peralatan Bluetooth dapat mensupport interoperabilitas dengan satu atau lebih peralatan bluetoth. Agar dua perangkat Bluetooth dapat berkomunikasi satu samalain, maka mereka harus melakukan sharing paling sedikit dengan sebuah common profil. Sebagai contoh, jika kita ingin agar PDA (Personal Digital Asistant) kita dapat berkomunikasi dengan sebuah embeddedBlue radio (eb500), maka kita harus dapat memastikan bahwa kedua perangkat Bluetooth tersebut telah mensupport profil yang sama. Perangkat EmbeddedBlue radio (eb500) telah mensupport *Serial Port Profil (SPP)* yang mana merupakan satu dari profil paling atas dan merupakan sebagian profil yang disupport Bluetooth.



Gambar 2-14, Elemen Utama Bluetooth[2]

Sebagai sebuah typical diagram dari sebuah TCP/IP, terdapat bagian-bagian detail yang tersembunyi oleh bagian tumpukan (stack) yang terlihat sederhana, atau lebih sepesifiknya bahwa terdapat bagian dari sebuah profil yang terletak diatas layer L2CAP yang menyediakan power yang banyak dan lebih komplek pada sebuah protocol Bluetooth. Profil tersebut merupakan jalan masuk utama menuju stack untuk sebuah aplikasi. Terutama ketika aplikasi tersebut mendefinisikan set dari sebuah layanan yang diinginkan oleh sebuah aplikasi. Sekarang ini terdapat lebih dari 25 profil berbeda yang ditetapkan oleh Bluetoth SIG *(Special Interest Group)*. Dengan banyaknya jenis profil tersebut, untuk memperoleh

beberapa seluk-beluk pengertian dari Bluetooth adalah bukan merupakan tugas yang sederhana. Akan tetapi, secara singkat pengertian bahwa sebuah single profile dapat disediakan untuk sebuah aplikasi yang menggunakan profil tersebut tanpa harus tau secara detail.

2.3. Modul EmbeddedBlue Radio EB500

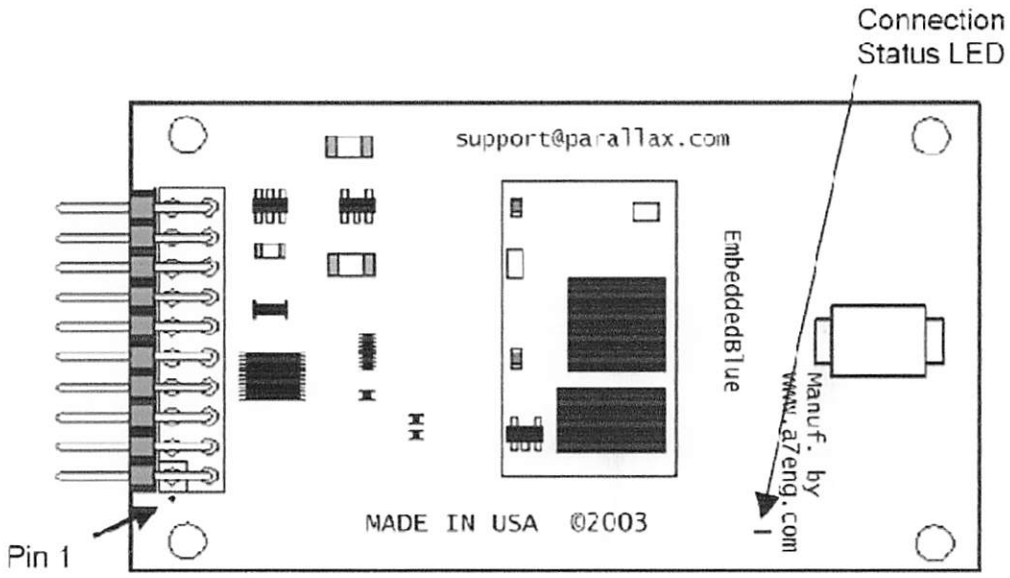
Modul eb500 memiliki dua mode operasi yaitu data mode dan command mode. Ketika power di hidupkan maka eb500 akan masuk pada kondisi command mode dan siap menerima instruksi-instruksi serial.

2.3.1. Command Mode

Pada mode ini terdapat bagian perintah yang dapat dikirim untuk mengubah nilai baudrate, menandai perangkat lain yang berada dalam jangkauan, mengecek versi program, dan sebagainya. Semua perintah dikirimkan menggunakan kode-kode ASCII. Ketika perintah pengiriman sukses dijalankan, maka ACK akan di kembalikan, dan jika terjadi masalah pada syntax dalam sebuah transmisi maka NAK akan di-return, setalah salah satu dari NAK atau ACK di return maka karakter Carriage-return <CR> akan dikembalikan. Ketika prompting di return itu berarti bahwa eb500 berada pada kondisi idle dan menunggu perintah lainya.

2.3.2. Data Mode

Sekali eb500 terkoneksi pada perangkat Bluetooth lainya, maka eb500 akan di switch pada kondisi data mode secara otomatis. Semua data yang dikirim pada mode ini akan dikirim pada remote device dan kemudian tidak akan ada perintah yang dapat dikirim lagi sampai eb500 di matikan atau di switch pada keadaan command mode dengan menggunakan mode control I/O. Status koneksi eb500 dapat dimonitor untuk mengetahui bahwa koneksi sedang aktif (terjadi) antara eb500 dengan perangkat Bluetooth lainya, indikator ini dapat diketahui pada led yang menyala pada modul eb500.



Gambar 2-15, Skema Modul EB500[2]

2.3.3. Spesifikasi EB500

Komunikasi *bluetooth* yang digunakan EB500 memiliki karakteristik sebagai berikut:

- Kekuatan pengiriman sinyal sebesar 4dBm (maksimum).
- Menggunakan tipe komunikasi serial dengan jangkauan komunikasi pada lapangan terbuka dapat lebih dari 100 meter (328 kaki).
- EB500 dapat bekerja dengan baik pada temperature 0° hingga 70°C.
- *Supply Power* sebesar 5 hingga 12 VDC.
- Konsumsi arus sebesar 3mA hingga 35mA (bergantung pada kondisi koneksi dan *Baud Rate*).
- Bluetooth support versi 1.1, compliant dengan L2CAP, RFCOMM, SDP, SPP.

Pin yang dipakai adalah VSS, VIN, RX, TX, dan *Connection Status*. *Pin* RX *Flow* dan TX *Flow* tidak digunakan karena format data serial yang dipakai tidak menggunakan sistem *flow control*. Sedangkan *pin Mode Control* tidak digunakan, karena proses perubahan *mode* pada EB500 dapat dilakukan secara *software*. Komunikasi yang digunakan antara EB500 dan *microcontroller* adalah secara Serial TTL (Standar Pabrik = 9600 *Baud*, 8 *Data Bits*, 1 *Stop Bits*, *No Parity*, dan *No Flow Control*) dimana *baud rate* dan konfigurasi *flow control*-nya

dapat dimodifikasi sesuai keinginan pemrogram. Berikut adalah konfigurasi pin-pin pada modul embedded Bluetooth EB500:

Tabel 2-17, Konfigurasi Pin EB500 Bluetooth[3]

Pin Name	Pin	Type	Description
VSS	1, 2	GND	Ground
RX	3	TTL output	UART data output
TX	4	CMOS/TTL input	UART data input
RX Flow (RTS)	5	CMOS/TTL input, weak pull down	Signaled high to stop module data transmission
TX Flow (CTS)	6	TTL output	Signaled high to stop host data transmission
	7	Reserved	Reserved for future use.
Connection Status	8	TTL output	High when there is an active wireless connection
Mode Control	9	CMOS/TTL input, weak pulled down	Low for command mode / High for data mode
	10 - 19	Reserved	Reserved for future use.
VIN	20	VCC	Module supply, 5 to 12V DC

BAB III

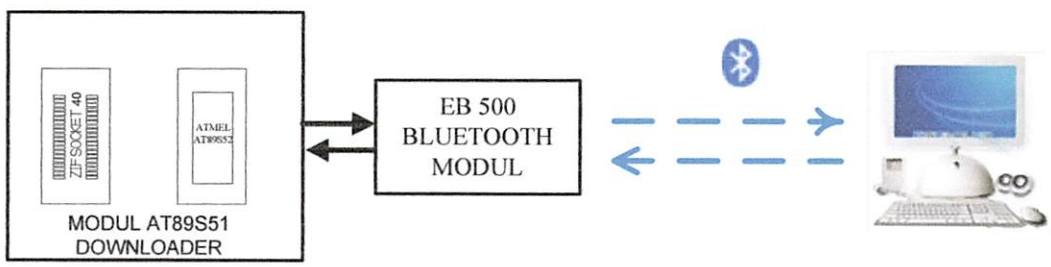
PERENCANAAN DAN PEMBUATAN ALAT

3.1. Pendahuluan

Hardware atau perangkat keras yang digunakan terdiri dari modul downloader AT89S51, modul embeddedblue transceiver (eb500) buatan parallax. Komputer yang digunakan adalah PC Desktop pentiun 4. Untuk perancangan perangkat lunak ditekankan pada desain grafik/ GUI (Graphical User Interface) untuk mempermudah pengaksesan yang dilakukan oleh user. Bahasa program yang dipergunakan adalah Borland Delphi.

3.2. Perancangan Perangkat Keras (Hardware)

Secara umum rancangan yang dibuat terdiri dari tiga bagian yaitu , modul downloader mikrokontroler, modul embbedeblue transceiver EB500 buatan parallax, dan komputer Pentium 4 kompatibel yang telah mendukung teknologi Bluetooth.



Gambar 3-1, Diagram Blok Hardware

Blok diagram dapat dibagi menjadi tiga bagian yaitu:

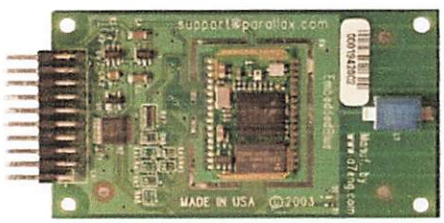
1. Modul AT89S51 Downloader

   Modul ini melakukan pemrosesan data yang dikirimkan oleh komputer secara serial melalui Bluetooth dan memasukkan programnya kepada mikrokontroler AT89S51 target .

2. EB500 Bluetooth Modul

   EB500 Bluetooth Modul adalah suatu modul *bluetooth* buatan Parallax yang dapat menerima maupun mengirim sinyal *bluetooth*. EB500 ini dapat dihubungkan langsung dengan *microcontroler* secara serial sehingga sinyal kontrol yang diterima oleh EB500 dapat langsung diolah oleh *microcontroler*. EB500 bluetooth module menyediakan koneksi secara point to point sama seperti cara koneksi yang terdapat pada kabel serial standar. Berikut adalah spesifikasi dari eb500:

   - Menggunakan tipe komunikasi serial dengan jangkauan komunikasi pada lapangan terbuka dapat lebih dari 100 meter (328 kaki).
   - Kekuatan pengiriman sinyal sebesar 4dBm (maksimum).
   - EB500 dapat bekerja dengan baik pada temperature 0° hingga 70°C.
   - *Supply Power* sebesar 5 hingga 12 VDC.
   - Konsumsi arus sebesar 3mA hingga 35mA (bergantung pada kondisi koneksi dan *Baud Rate*).
   - Bluetooth support versi 1.1, compliant dengan L2CAP, RFCOMM, SDP, SPP.



Gambar 3-2, EB500 Embedded Bluetooth[3]

3. Komputer

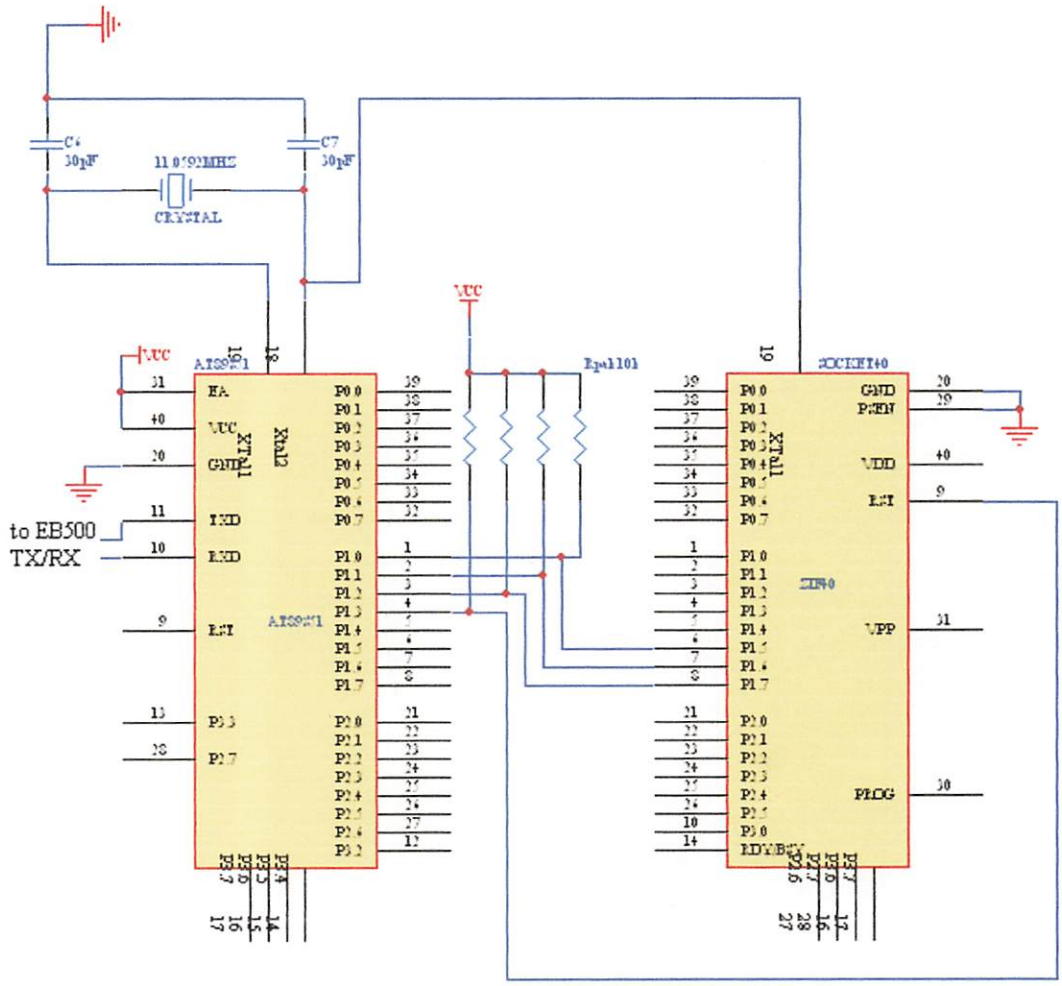
   Digunakan untuk mengetikkan program mikrokontroler yang nantinya akan dikirim ke mikrokontroler target. Pada pembuatan alat ini bisa digunakan komputer pentium 4 atau laptop yang telah mendukung teknologi blutooth.

3.2.1. Modul Downloader AT89S51

Modul downloader AT89S51 ini melakukan pemrosesan data yang dikirimkan oleh komputer secara serial melalui Bluetooth dan memasukkan programnya ke dalam mikrokontroler AT89S51 target. Modul ini menggunakan metode *master and slave*, yang mana menggunakan sebuah ic mikrokontroler AT89S51 sebagai master dan sebuah mikrokontroler target sebagai slave-nya. Ic master bertugas sebagai mikrokontroler pemroses pengiriman data ke mikrokontroler target(slave), sedangkan mikrokontroler target merupakan mikrokontroller yang ingin kita isi program. Modul ini nantinya dihubungkan dengan modul embedded Bluetooth eb500 buatan parallax melalui pin TX dan RXnya. Modul ini mebutuhkan tegangan DC sebesar 12-15 volt.

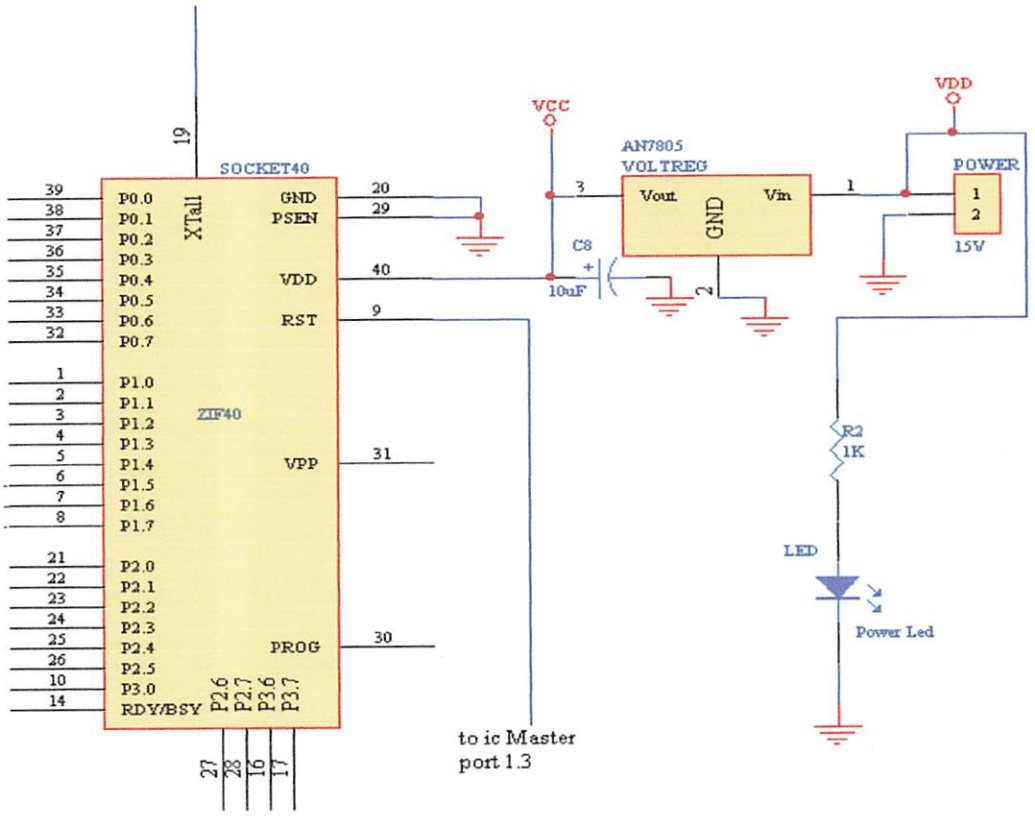
3.2.1.1. Rangkaian IC Master dan Slave

Rangkaian ini terdiri dari sebuah ic master, dalam hal ini saya menggunakan ic AT89S51 buatan atmel, ic ini bertugas sebagai ic inti yang mengolah aliran data saat kita mengirim atau membaca data pada ic slave dan juga melakukan proses verifikasi dan prosedur blank chek, pada rangkaian ini saya menggunakan osilator cristal dengan frekuensi sebesar 11,0592 MHz, dengan tegangan catu pada mikrokontroler master sebesar +5 volt. Pada soket 40 pin yang diggunakan untuk memasang ic slave, pin vcc-nya(pin40) dihubungkan dengan pin reset (pin9), agar proses pemrograman dapat dilakukan, sedangkan tegangan catunya terhubung pada tegangan DC +5 volt. Berikut adalah gambar rangkaiannya:



Gambar 3-3, Rangkaian Master dan Slave

Dapat dilihat pada gambar diatas bahwa kaki no.6 (MOSI pin), no.7 (MISO pin), dan no.8 (Clock pin), pada slave terhubung pada kaki no.1, no.2, dan no.3 pada master, data dikirimkan melalui pin MOSI (Master Output Slave Input) dengan menginputkan sinyal logika '1' pada pin clock. Sedangkan penerimaan data dapat dibaca melalui pin MISO (Master Input Slave Output).

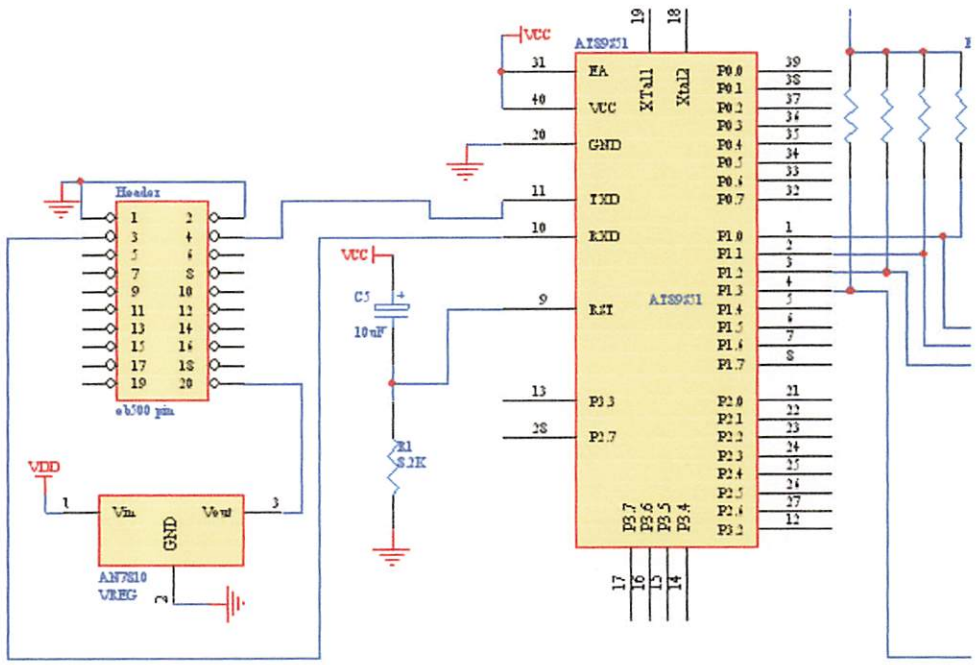


Gambar 3-4, Konfigurasi Pemasangan Tegangan Catu

Untuk mendapatkan tegangan catu +5 volt, saya menggunakan sebuah ic voltage regulator tipe 7805, dengan tegangan output sebesar +5Volt dan tegangan input sebesar 12 volt.

3.2.1.2. Rangkaian Komunikasi Serial EB500

Pada rangkaian ini, terdiri dari rangkaian serial RS232, yang terhubung pada modul Embedded Bluetooth produksi parallax EB500, untuk lebih jelasnya dapat dilihat pada gambar dibawah:



Gambar 3-5, Komunikasi Serial EB500

EB500 , harus mendapatkan tegangan catu sekitar 9 volt agar bisa beroprasi, tegangan ini harus terpasang pada pin 20 sedangkan groundingnya adalah pin 1 dan pin 2 (untuk lebih jelasnya mengenai konfigurasi pin-pin pada EB500, dapat dilihat kembali pada sob pokok bahasan tentang spesifikasi EB500 pada bab 2).

EB500 ini berfungsi sebagai penerima dan pengirim data dari/ke computer, melalui gelombang radio Bluetooth dengan frekuensi sebesar 2,4 GHz, dengan jarak pengiriman maximum 10 meter (jika modul berada dalam ruangan) dan 100 meter jika modul berada di udara terbuka (lapangan).

3.2.2. Modul Embedded Bluetooth EB500 Buatan Parallax

EB500 Bluetooth Modul adalah suatu modul *bluetooth* buatan Parallax yang dapat menerima maupun mengirim sinyal *bluetooth*. EB500 ini dapat

dihubungkan langsung dengan *microcontroller* secara serial sehingga sinyal kontrol yang diterima oleh EB500 dapat langsung diolah oleh *microcontroller.* EB500 bluetooth module menyediakan koneksi secara point to point sama seperti cara koneksi yang terdapat pada kabel serial standar.

EB500 memiliki 2 mode operasi utama yaitu *command mode* dan *data mode.* Setiap kali dilakukan *power up,* EB500 akan selalu masuk dalam *command mode* dan siap untuk menerima *serial command.* Pada *command mode* ini terdapat beberapa perintah yang dapat dikirim untuk menggunakan berbagai macam fitur yang dimiliki oleh EB500. Perintah-perintah itu antara lain adalah : *get con* <CR> (notasi <CR> merupakan 0Ah dan 0Dh) : perintah untuk melihat kondisi *connectable* dari EB500 yang digunakan, *get dis* <CR> : Perintah untuk melihat kondisi *discoverable* dari EB500 yang digunakan, *set dis status* <CR> : Perintah untuk merubah kondisi *discoverable* dari EB500. Variabel status diisi *on* (aktif) atau *off* (non aktif).

3.2.3. Komputer

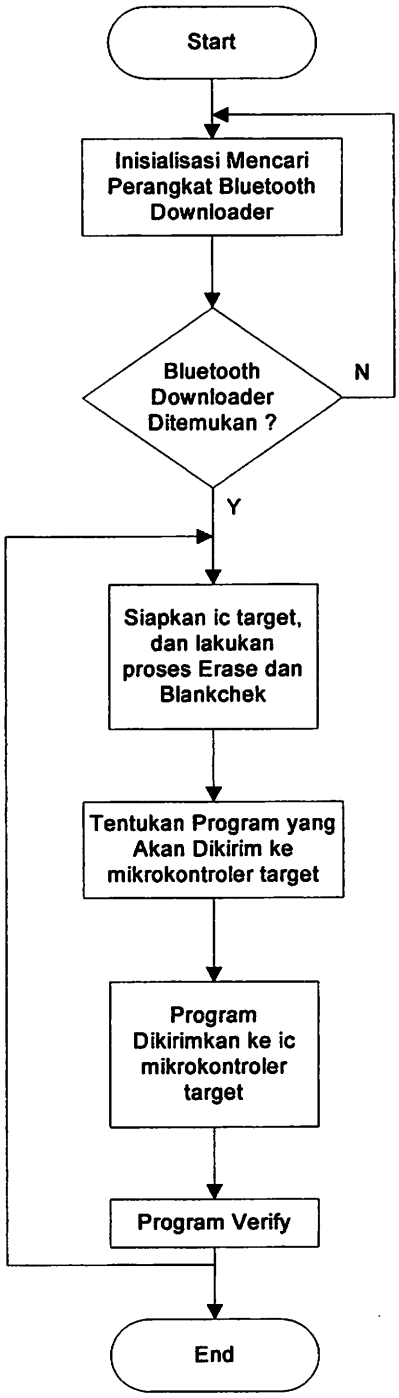
Komputer digunakan untuk mengirimkan dan membuat file-file ber-extension hexa. File-file hexa ini dibuat menggunakan asm51 dengan bahasa assembler. Penulisan file bisa dilakukan dengan program notepad (program bawaan windows) dan kemudian disimpan dengan nama bebas dan dibelakan nama file diketikkan titik(.)asm, dan kemudian di-compile dengan program asm51. Kemudian setelah dikompile maka akan muncul dua file

lagi yaitu file (.lst) dan file (.hex), nah file hex inilah yang nantinya dikirimkan menggunakan program downloader ke mikrokontroler target.

3.2.4. Flowchart dan Cara Kerja Alat

Saat sistem dihidupkan maka sistem harus melakukan pencarian terhadap perangkat bluetooth downloader mikrokontroler, jika ditemukan maka sistem akan berada dalam keadaan ready, maka kita bisa menyiapkan ic target dan file program yang berektension hex, sebelum melakukan pemrograman ke mikrokontroler target maka kita harus melakukan proses erase dan blank chek terlebih dahulu, selanjutnya kita bisa melakukan proses pemrograman, setelah kira-kira 5 detik maka kita harus melakukan proses verifikasi program untuk memastikan bahwa tidak terjadi lost bit/kehilangan bit data pada saat pemrograman, setelah itu sistem akan kembali pada posisi ready sampai sistem dimatikan.

Berikut adalah gambar flowchart dari cara kerja alat:



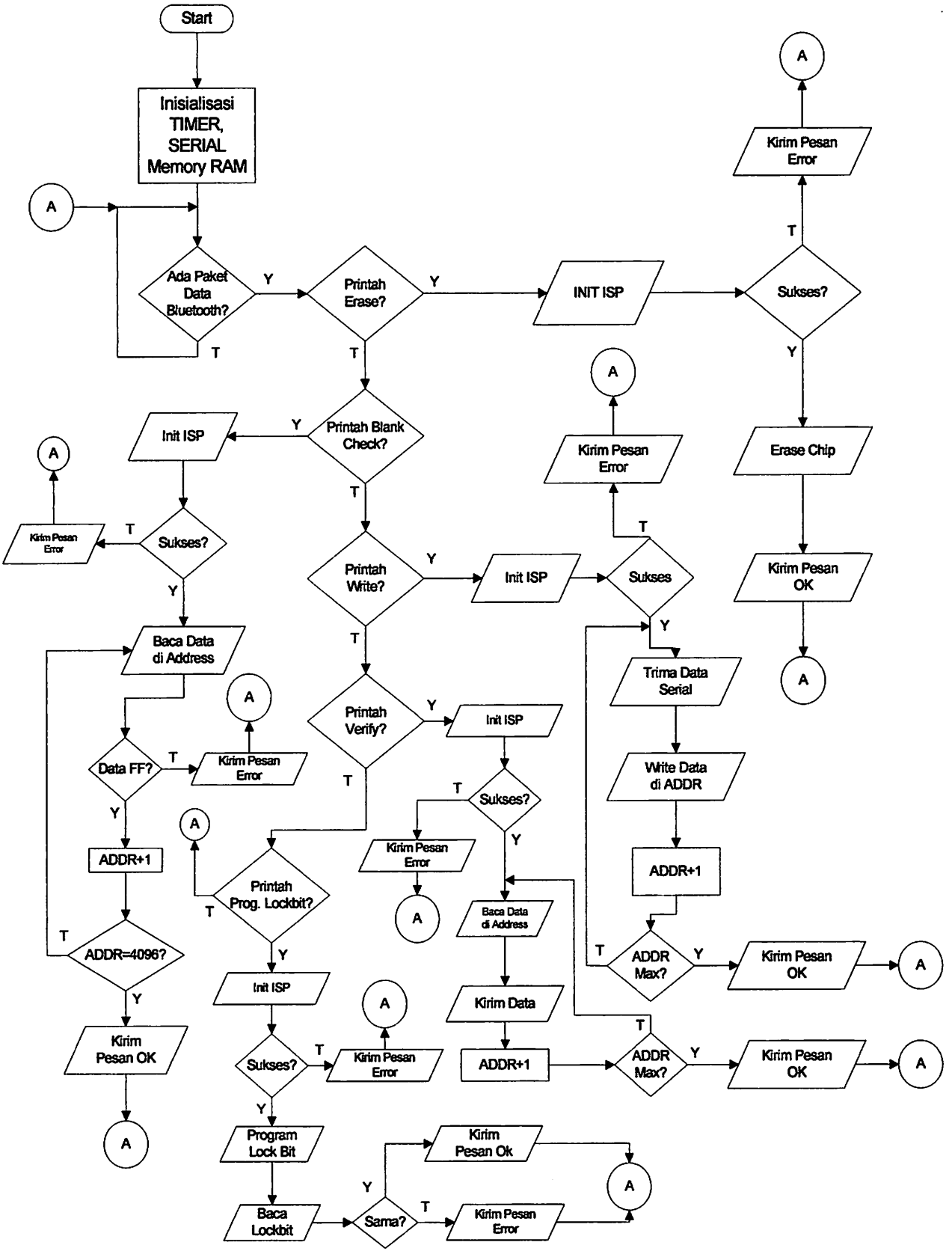
GAMBAR 3-6, Cara Kerja Downloader

3.3. Perancangan Perangkat Lunak (Sofware)

Bahasa program yang digunakan ada 2 buah, untuk mikrokontroler masternya kita menggunakan bahasa assembler, dan untuk program di kumputer, kita menggunakan bahasa program yang telah mendukung GUI (Graphical User Interface) dalam hal ini saya menggunakan bahasa program Delphi.

3.3.1. Bahasa Assembler

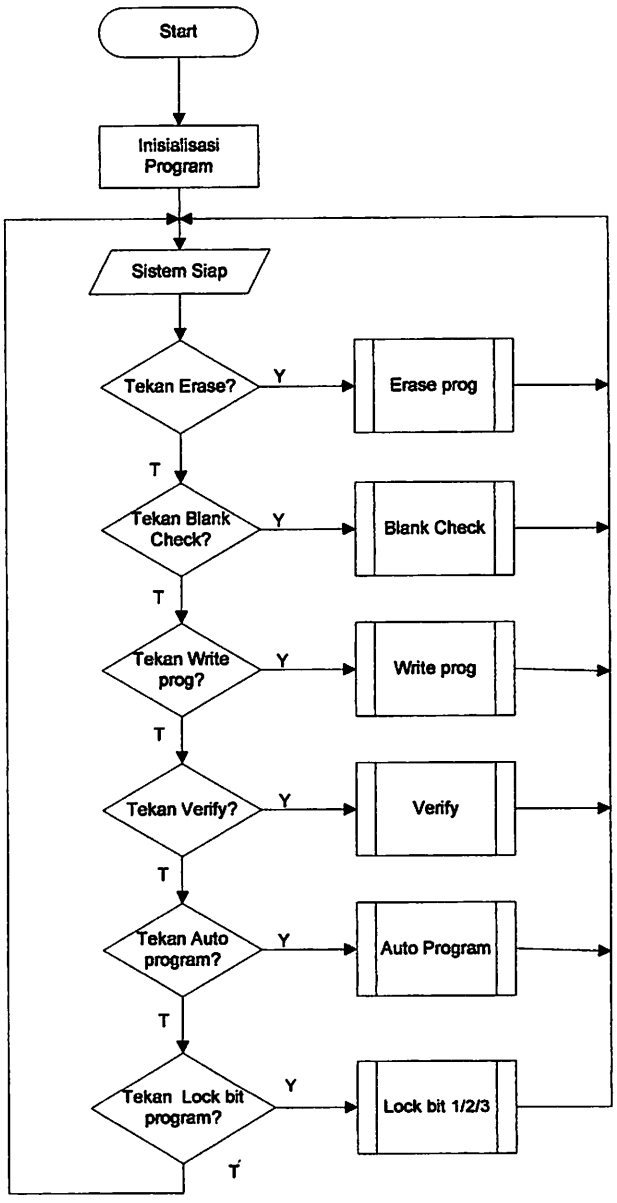
Pada program assembler-nya kita melakukan pengaturan-pengaturan pada komunikasi serialnya dan pengaturan-pengaturan untuk melakukan proses erase, blank chek, programing, verify, dan juga melakukan proses lock. Pada program assembler yang dibuat, timer mode yang digunakan adalah timer mode 1, timer yang diggunakan adalah timer 1, dengan ketentuan baudrate yang digunakan sebesar 9600 dengan resonator Kristal sebesar 11,0592 MHz. Berikut adalah gambar flowchart dari program assembler yang nantinya akan diprogramkan pada ic master.



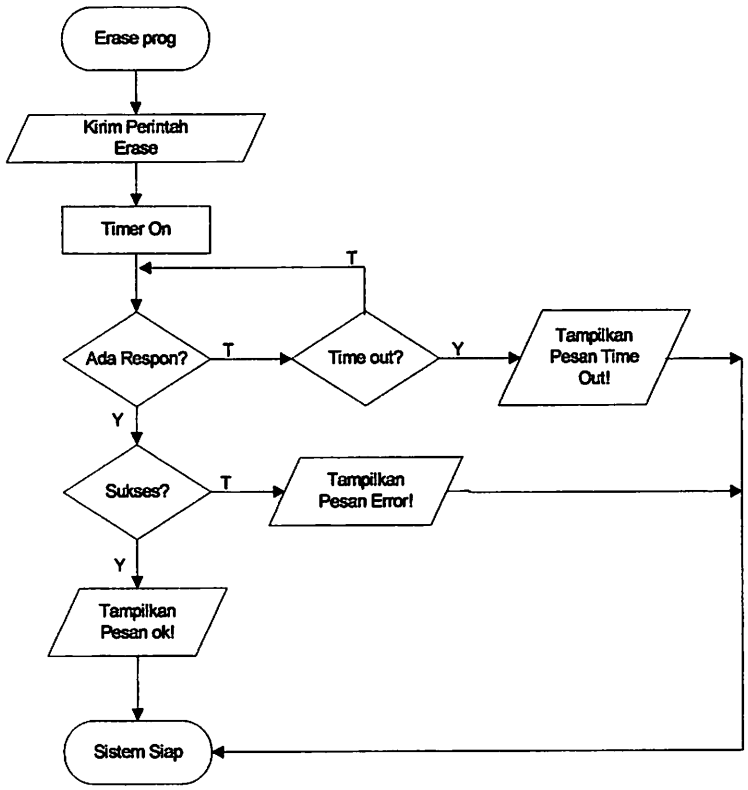
GAMBAR 3-7, Flowchart Cara Kerja Program Assembler ic Master

3.3.2. Bahasa Program Delphi

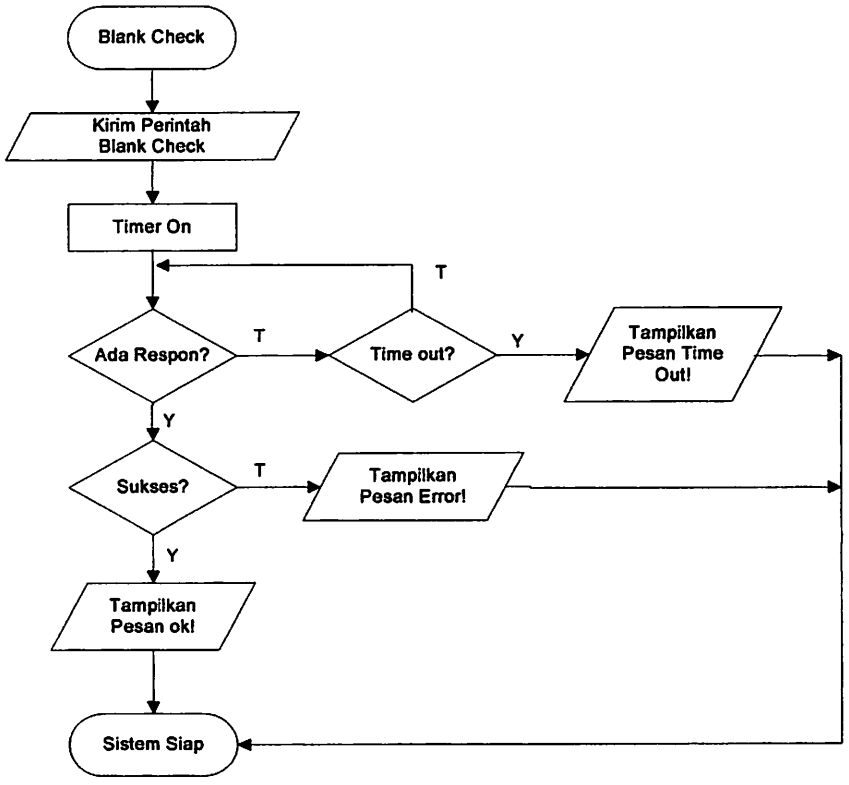
Pada program Delphi dibuat suatu program yang dapat berkomunikasi dengan program assembler yang berada pada mikrokontroler master, sehingga nantinya kita dapat melakukan proses erase, blank check, read memori, write memori, dan program verify.



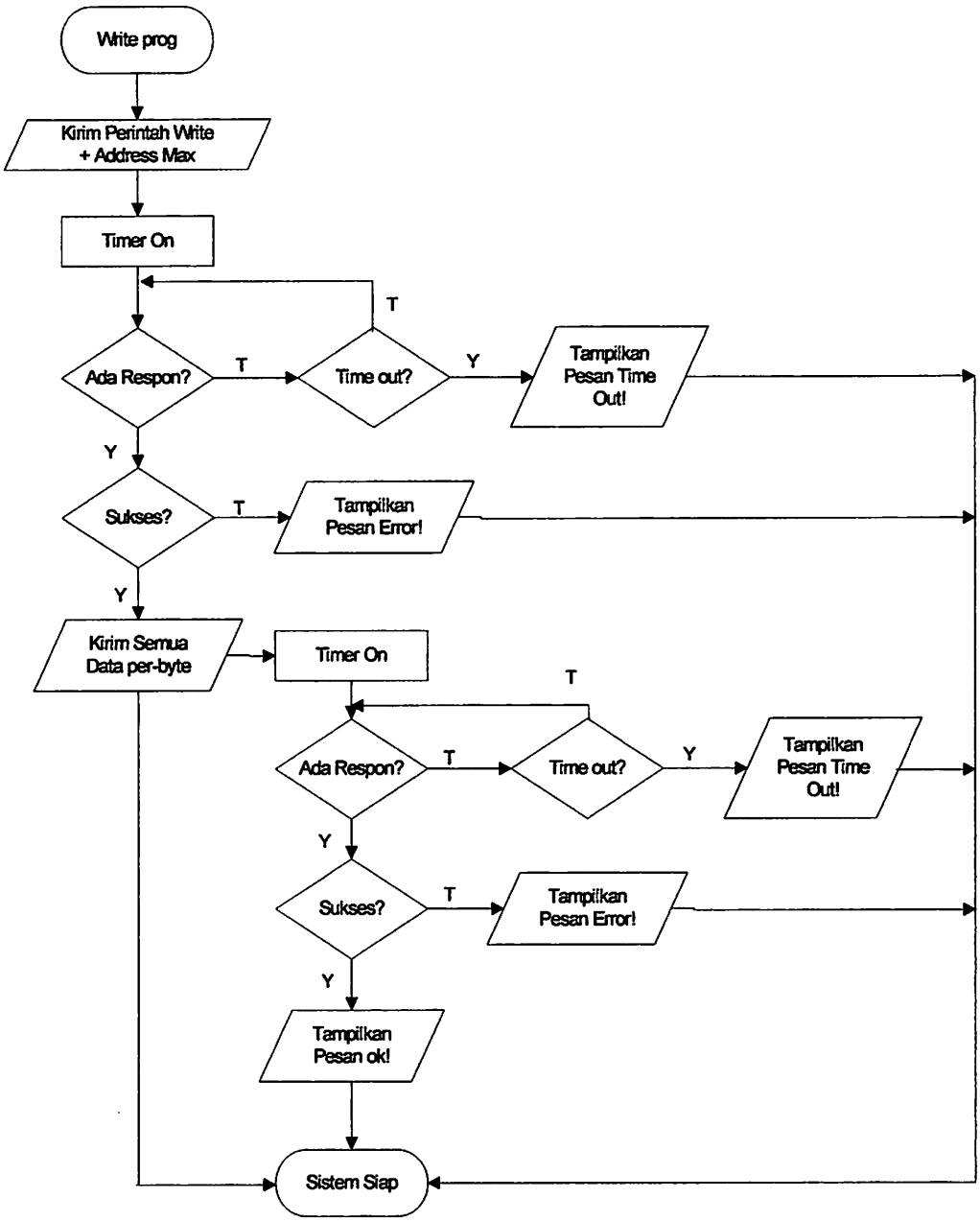
GAMBAR 3-8, Prinsip Kerja Program Downloader Mikrokontroler Secara Keseluruhan



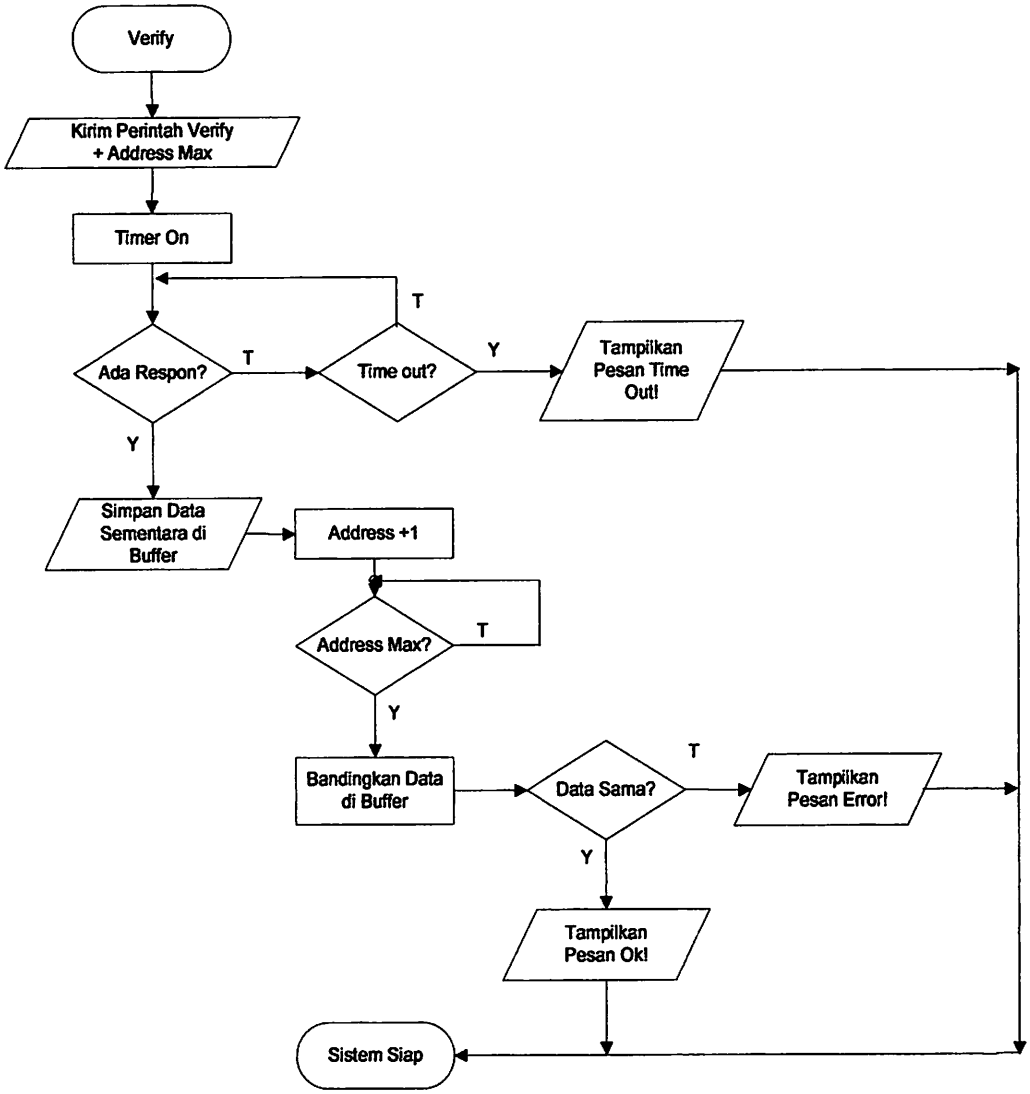
GAMBAR 3-9, Flowchart Prosedur Erase Program



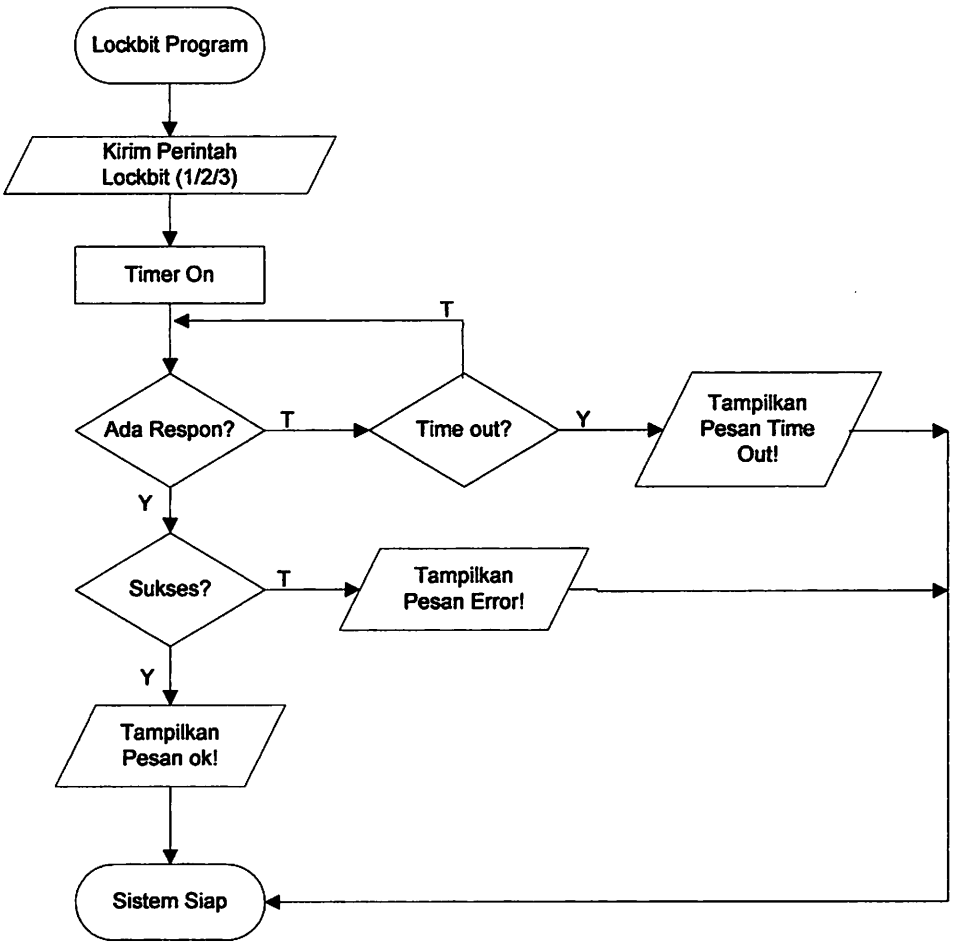
GAMBAR 3-4, Flowchart Prosedur Blank Check



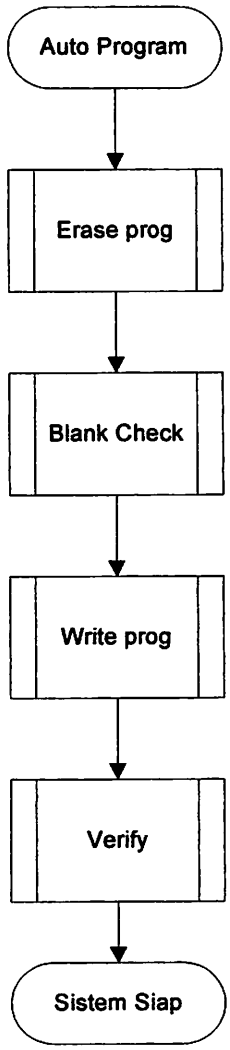
GAMBAR 3-10, Flowchart Prosedur Write Program



GAMBAR 3-11, Flowchart Prosedur Verify



GAMBAR 3-12, Flowchart Prosedur Lockbit



GAMBAR 3-13, Flowchart Prosedur Auto Program

BAB IV

PENGUJIAN ALAT

Setelah alat selesai dikerjakan maka untuk memastikan bahwa alat tersebut bekerja apa tidak maka perlu diadakannya suatu pengujian alat, pengujian ini meliputi pengujian pada perangkat keras atau disebut hardware, dan pengujian pada perangkat lunaknya atau disebut software.

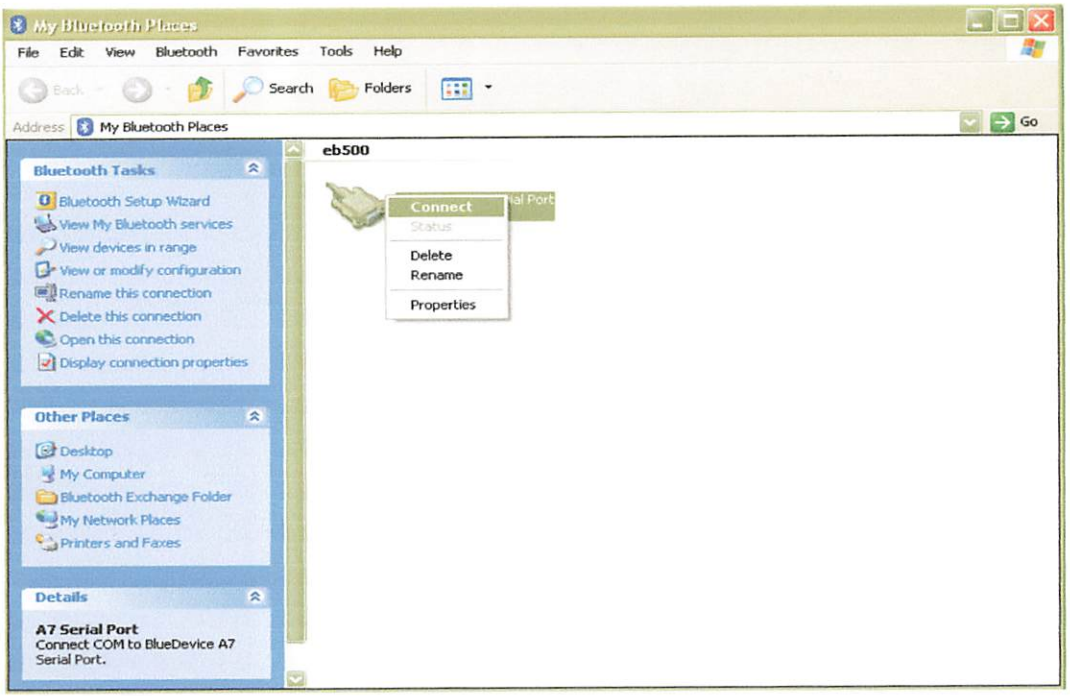
4.1. Pengujian Modul Downloader

Pengujian dilakukan dengan melakukan proses kirim/write pada sebuah mikrokontroler target, untuk komunikasi dengan komputer kita gunakan kabel serial DB9. Pertama kita harus membuat sebuah program untuk menyalakan 8 buah led yg dihubungkan pada port 1 mikrokontroler. Kita akan mengisikan data 0feH pada akumulator lalu dikirimkan ke port 1, selanjutnya isi data akumulator digeser 1 bit ke kiri, dan proses ini akan berlangsung secara terus menerus. Setelah itu kita buat suatu rangkaian sederhana berbasis mikrokontroler yang mana port 1-nya masing-masing terhubung dengan sebuah lampu led. Setelah selesai maka kita akan mencoba menjalankan program yang telah kita isikan dengan perangkat downloader tadi, dan telah didapatkan hasil bahwa lampu led menyala secara bergantian dan bergeser satu demi satu bit ke kiri, hal ini menandakan bahwa perangkat downloader telah berhasil mengirimkan program hex dengan baik, untuk selanjutnya dilakukan proses baca program, erase program dan lock bit, dan hasil uji

menandakan bahwa perangkat downloader telah berfungsi seperti apa yang diinginkan.

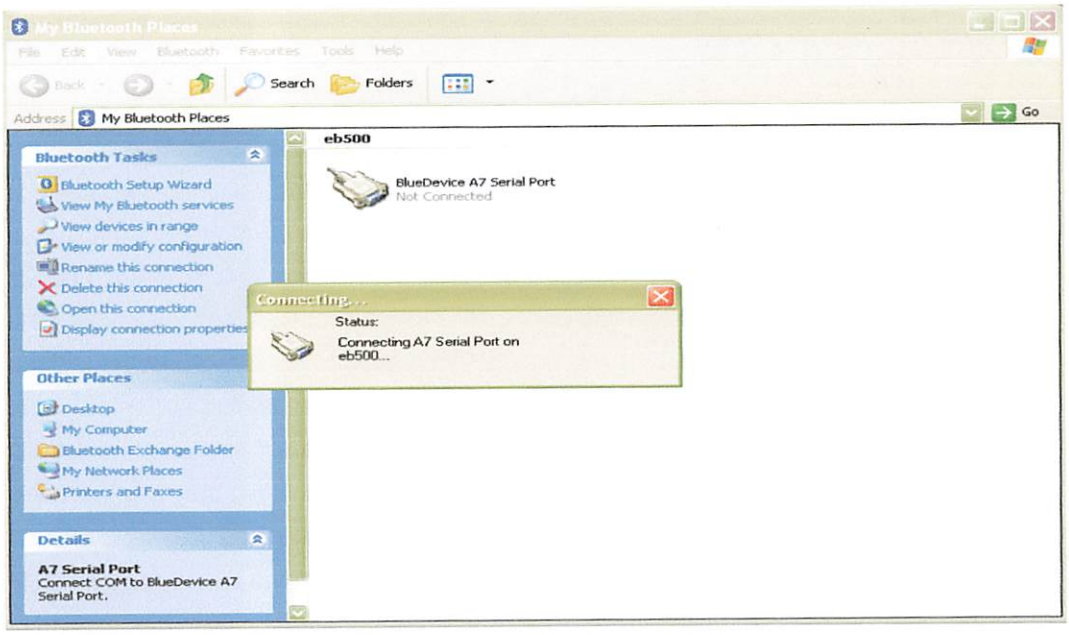
4.2. Pengujian Modul Embedded Bluetooth EB500

Pengujian dilakukan dengan melakukan koneksi melalui gelombang radio Bluetooth, maka untuk melakukan pengujian dibutuhkan sebuah USB Bluetooth dongle, eb500, dan sebuah adaptor dc 9 volt. Langkah pertama ialah menancapkan Bluetooth USB dongle ke port usb di komputer, lalu menghubungkan eb500 dengan adapter 9 volt sedangkan pin TX dan RX di jumper jadi satu, selanjutnya dilakukan proses pencarian eb500 melalui computer, setelah ditemukan maka bisa dilihat comport tambahan pada computer, comport tersebut merupakan comport virtual yang digunakan computer untuk ber-komunikasi dengan eb500. Jika eb500 ditemukan maka akan muncul tulisan connected pada computer, hal ini dapat dilihat pada ikon my Bluetooth places pada layar dekstopnya, sedangkan pada eb500 jika terjadi koneksi maka indicator led pada eb500 tersebut akan menyala. Langkah selanjutnya ialah dengan membuka program hyperterminal untuk mengetest pengiriman pada TX dan RXnya. Gambar berikut adalah gambar ketika kita ingin mengkoneksikan eb500 melalui my Bluetooth places:

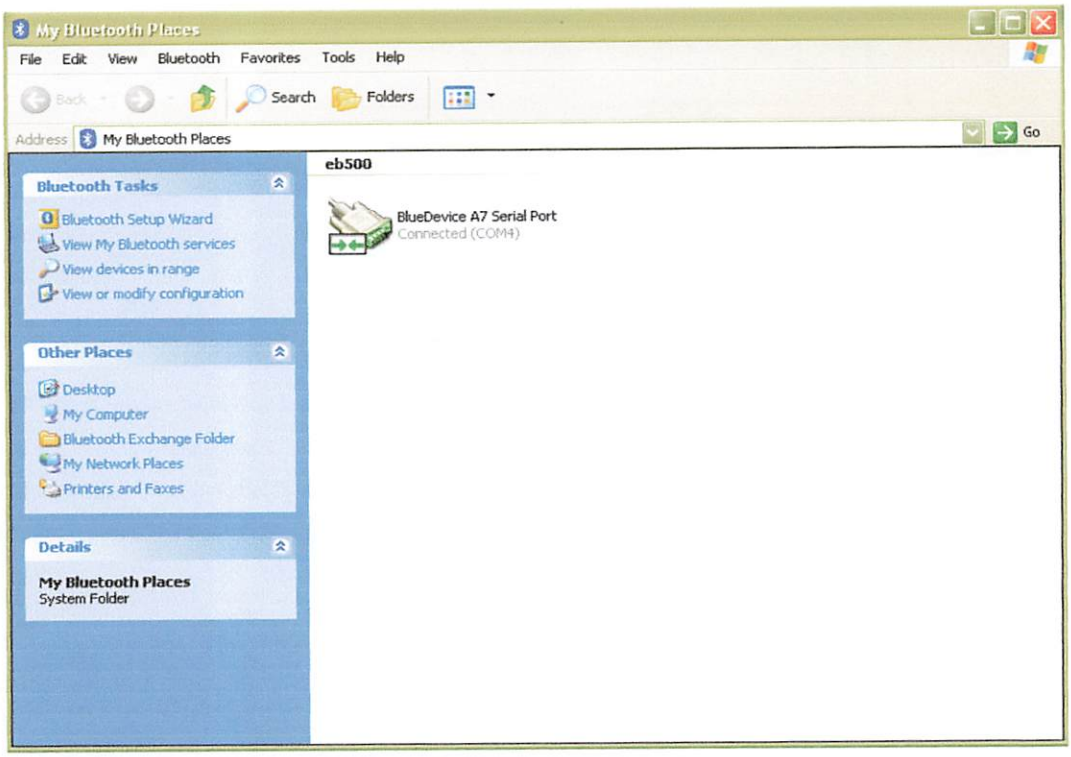


GAMBAR 4-1, Option Connect

Setelah itu akan muncul status connecting, hal ini memberitahukan user bahwa eb500 sedang dikoneksikan ke pc. Setelah eb500 terconect maka akan muncul gambar bahwa eb500 telah terhubung ke pc.

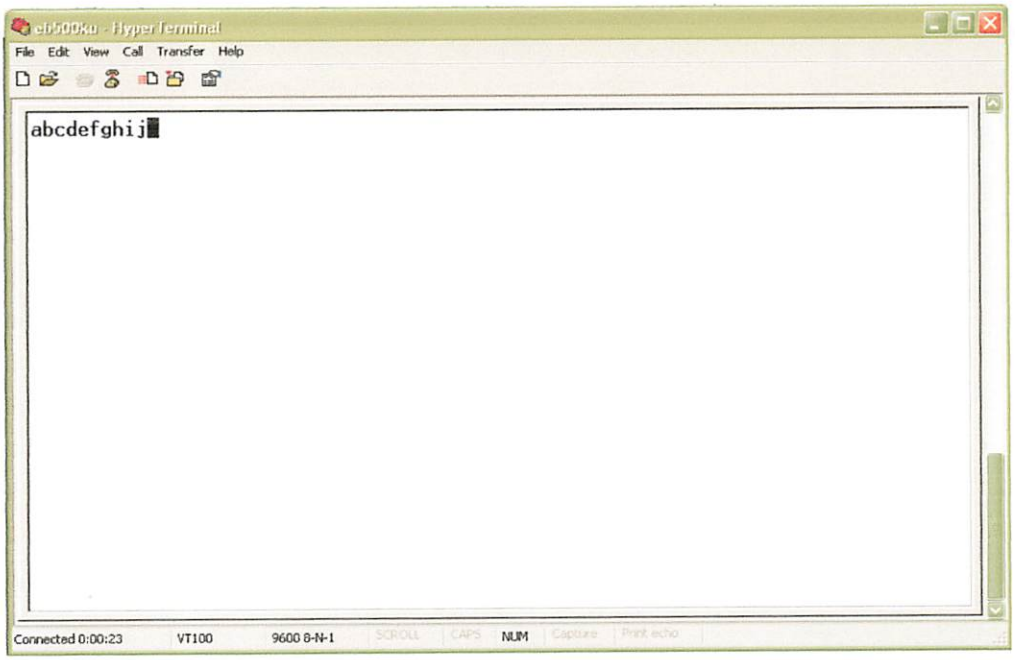


GAMBAR 4-2, Connect Request Status



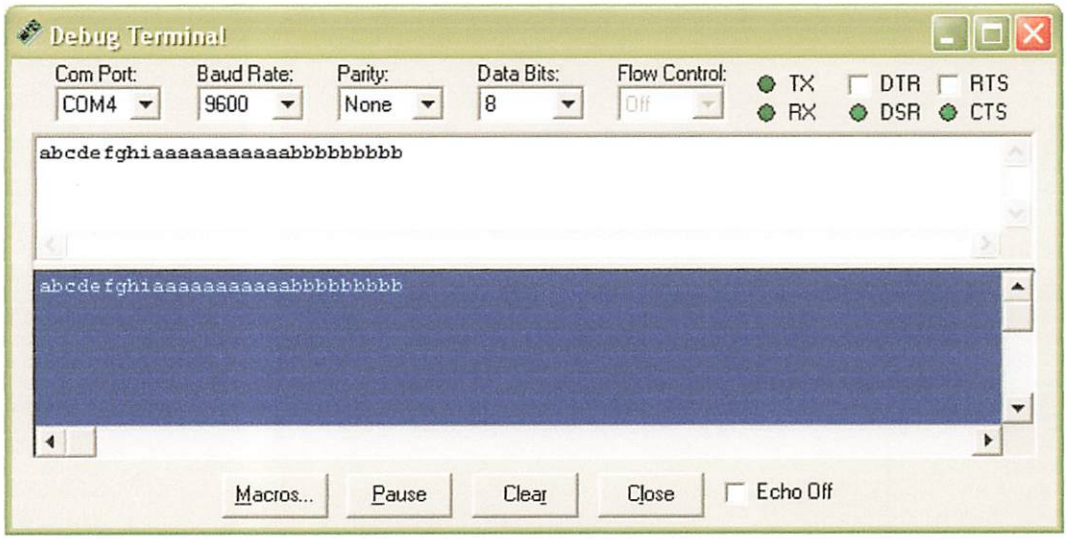
GAMBAR 4-3, EB500 Connected

Setelah itu kita akan melakukan penyetingan pada hyperterminal, dengan menggunakan Com4 (com virtual yang digunakkan computer untuk berkomunikasi dengan eb500. Setelah selesai setting maka kita bisa melihat bahwa saat kita menekan tombol 'a' , maka eb500 juga akan membalas menampilkan 'a' di hyperterminal, begitu seterusnya saat kita menekan tombol 'b' maka eb500 juga akan membalas menampilkan 'b'.



GAMBAR 4-4, Tampilan Hyperterminal saat menerima balasan dari EB500

Kita juga bisa melakukan tes pengiriman dan penerimaan via program debug terminal, yang terdapat pada program basic stamp buatan parallax. Pada gambar dibawah bisa dilihat bahwa pada daerah yang putih adalah huruf yang kita ketikan, dan pada daerah yang biru adalah balasan dari eb500.



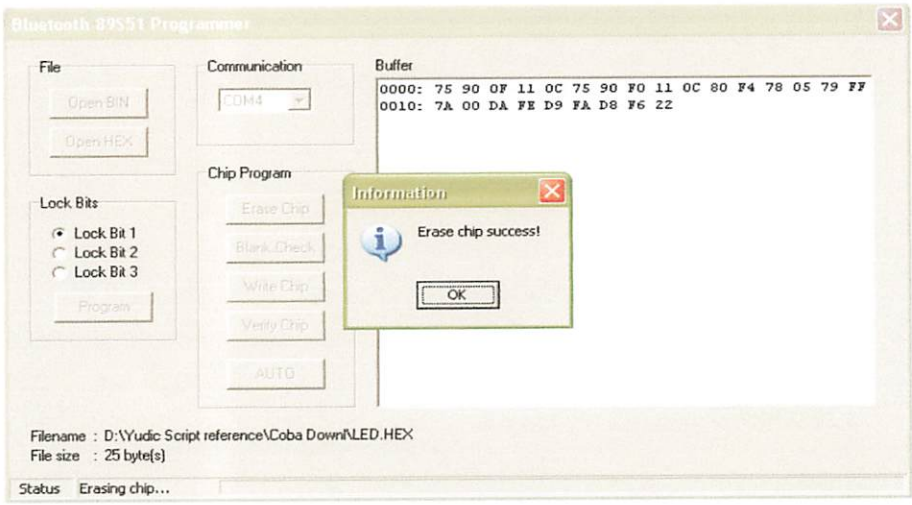
GAMBAR 4-5, Tampilan Program Basic Stamp parallax saat melakukan pengiriman dan penerimaan

Setelah benar-benar yakin bahwa eb500 telah berfungsi dengan baik. Langkah selanjutnya ialah dengan menghubungkan modul eb500 dengan modul downloader mikrokontroler dan menjalankan program downloader mikrokontroler pada computer, setelah itu dilakukan proses hapus, tulis, verify, dan lockbit pada mikrokontroler target.

4.3. Pengujian Software Downloader

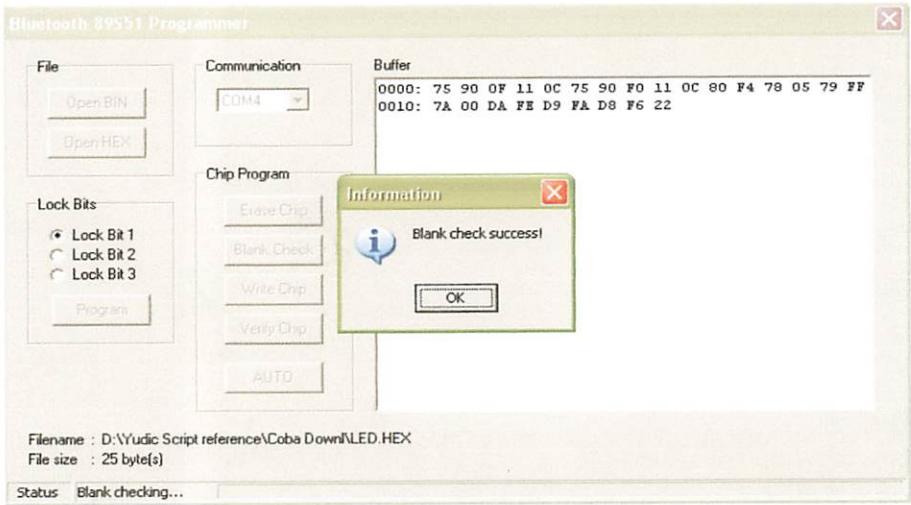
Pengujian Software downloader dapat dilakukan dengan melakukan proses erase, write, verify, blank check, dan lockbit. Langkah selanjutnya adalah dengan melakukan pengujian program sederhana untuk meng-hidup/matikan led. Setelah led bertingkah seperti yang kita inginkan maka software downloader telah

berfungsi sebagaimana mestinya. Gambar berikut adalah gambar ketika program melakukan perintah chip erase:



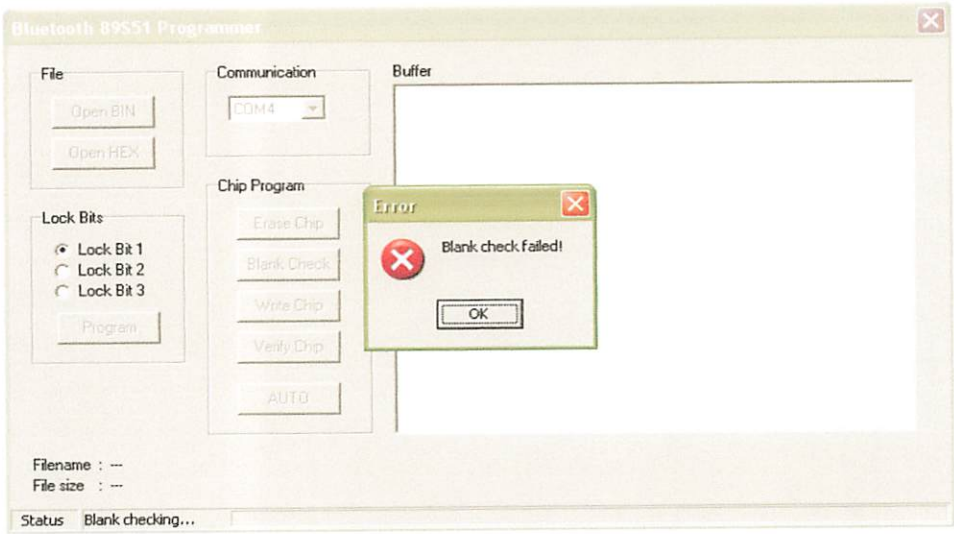
GAMBAR 4-6, Tampilan saat program melakukan chip erase

Saat kita ingin mengetahui apakah chip tersebut telah berisi program apa tidak kita bisa melakukan perintah blank check, jika chip kosong maka akan muncul informasi blank chek success:



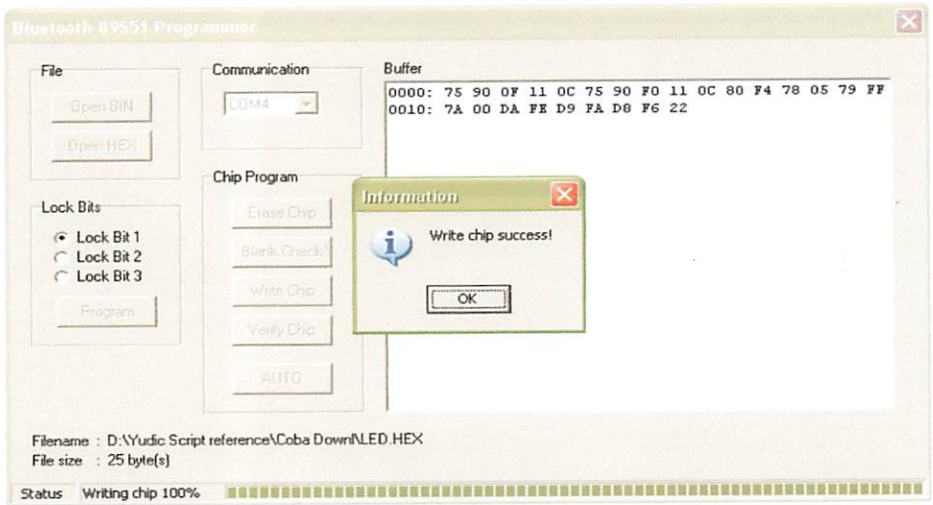
GAMBAR 4-7, Tampilan informasi bahwa chip tidak berisi program

Akan tetapi jika chip yang ingin kita program telah berisi program maka akan muncul error dengan informasi *blank check failed*!



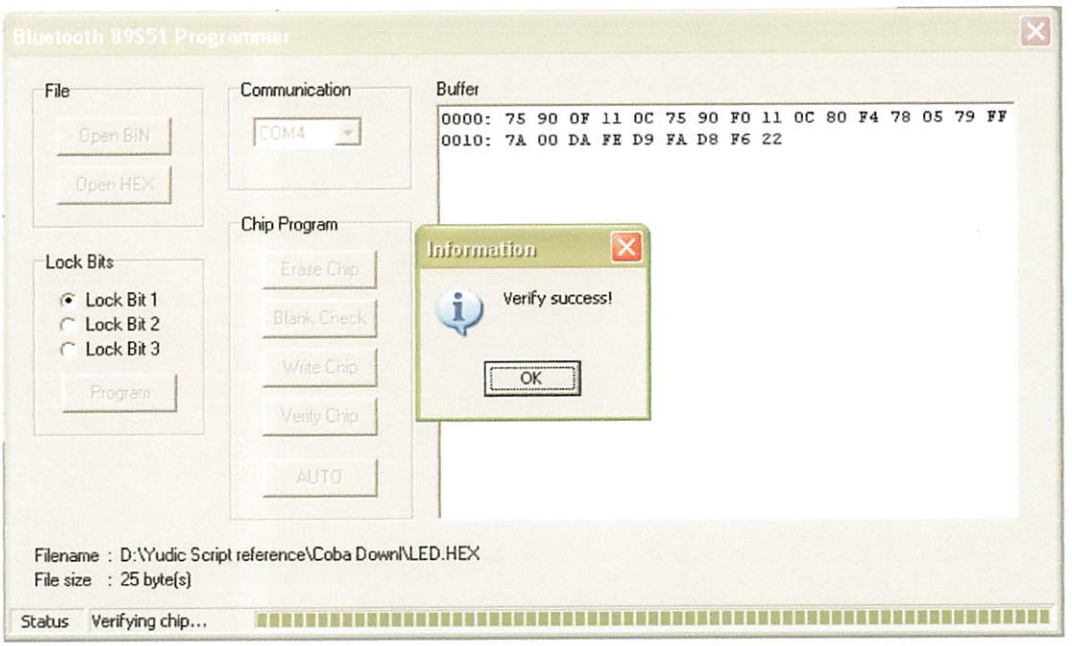
GAMBAR 4-8, Tampilan informasi bahwa chip telah berisi program.

Tampilan Berikut adalah tampilan ketika perintah write sukses dilakukan:



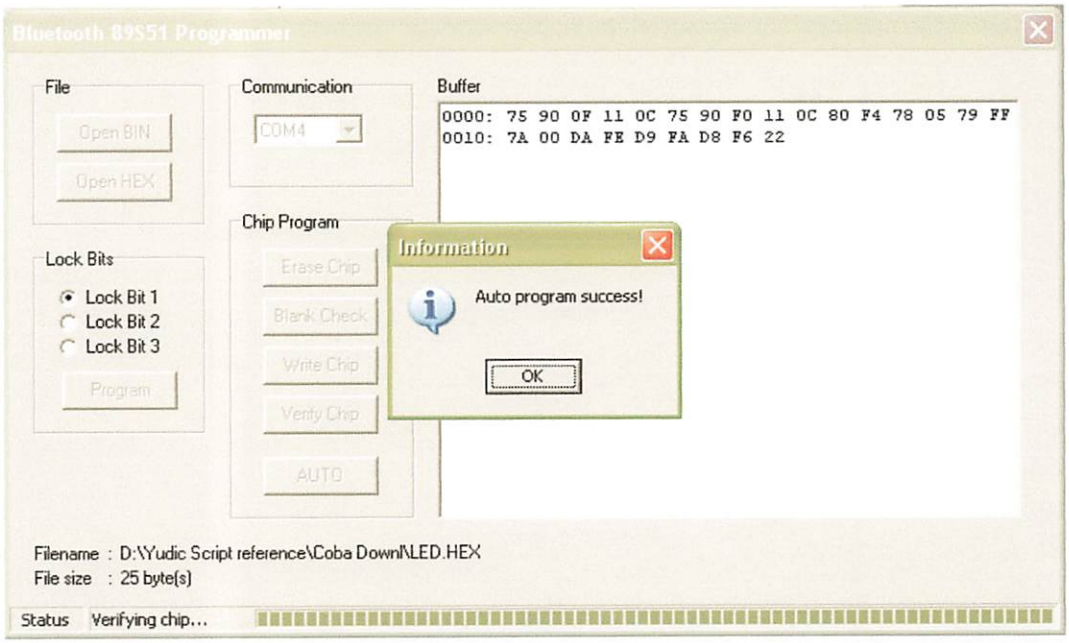
GAMBAR 4-9, Tampilan informasi bahwa operasi write to chip sukses dilakukan

Tampilan Berikut adalah tampilan ketika perintah verify sukses dilakukan:



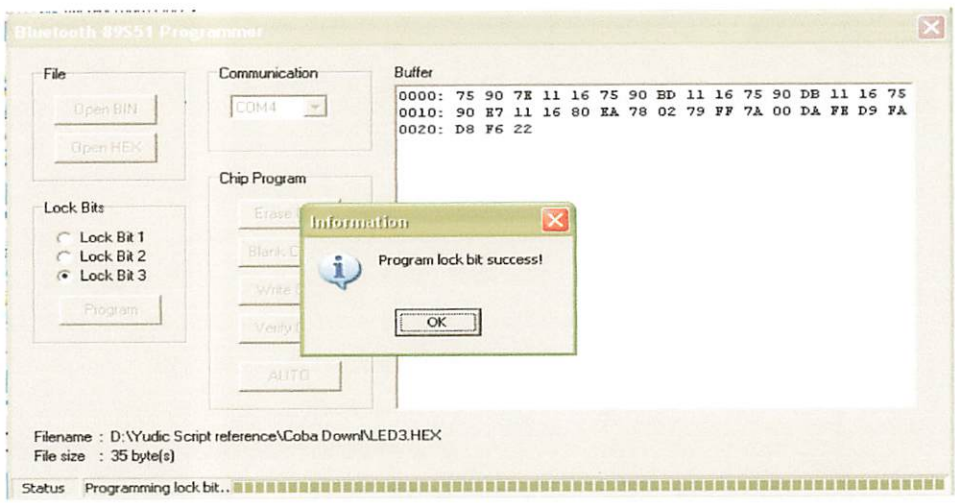
GAMBAR 4-10, Tampilan informasi bahwa operasi verify to chip sukses dilakukan.

Namun dengan alasan keterbatasan waktu, kita bisa juga melakukan pemrograman otomatis, pemrograman otomatis ini secara otomatis melakukan printah erase, blank check, write chip, dan sekaligus melakukan verify secara berurutan. Berikut adalah tampilan bahwa auto programming sukses dilakukan.



GAMBAR 4-11, Tampilan informasi bahwa operasi auto programming sukses dilakukan.

Jika kita ingin memproteksi program yang kita tanam di mikrokontroler target kita bisa melakukan proses lockbit.



GAMBAR 4-12, Tampilan informasi bahwa operasi lock bit sukses dilakukan.

Dengan melakukan semua fitur perintah yang ada pada program downloader maka dapat diketahui bahwa program downloader yang dibuat telah berfungsi dengan baik.

BAB V

KESIMPULAN DAN SARAN

5.1.Kesimpulan

Kesimpulan yang dapat diambil selama peracangan dan pengujian sistem sebagai berikut:

1. Perangkat downloader telah dapat berfungsi dengan baik , baik pada proses write/read program memory, maupun pada proses lockbit dan verifikasi programnya.
2. Dari segi harga, ternyata downloader-downloader yang ada di pasaran yang masih menggunakan kabel serial masih lebih unggul, akan tetapi dari segi kepraktisan alat ini bisa dikatakan lebih unggul daripada downloader yang masih menggunakan kabel serial.

5.2.Saran-Saran

Sedangkan saran yang dapat diberikan untuk penyempurnaan dan pengembangan dari sistem adalah sebagai berikut:

1. Perlunya dilakukan pengembangan-pengembangan, agar bisa meminimalisir harga dalam pembuatan modul.
2. Perlu dipikirkan untuk menggunakan tenaga batre agar lebih efisien dan praktis.
3. Perlunya dilakukan pengembangan agar, modul juga bisa untuk ic-ic mikrokontroler jenis lainya.

DAFTAR PUSTAKA

[1] Atmel AT89S52 Data sheet, www.atmel.com.

[2] Parallax, 2004, *A7 Engineering, Inc. EB500 User Manual*, (Online), (http://www.parallax.com, didownload 15 Jan. 2008)

[3] **Buletin 04/02 eb500** 2003, *A7 Engineering, Inc.* (http://www.parallax.com/dl/docs/prod/comm/eb500.pdf, didownload 15 Jan. 2008)

[4] Wahana Komputer, *Teknik Antarmuka Mikrokontroler dengan Komputer berbasis Delphi*, Penerbit Salemba Infotek.

[5] Agfianto Eko Putra, *Belajar Mikrokontroler AT89C51/52/55(Teori dan Aplikasi)*, Penerbit Gava Media.

LAMPIRAN-LAMPIRAN

Listing Program ic master

```
; PROGRAM INFO
; ============
;
; Name : Yudi - Bluetooth IC programmer
; Date : 29/08/08
; IC  : 89S51
; XTAL : 11.0592 MHz
; ver  : 1

; Additional info :
; ----------------

;-------------------------
;     CONSTANTS
;-------------------------

TH0Val_C        EQU   04Bh        ; timer 50ms
TL0Val_C        EQU   0FFh

Timeout_C       EQU   40          ; per 50ms

Footer_C        EQU   13          ; end marker data serial

ISPOK_C         EQU   69h         ; respon MISO apabila init ISP sukses
```

```
;-------------------------
;       PORTS
;-------------------------

MOSI_P          BIT     p1.0
MISO_P          BIT     p1.1
Clk_P                   BIT     p1.2
Rst_P                   BIT     p1.3

Status_P                BIT     p2.0
Mode_P                  BIT     p2.1

Button1_P               BIT     p2.2            ; for testing purpose only

;-------------------------
;       MEMORY
;-------------------------

TOCtr_M                 EQU     30h             ; variabel penghitung cacahan timer 50ms
SrCtr_M                 EQU     31h             ; variabel penghitung jumlah data serial

JData0_M                EQU     32h             ; jumlah total data yang akan diwrite/read
JDAta1_M                EQU     33h

CtrData0_M              EQU     34h             ; counter jumlah data
```

```
CtrData1_M          EQU   35h

SrBuff_M            EQU   36h           ; buffer data serial dari EB500

;-------------------------
;    Bit Memory
;-------------------------

Button1_F           BIT   2Fh.0
SrValid_F           BIT   2Fh.1         ; flag bila data serial sudah mencapai footer
WriteCmd_F          BIT   2Fh.2         ; flag penanda dalam kondisi command write chip
StartWrite_F   BIT  2Fh.3          ; flag penanda sedang melakukan writing chip

;-------------------------
;   PROGRAM RESET
;-------------------------

                    ORG   0000h
                    LJMP  Start

                    ORG   000Bh
                    AJMP  Timer_0

                    ORG   0023h
                    AJMP  Serial
```

```
;-------------------------
;  Interrupt procedures
;-------------------------


;-------------------------------------------------------------------------
; fungsi : melakukan perhitungan delay setiap 50 ms

Timer_0             ; Timer 50ms

                    MOV   th0,#TH0Val_C
                    MOV   tl0,#TL0Val_C
                    PUSH  a
                    PUSH  psw
                    INC   TOCtr_M
                    MOV   a,TOCtr_M
                    CJNE  a,#Timeout_C,T0J1
                    MOV   SrCtr_M,#0
                    MOV   TOCtr_M,#0
                    CLR   tr0
                    CLR   WriteCmd_F
                    CLR   StartWrite_F

T0J1                POP   psw
                    POP   a
```

```
                RETI

;-------------------------------------------------------------------------

; fungsi : menerima data serial per byte dari EB500

Serial

                CLR     ri
                PUSH    a
                PUSH    psw
                PUSH    b

                MOV     b,sbuf
                MOV     a,sbuf

                JNB     StartWrite_F,SrJ5
                CLR     tr0
                MOV     a,#40h
                ACALL   Send_Byte_SPI
                MOV     a,CtrData1_M
                ACALL   Send_Byte_SPI
                MOV     a,CtrData0_M
                ACALL   Send_Byte_SPI
                MOV     a,b
                ACALL   Send_Byte_SPI
                ACALL   Delay_Write
```

```
        MOV   a,CtrData0_M
        ADD   a,#1
        MOV   CtrData0_M,a
        MOV   a,CtrData1_M
        ADDC  a,#0
        MOV   CtrData1_M,a
        MOV   a,CtrData0_M
        CJNE  a,JData0_M,SrJ6
        MOV   a,CtrData1_M
        CJNE  a,JData1_M,SrJ6
        MOV   dptr,#WriteOK_T
        ACALL Send_Text_PC
        CLR   WriteCmd_F
        CLR   StartWrite_F
        CLR   tr0
        AJMP  SrJX

SrJ6    MOV   TOCtr_M,#0
        MOV   th0,#TH0Val_C
        MOV   tl0,#TL0Val_C
        SETB  tr0
        AJMP  SrJX

SrJ5    JNB   WriteCmd_F,SrJ3
```

```
                MOV     a,SrCtr_M
                CJNE    a,#1,SrJ4
                INC     SrCtr_M
                MOV     JData1_M,b
                AJMP    SrJX
SrJ4            CJNE    a,#2,SrJX
                MOV     JData0_M,b
                SETB    SrValid_F
                CLR     es
                CLR     tr0
                AJMP    SrJX

SrJ3            JB      tr0,SrJ2
                CJNE    a,#'W',SrJ2a
                SETB    WriteCmd_F
                MOV     SrCtr_M,#0
                SJMP    SrJ2

SrJ2a           CLR     WriteCmd_F
SrJ2
                SETB    tr0
                MOV     TH0,#TH0Val_C
                MOV     TL0,#TL0Val_C
                MOV     TOCtr_M,#0
```

```
                CJNE    a,#Footer_C,SrJ1
                CLR     tr0
                SETB    SrValid_F
                CLR     es
                SJMP    SrJX
SrJ1
                MOV     r0,#SrBuff_M
                MOV     a,SrCtr_M
                ADD     a,r0
                MOV     r0,a
                MOV     @r0,b
                INC     SrCtr_M

SrJX            POP     b
                POP     psw
                POP     a
                RETI

;-------------------------
;  COMMON PROCEDURES
;-------------------------

;-------------------------------------------------------------------------

; fungsi : melakukan delay 100ms
```

```
Delay_100ms
                PUSH    07h             ; Push R7
                PUSH    06h             ; Push R6
                PUSH    05h             ; Push R5

                MOV     R7,#0Bh
L0000:
                MOV     R6,#6Ah
L0001:
                MOV     R5,#26h
L0002:
                DJNZ    R5,L0002
                DJNZ    R6,L0001
                DJNZ    R7,L0000

                POP     05h             ; Pop R5
                POP     06h             ; Pop R6
                POP     07h             ; Pop R7
                RET

;-------------------------------------------------------------------------
; fungsi : melakukan inisialisasi timer, interrupt, memory byte dan memory bit

Init_Data
                CLR     Rst_P
```

```
ACALL  Delay_100ms
CLR    Clk_P
ACALL  Delay_100ms

MOV    SCON,#50h            ; 9600bps @11.0592 MHz
MOV    A,PCON
ANL    A,#7Fh
MOV    PCON,A
MOV    A,TMOD
ANL    A,#0Fh
ORL    A,#20h
MOV    TMOD,A
MOV    TH1,#0FDh
MOV    TL1,#0FDh
CLR    ET1
SETB   TR1
SETB   EA
SETB   ES
SETB   PS

MOV    tmod,#21h
MOV    th0,#TH0Val_C
MOV    tl0,#TL0Val_C
SETB   et0
```

```
                MOV     SrCtr_M,#0


                CLR     SrValid_F
                CLR     WriteCmd_F
                CLR     StartWrite_F


                RET

;-------------------------------------------------------------------
; fungsi : mengirimkan 1 byte via serial UART
; input  : a

Send_Byte_PC
                CLR     es
                MOV     sbuf,a
                JNB     ti,$
                CLR     ti
                SETB    es
                RET

;-------------------------------------------------------------------
; fungsi : mengirimkan 1 kalimat text via serial UART
; input  : dptr (address awal text)

Send_Text_PC
```

```
                    CLR     a
                    MOVC    a,@a+dptr
                    JNZ     STPCJ1
                    RET
STPCJ1      ACALL   Send_Byte_PC
                    INC     dptr
                    SJMP    Send_Text_PC

;-------------------------------------------------------------------
; fungsi : mengirimkan 1 byte data ke slave via MOSI sekaligus menerima 1
;       byte dari dari slave via MISO
; input  : a (to MOSI)
; output : a (from MISO)

Send_Byte_SPI
                    PUSH    psw
                    PUSH    7
                    PUSH    b
                    CLR     Clk_P
                    MOV     b,#0
                    MOV     r7,#8
SByJ1
                    ; read from MISO

                    MOV     c,MISO_P
```

```
        PUSH    a
        MOV     a,b
        RLC     a
        MOV     b,a
        POP     a

        ; write to MOSI

        RLC     a
        MOV     MOSI_P,c
        SETB    Clk_P
        CLR     Clk_P
        DJNZ    r7,SByJ1

        MOV     a,b

        POP     b
        POP     7
        POP     psw
        RET

;-------------------------------------------------------------------------
; fungsi : melakukan delay untuk erase chip sekitar 600ms

Delay_Erase             ; sekitar 600ms
```

```
                PUSH    07h             ; Push R7
                PUSH    06h             ; Push R6
                PUSH    05h             ; Push R5
                MOV     R7,#1Fh
L0003:      MOV     R6,#4Ah
L0004:      MOV     R5,#77h
L0005:      DJNZ    R5,L0005
                DJNZ    R6,L0004
                DJNZ    R7,L0003
                POP     05h             ; Pop R5
                POP     06h             ; Pop R6
                POP     07h             ; Pop R7
                RET

;-------------------------------------------------------------------------

; fungsi : melakukan delay untuk write chip sekitar 600us

Delay_Write             ; sekitar 600us

                PUSH    07h             ; Push R7
                PUSH    06h             ; Push R6
                MOV     R7,#20h
L0006:      MOV     R6,#07h
L0007:      DJNZ    R6,L0007
```

```
                DJNZ    R7,L0006
                POP     06h             ; Pop R6
                POP     07h             ; Pop R7
                RET

;-------------------------------------------------------------------------
; fungsi : melakukan inisialisasi ISP
; output : a (hasil init ISP dari MISO -> byte ke 4)

Init_ISP
                CLR     Rst_P
                ACALL   Delay_100ms
                SETB    Rst_P
                ACALL   Delay_100ms

                MOV     a,#10101100b
                ACALL   Send_Byte_SPI
                MOV     a,#01010011b
                ACALL   Send_Byte_SPI
                MOV     a,#0
                ACALL   Send_Byte_SPI
                MOV     a,#0
                ACALL   Send_Byte_SPI
                RET
```

```
        acall   send_byte_pc
        mov     a,#0
        acall   send_byte_spi
        acall   send_byte_pc
        mov     a,#0
        acall   send_byte_spi
        acall   send_byte_pc

        acall   read_lock_bit
        acall   send_byte_pc

        ; erasing chip

;       mov     a,#10101100b
;       acall   send_byte_spi
;       acall   send_byte_pc
;       mov     a,#10000000b
;       acall   send_byte_spi
;       acall   send_byte_pc
;       mov     a,#0
;       acall   send_byte_spi
;       acall   send_byte_pc
;       mov     a,#0
;       acall   send_byte_spi
;       acall   send_byte_pc
```

```
;

                    RET

CBJ1                CLR     Button1_F
CBJ1a
                    RET

;-------------------------------------------------------------
; fungsi : melakukan write lock bit
; input  : a (1-3)

Program_Lock_Bit
                    ADD     a,#0E0h
                    PUSH    a
                    MOV     a,#0ACh
                    ACALL   Send_Byte_SPI
                    POP     a
                    ACALL   Send_Byte_SPI
                    MOV     a,#0
                    ACALL   Send_Byte_SPI
                    MOV     a,#0
                    ACALL   Send_Byte_SPI
                    RET
```

```
;-------------------------------------------------------------------------
; fungsi : membaca status lock bit
; output : a (xxxxx321b)

Read_Lock_Bit
                    MOV   a,#24h
                    ACALL Send_Byte_SPI
                    MOV   a,#0
                    ACALL Send_Byte_SPI
                    MOV   a,#0
                    ACALL Send_Byte_SPI
                    MOV   a,#0
                    ACALL Send_Byte_SPI
                    RR    a
                    RR    a
                    ANL   a,#7
                    RET

;-------------------------------------------------------------------------
; fungsi : mengecek isi data serial dari EB500

Check_Serial
                    JBC   SrValid_F,CSrJ1
                    RET
CSrJ1
```

```
                    MOV    a,SrBuff_M


; ---------- cmd Erase chip

                    CJNE   a,#'E',CSrJ2
                    ACALL  Init_ISP
                    CJNE   a,#ISPOK_C,CSr1J1
                    MOV    a,#0ACh
                    ACALL  Send_Byte_SPI
                    MOV    a,#80h
                    ACALL  Send_Byte_SPI
                    MOV    a,#0
                    ACALL  Send_Byte_SPI
                    MOV    a,#0
                    ACALL  Send_Byte_SPI
                    ACALL  Delay_Erase
                    MOV    dptr,#EraseOK_T
                    ACALL  Send_Text_PC
                    AJMP   CsrJX
CSr1J1      MOV     dptr,#EraseErr_T
                    ACALL  Send_Text_PC
                    AJMP   CSrJX


; ---------- Cmd Blank check
```

```
CSrJ2           CJNE    a,#'B',CSrJ3
                ACALL   Init_ISP
                CJNE    a,#ISPOK_C,CSr2J1
                MOV     dptr,#0
                MOV     r7,#128
CSr2J3      MOV r6,#32
CSr2J2      MOV a,#020h
                ACALL   Send_Byte_SPI
                MOV     a,dph
                ACALL   Send_Byte_SPI
                MOV     a,dpl
                ACALL   Send_Byte_SPI
                MOV     a,#0
                ACALL   Send_Byte_SPI
                CJNE    a,#0FFh,CSr2J1
                INC     dptr
                CLR     a
                ACALL   Send_Byte_PC
                DJNZ    r6,CSr2J2
                DJNZ    r7,CSr2J3

                MOV     dptr,#BLChkOK_T
                ACALL   Send_Text_PC
                AJMP    CsrJX
CSr2J1      MOV dptr,#BLChkErr_T
```

```
                    ACALL  Send_Text_PC
                    AJMP   CSrJX

; ---------- Cmd write chip

CSrJ3               CJNE   a,#'W',CSrJ4
                    ACALL  Init_ISP
                    CJNE   a,#ISPOK_C,CSr3J1
                    MOV    dptr,#WriteReady_T
                    ACALL  Send_Text_PC
                    CLR    WriteCmd_F
                    MOV    th0,#TH0Val_C
                    MOV    tl0,#TL0Val_C
                    MOV    TOCtr_M,#0
                    SETB   StartWrite_F
                    MOV    CtrData0_M,#0
                    MOV    CtrData1_M,#0
                    SETB   tr0
                    AJMP   CSrJX
CSr3J1        MOV   dptr,#WriteErr_T
                    ACALL  Send_Text_PC
                    CLR    WriteCmd_F
                    AJMP   CSrJX

; ---------- Cmd verify chip
```

```
CSrJ4           CJNE    a,#'V',CSrJ5
                MOV     JData0_M,SrBuff_M+2
                MOV     JData1_M,SrBuff_M+1
                MOV     CtrData0_M,#0
                MOV     CtrData1_M,#0
                ACALL   Init_ISP
                CJNE    a,#ISPOK_C,CSr4J1
CSr4J2      MOV     a,#20h
                ACALL   Send_Byte_SPI
                MOV     a,CtrData1_M
                ACALL   Send_Byte_SPI
                MOV     a,CtrData0_M
                ACALL   Send_Byte_SPI
                MOV     a,#0
                ACALL   Send_Byte_SPI
                ACALL   Send_Byte_PC

                MOV     dph,CtrData1_M
                MOV     dpl,CtrData0_M
                INC     dptr
                MOV     CtrData1_M,dph
                MOV     CtrData0_M,dpl
                MOV     a,CtrData1_M
                CJNE    a,JData1_M,CSr4J2
```

```
                    MOV   a,CtrData0_M
                    CJNE  a,JData0_M,CSr4J2
                    AJMP  CSrJX

CSr4J1      MOV   dptr,#VerifyErr_T
                    ACALL Send_Text_PC
                    AJMP  CSrJX

; ---------- cmd Program lock bit

CSrJ5               CJNE  a,#'L',CSrJ6
                    ACALL Init_ISP
                    CJNE  a,#ISPOK_C,CSr5J1
                    MOV   a,SrBuff_M+1

                    CJNE  a,#1,CSr5J2
                    ACALL Program_Lock_bit
                    ACALL Read_Lock_Bit
                    JNB   a.0,CSr5J1
                    SJMP  CSr5J4
CSr5J2
                    CJNE  a,#2,CSr5J3
                    MOV   a,#2
                    ACALL Program_Lock_Bit
                    ACALL Read_Lock_Bit
```

```
                JNB     a.1,CSr5J1
                SJMP    CSr5J4
CSr5J3
                MOV     a,#3
                ACALL   Program_Lock_Bit
                ACALL   Read_Lock_Bit
                JNB     a.2,CSr5J1
CSr5J4          MOV     dptr,#LockBitOK_T
                ACALL   Send_Text_PC
                AJMP    CSrJX

CSr5J1          MOV     dptr,#LockBitErr_T
                ACALL   Send_Text_PC
                AJMP    CSrJX

CSrJ6

CSrJX           MOV     SrCtr_M,#0
                CLR     ri
                SETB    es
                RET

;-----------------------
;   MAIN PROGRAM
;-----------------------
```

```
Start
                    MOV   sp,#7
                    ACALL Init_Data

Main_Loop
                    ACALL Check_Button
                    ACALL Check_Serial
                    SJMP  Main_Loop

;-----------------------
;   BYTE CONSTANTS
;-----------------------
; panjang response = 2 byte + $13
EraseOK_T           DB    'EO',13,0
EraseErr_T          DB    'EE',13,0
BlChkOK_T           DB    'BO',13,0
BlChkErr_T          DB    'BE',13,0
WriteReady_T  DB    'WR',13,0
WriteErr_T          DB    'WE',13,0
WriteOK_T           DB    'WO',13,0
VerifyErr_T         DB    'VE',13,0
LockBitErr_T  DB    'LE',13,0
LockBitOK_T         DB    'LO',13,0
```

Listing Program Downloader AT89S51

```
unit Unit1;

interface

uses
 Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
 StdCtrls, VaClasses, VaComm, ExtCtrls, Grids, ComCtrls;

type
 TForm1 = class(TForm)
  GroupBox1: TGroupBox;
  Lockbit1: TRadioButton;
  Lockbit2: TRadioButton;
  Lockbit3: TRadioButton;
  ProgLB: TButton;
  GroupBox2: TGroupBox;
  OpenBIN: TButton;
  OpenHEX: TButton;
  GroupBox3: TGroupBox;
  Erase: TButton;
  BLCheck: TButton;
  Write: TButton;
  Verify: TButton;
  Auto: TButton;
  GroupBox4: TGroupBox;
```

```
COMPort: TComboBox;
Label1: TLabel;
Label2: TLabel;
Label3: TLabel;
Label4: TLabel;
Label5: TLabel;
FilenameLbl: TLabel;
FilesizeLbl: TLabel;
VaComm1: TVaComm;
Timer1: TTimer;
OpenBINDlg: TOpenDialog;
OpenHEXDlg: TOpenDialog;
Buffer: TMemo;
StatusBar1: TStatusBar;
ProgressBar1: TProgressBar;
procedure FormCreate(Sender: TObject);
procedure OpenBINClick(Sender: TObject);
procedure OpenHEXClick(Sender: TObject);
procedure EraseClick(Sender: TObject);
procedure Timer1Timer(Sender: TObject);
procedure VaComm1Data(Sender: TObject; Count: Integer);
procedure BLCheckClick(Sender: TObject);
procedure WriteClick(Sender: TObject);
procedure VerifyClick(Sender: TObject);
procedure AutoClick(Sender: TObject);
```

```
  procedure ProgLBClick(Sender: TObject);
 private
  function Byte2HexStr (Num : Byte) : String;
  function HexStr2Word (Num : String) : Word;
  procedure Closing_Command;
  procedure Starting_Command;
  { Private declarations }
 public
  { Public declarations }
 end;

const
 MaxAddr = 4095;

 // Cmd
 ChErase  = 0;
 BlChk    = 1;
 ChWrite  = 2;
 ChVerify = 3;
 Lockbit  = 4;

var
 Form1: TForm1;
 ByteBuffer, VerBuffer : Array [0..MaxAddr] of Byte;
 FileOpened : Boolean;
```

```
 Cmd : Byte;
 AddrCtr : Word;
 SrData : String;
 ByteSize : Word;
 AutoProg : Boolean;

implementation

{$R *.DFM}

function TForm1.HexStr2Word (Num : String) : Word;
var
 HexStr : String;
 Ctr,Ctr2 : Byte;
 DumWord,Mul : Word;
 DumChar : Char;

Begin
 HexStr := '0123456789ABCDEF';
 DumWord := 0;
 For Ctr := 1 to Length (Num) do Begin
  Mul := 1;
  For Ctr2 := 1 to (Ctr-1) do Mul := Mul * 16;
  DumChar := Upcase (Num [Length (Num) + 1 - Ctr]);
  For Ctr2 := 1 to 16 do
```

```
end;

procedure TForm1.OpenBINClick(Sender: TObject);
var
 FileBIN : File of byte;
 Filename : String;
 Fsize : Word;
 Ctr, Ctr2 : Word;
 DumByte : Byte;
 DumStr : String;

begin
 If OpenBINDlg.Execute then Begin
  Filename := OpenBINDlg.Files.Strings [0];
  AssignFile (FileBIN, Filename);

  {$I-}
  Reset (FileBIN);
  {$I+}
  If IOResult <> 0 then Begin
   MessageDlg ('Error opening file!', mtError, [mbOk], 0);
   Exit;
  End else Begin
   FilenameLbl.Caption := Filename;
   FSize := FileSize (FileBIN);
```

```
ByteSize := FSize;
FileSizeLbl.Caption := inttoStr (FSize) + ' byte(s)';
Seek (FileBIN, 0);
For Ctr := 1 to FileSize (FileBIN) do Begin
 Read (FileBIN, DumByte);
 ByteBuffer [Ctr-1] := DumByte;
End;
CloseFile (FileBIN);
Buffer.Lines.Clear;
Ctr := 0;
Ctr2 := 0;
DumStr := '0000: ';
While Ctr < FSize do Begin
 DumStr := DumStr + Byte2HexStr (ByteBuffer [Ctr]) + ' ';
 Inc (Ctr2);
 If Ctr2 = 16 then Begin
  Buffer.Lines.Add (DumStr);
  Ctr2 := 0;
  Inc (Ctr);
  DumStr := Byte2HexStr (Hi (Ctr)) + Byte2HexStr (Lo (Ctr)) + ': ';
  Dec (Ctr);
 End;
 Inc (Ctr);
End;
If Ctr2 <> 0 then Buffer.Lines.Add (DumStr);
```

```
    FileOpened := True;
   End;

  End;
end;

procedure TForm1.OpenHEXClick(Sender: TObject);
var
  FileHEX : Textfile;
  Filename : String;
  Ctr, Ctr2 : Word;
  JData, StAddr : Word;
  DumByte : Byte;
  DumStr : String;
  HiAddr : Word;

begin
If OpenHEXDlg.Execute then Begin
   Filename := OpenHEXDlg.Files.Strings [0];
   AssignFile (FileHEX, Filename);

   {$I-}
   Reset (FileHEX);
   {$I+}
   If IOResult <> 0 then Begin
```

```
 MessageDlg ('Error opening file!', mtError, [mbOk], 0);
 Exit;
End else Begin
 FilenameLbl.Caption := Filename;
 HiAddr := 0;
 While not (EOF (FileHEX)) do Begin
  ReadLn (FileHEX, DumStr);
  If DumStr [1] <> ':' then Begin
   MessageDlg ('HEX file error!', mtError, [mbOK], 0);
   CloseFile (FileHEX);
   Exit;
  End;
  JData := HexStr2Word (Copy (DumStr, 2, 2));
  StAddr := HexStr2Word (Copy (DumStr, 4, 4));
  If StAddr+JData-1 > MaxAddr then Begin
   MessageDlg ('Program is oversized!', mtError, [mbOK], 0);
   CloseFile (FileHEX);
   Exit;
  End;
  If StAddr+JData-1 > HiAddr then HiAddr := StAddr+JData-1;
  If Copy (DumStr, 8, 2) <> '01' then Begin
   For Ctr := 1 to JData do Begin
    DumByte := HexStr2Word (Copy (DumStr, (Ctr-1) * 2 + 10,2));
    ByteBuffer [StAddr + Ctr - 1] := DumByte;
   End;
```

```
    End;
  End;
  CloseFile (FileHEX);
  FileSizeLbl.Caption := InttoStr (HiAddr + 1) + ' byte(s)';
  ByteSize := HiAddr+1;
  Buffer.Lines.Clear;
  Ctr := 0;
  Ctr2 := 0;
  DumStr := '0000: ';
  While Ctr <= HiAddr do Begin
    DumStr := DumStr + Byte2HexStr (ByteBuffer [Ctr]) + ' ';
    Inc (Ctr2);
    If Ctr2 = 16 then Begin
      Buffer.Lines.Add (DumStr);
      Ctr2 := 0;
      Inc (Ctr);
      DumStr := Byte2HexStr (Hi (Ctr)) + Byte2HexStr (Lo (Ctr)) + ': ';
      Dec (Ctr);
    End;
    Inc (Ctr);
  End;
  If Ctr2 <> 0 then Buffer.Lines.Add (DumStr);
  FileOpened := True;
 End;
End;
```

```
end;

procedure TForm1.Closing_Command;
begin
 Vacomm1.Close;
 StatusBar1.Panels[1].Text := 'Ready';
 COMPort.Enabled := True;
 ProgLB.Enabled := True;
 Erase.Enabled := True;
 BLCheck.Enabled := True;
 Write.Enabled := True;
 Verify.Enabled := True;
 Auto.Enabled := True;
 OpenBIN.Enabled := True;
 OpenHEX.Enabled := True;
end;

procedure TForm1.Starting_Command;
begin
 StatusBar1.Panels[1].Text := 'Connecting...';
 Vacomm1.PortNum := COMPort.ItemIndex + 1;
 Try
  Vacomm1.Open;
  COMPort.Enabled := False;
  ProgLB.Enabled := False;
```

```
    Erase.Enabled := False;
    BLCheck.Enabled := False;
    Write.Enabled := False;
    Verify.Enabled := False;
    Auto.Enabled := False;
    OpenBIN.Enabled := False;
    OpenHEX.Enabled := False;
  except
    StatusBar1.Panels[1].Text := 'Ready';
  end;
end;

procedure TForm1.Timer1Timer(Sender: TObject);
begin
  Timer1.Enabled := False;
  MessageDlg ('No response!', mtError, [mbOK], 0);
  Closing_Command;
end;

procedure TForm1.VaComm1Data(Sender: TObject; Count: Integer);
var
  Ctr : Word;
  DataChar : Char;
```

```
begin
 Timer1.Enabled := False;
 If Cmd = ChVerify then Begin
  For Ctr := 1 to Count do begin
   Vacomm1.ReadChar (DataChar);
   VerBuffer [AddrCtr] := Ord (DataChar);
   Inc (AddrCtr);
  end;
  If AddrCtr >= ByteSize then Begin
{    for ctr:=1 to bytesize do begin
   buffer.lines.add (inttostr(verbuffer[ctr-1]));
   end;
   closing_command;
   exit;}
   For Ctr := 1 to ByteSize do Begin
    ProgressBar1.Position := Round ((Ctr/ByteSize)*100);
    If VerBuffer [Ctr-1] <> ByteBuffer [Ctr-1]
    then Begin
     MessageDlg ('Verify error at '+
      Byte2HexStr(Hi(Ctr-1))+Byte2HexStr(Lo(Ctr-1))+'h'
      , mtError, [mbOK], 0);
     Closing_Command;
     Exit;
    End;
   End;
```

```
    If Not (AutoProg) then Begin
      MessageDlg('Verify success!', mtInformation, [mbOK], 0);
      Closing_Command;
    End else Begin
      MessageDlg('Auto program success!', mtInformation, [mbOK], 0);
      Closing_Command;
    End;
    Exit;
  End else Timer1.Enabled := True;
  Exit;
End;
SrData := SrData + Vacomm1.ReadText;
If Pos (Chr(13), SrData) <> 0 then Begin
  SrData := Copy (SrData, Length (SrData) - 2, 2);
  Case Cmd of
    ChErase : Begin
      If SrData = 'EO' then Begin
        If Not (AutoProg) then Begin
          MessageDlg ('Erase chip success!', mtInformation, [mbOK], 0);
          Closing_Command;
        End else Begin
          Cmd  := BlChk;
          StatusBar1.Panels[1].Text := 'Blank checking...';
          SrData := '';
          Vacomm1.WriteText ('B'+Chr(13));
```

```
      Timer1.Enabled := True;
    End;
   end else Begin
    MessageDlg ('Erase chip failed!', mtError, [mbOK], 0);
    Closing_Command;
  End;
End;

BLChk : Begin
 If SrData = 'BO' then Begin
  If Not (AutoProg) then Begin
   MessageDlg ('Blank check success!', mtInformation, [mbOK], 0);
   Closing_Command;
  End else Begin
   Cmd := ChWrite;
   StatusBar1.Panels[1].Text := 'Writing chip...';
   SrData := '';
   Vacomm1.WriteText ('W'+Chr(Hi(ByteSize))+Chr(Lo(ByteSize)));
   Timer1.Enabled := True;
  End;
 end else Begin
  MessageDlg ('Blank check failed!', mtError, [mbOK], 0);
  Closing_Command;
 End;
End;
```

```
ChWrite : Begin
 If SrData = 'WR' then Begin
  For Ctr := 1 to ByteSize do Begin
   Vacomm1.WriteChar (Chr (ByteBuffer [Ctr-1]));
   progressbar1.position := round ((ctr/bytesize)*100);
   StatusBar1.Panels[1].Text := 'Writing chip '+
    Inttostr(Round((Ctr/ByteSize)*100))+'%';
  End;
  SrData := '';
  Timer1.Enabled := True;
 End else
 If SrData = 'WO' then Begin
  If Not (AutoProg) then Begin
   MessageDlg ('Write chip success!', mtInformation, [mbOK], 0);
   Closing_Command;
  End else Begin
   Cmd := ChVerify;
   StatusBar1.Panels[1].Text := 'Verifying chip...';
   ProgressBar1.Position := 0;
   SrData := '';
   Vacomm1.WriteText ('V'+Chr(Hi(ByteSize))+Chr(Lo(ByteSize))+Chr(13));
   AddrCtr := 0;
   Timer1.Enabled := True;
  End;
```

```
      End else Begin
       MessageDlg ('Write chip failed!', mtError, [mbOK], 0);
       Closing_Command;
      End;
    End;

    LockBit : Begin
     If SrData = 'LO' then Begin
      MessageDlg ('Program lock bit success!', mtInformation, [mbOK], 0);
      Closing_Command;
     end else Begin
      MessageDlg ('Program lock bit failed!', mtError, [mbOK], 0);
      Closing_Command;
     End;
    End;
   End;
 end else Timer1.Enabled := True;
end;

procedure TForm1.EraseClick(Sender: TObject);
begin
 AutoProg := False;
 Cmd := ChErase;
 Starting_Command;
 StatusBar1.Panels[1].Text := 'Erasing chip...';
```

```
 SrData := '';
 Vacomm1.WriteText ('E'+Chr(13));
 Timer1.Enabled := True;
end;

procedure TForm1.BLCheckClick(Sender: TObject);
begin
 AutoProg := False;
 Cmd  := BlChk;
 Starting_Command;
 StatusBar1.Panels[1].Text := 'Blank checking...';
 SrData := '';
 Vacomm1.WriteText ('B'+Chr(13));
 Timer1.Enabled := True;
end;

procedure TForm1.WriteClick(Sender: TObject);
begin
 If Not (FileOpened) then
  MessageDlg ('Buffer empty!', mtError, [mbOK], 0)
 else Begin
  AutoProg := False;
  Cmd := ChWrite;
  Starting_Command;
  StatusBar1.Panels[1].Text := 'Writing chip...';
```

```
  SrData := '';
  Vacomm1.WriteText ('W'+Chr(Hi(ByteSize))+Chr(Lo(ByteSize)));
  Timer1.Enabled := True;
 End;
end;

procedure TForm1.VerifyClick(Sender: TObject);
begin
 If Not (FileOpened) then
  MessageDlg ('Buffer empty!', mtError, [mbOK], 0)
 else Begin
  AutoProg := False;
  Cmd := ChVerify;
  Starting_Command;
  StatusBar1.Panels[1].Text := 'Verifying chip...';
  ProgressBar1.Position := 0;
  SrData := '';
  Vacomm1.WriteText ('V'+Chr(Hi(ByteSize))+Chr(Lo(ByteSize))+Chr(13));
  AddrCtr := 0;
  Timer1.Enabled := True;
 End;
end;

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

```
 If Not (FileOpened) then
  MessageDlg ('Buffer empty!', mtError, [mbOK], 0)
 else Begin
  AutoProg := True;
  Cmd := ChErase;
  Starting_Command;
  StatusBar1.Panels[1].Text := 'Erasing chip...';
  SrData := '';
  Vacomm1.WriteText ('E'+Chr(13));
  Timer1.Enabled := True;
 End;
end;

procedure TForm1.ProgLBClick(Sender: TObject);
var
 DumByte : Byte;

begin
 Cmd := Lockbit;
 Starting_Command;
 StatusBar1.Panels[1].Text := 'Programming lock bit...';
 SrData := '';
 If LockBit1.Checked then DumByte := 1;
 If LockBit2.Checked then DumByte := 2;
 If LockBit3.Checked then DumByte := 3;
```

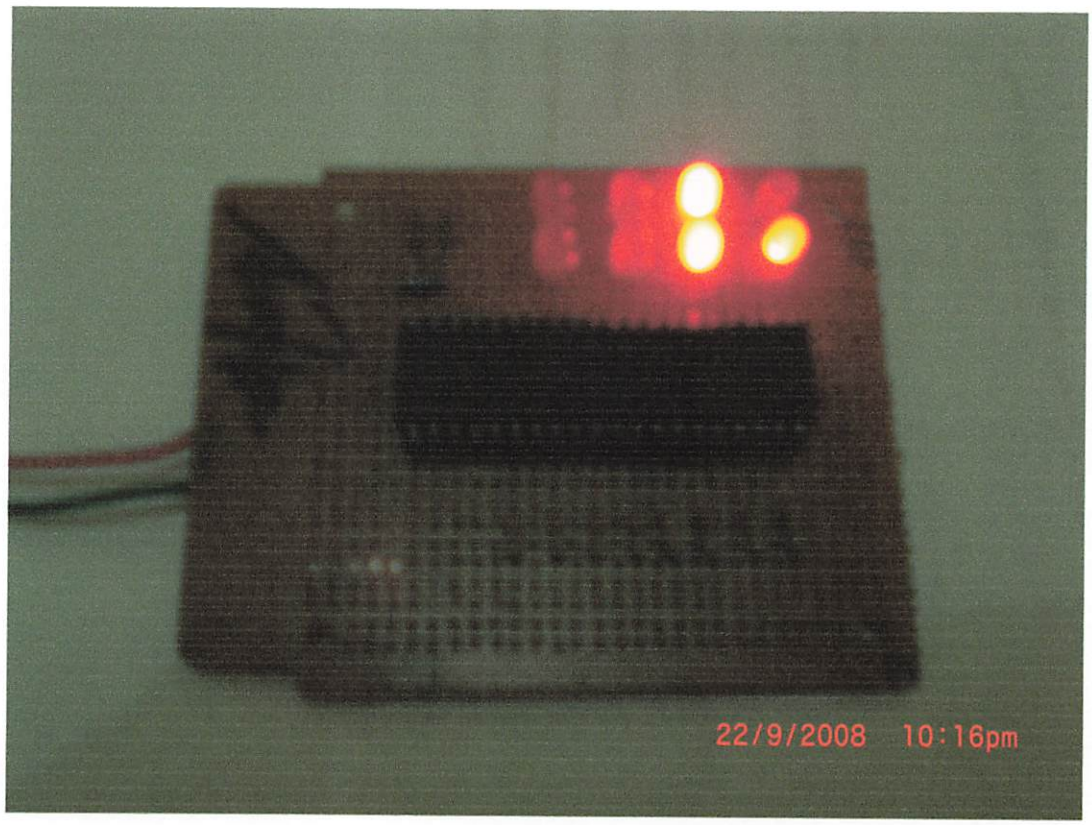
```
  Vacomm1.WriteText ('L'+Chr(DumByte)+Chr(13));
  AddrCtr := 0;
  Timer1.Enabled := True;
end;


end.
```

Dokumentasi Alat Downloader AT89S51 Menggunakan Teknologi Bluetooth

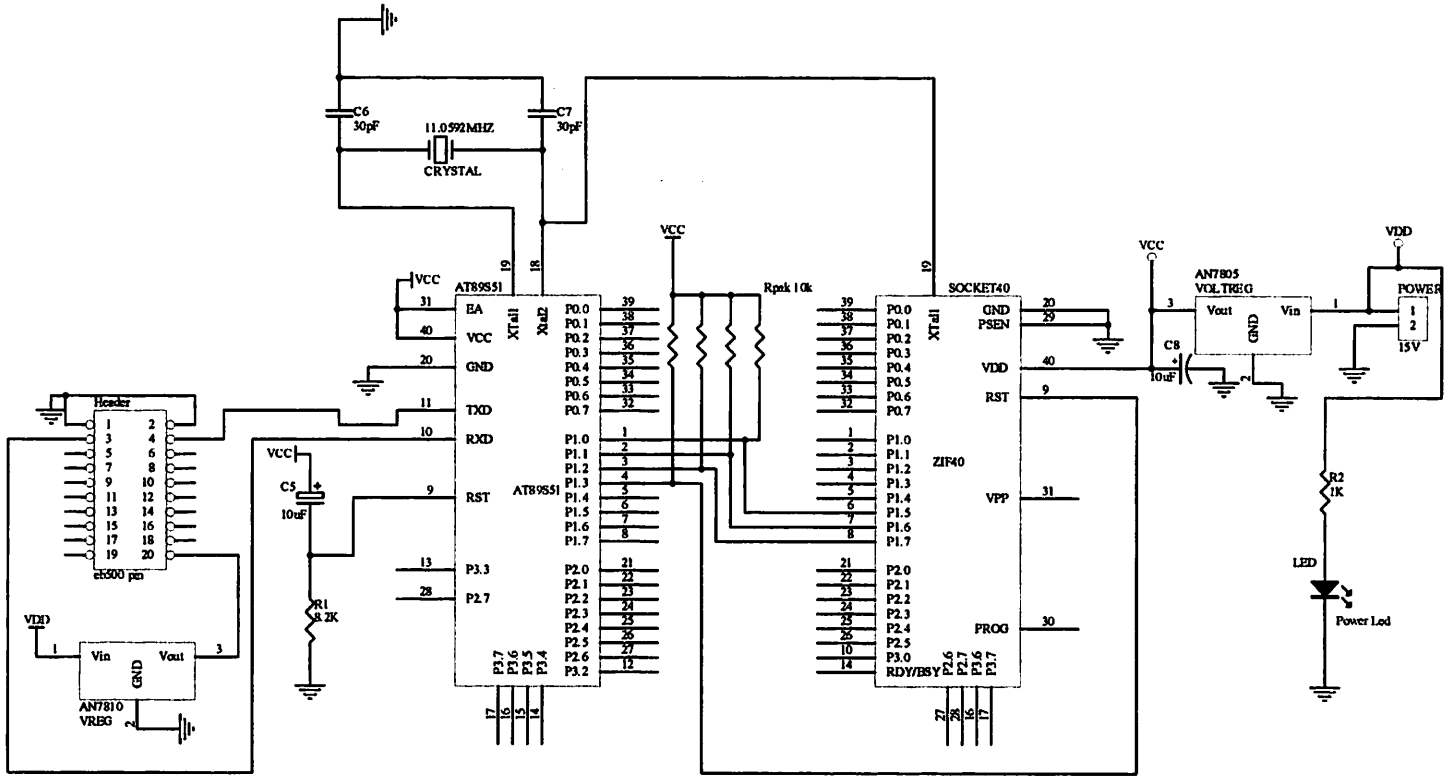
Alat Downloader Mikrokontroller AT89S51 & Alat tester LED

Downloader Mikrokontroller AT89S51 saat Melakukan Pemrograman ic AT89S51 lewat Bluetooth



Uji Coba ic Mikrokontroller yang telah di-isi program menggunakan Alat Downloader AT89S51

Downloader dan alat uji LED



Title	Downloader Mikrokontroler AT89S51 Menggunakan Teknologi Bluetooth	
Size: B	Number: 01	Revision: 02
Date: 10/13/2008	Sheet of	
File: F:\Yudie\..\bludownsrv.Sch	Drawn By: *Yudie*	

Data Sheet

tures

- npatible with MCS-51® Products
- 3ytes of In-System Programmable (ISP) Flash Memory
  - Endurance: 1000 Write/Erase Cycles
- ' to 5.5V Operating Range
- y Static Operation: 0 Hz to 33 MHz
- ee-level Program Memory Lock
- x 8-bit Internal RAM
- 'rogrammable I/O Lines
- 16-bit Timer/Counters
- Interrupt Sources
- Duplex UART Serial Channel
- '-power Idle and Power-down Modes
- rrupt Recovery from Power-down Mode
- chdog Timer
- l Data Pointer
- 'er-off Flag
- t Programming Time
- ible ISP Programming (Byte and Page Mode)

cription

T89S51 is a low-power, high-performance CMOS 8-bit microcontroller with 4K of in-system programmable Flash memory. The device is manufactured using 's high-density nonvolatile memory technology and is compatible with the indus- ndard 80C51 instruction set and pinout. The on-chip Flash allows the program ry to be reprogrammed in-system or by a conventional nonvolatile memory pro- ner. By combining a versatile 8-bit CPU with in-system programmable Flash on a lithic chip, the Atmel AT89S51 is a powerful microcontroller which provides a -flexible and cost-effective solution to many embedded control applications.

T89S51 provides the following standard features: 4K bytes of Flash, 128 bytes of 32 I/O lines, Watchdog timer, two data pointers, two 16-bit timer/counters, a five- two-level interrupt architecture, a full duplex serial port, on-chip oscillator, and circuitry. In addition, the AT89S51 is designed with static logic for operation to zero frequency and supports two software selectable power saving modes. le Mode stops the CPU while allowing the RAM, timer/counters, serial port, and pt system to continue functioning. The Power-down mode saves the RAM con- ut freezes the oscillator, disabling all other chip functions until the next external pt or hardware reset.

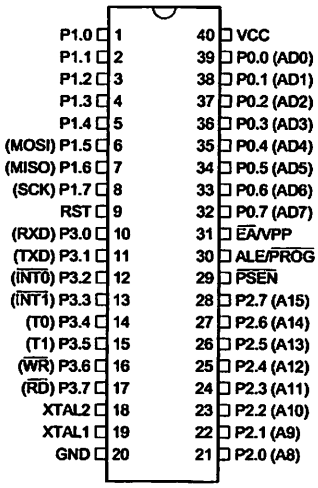
8-bit Microcontroller with 4K Bytes In-System Programmable Flash

AT89S51

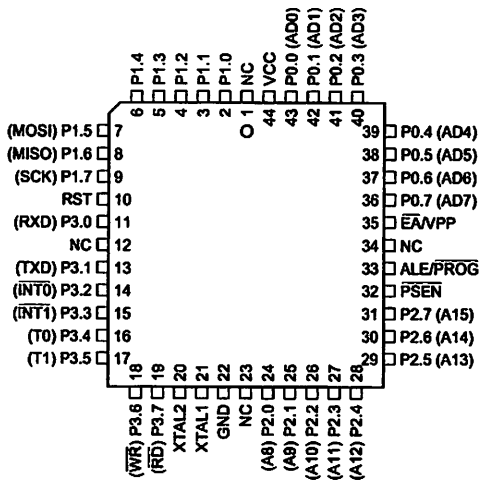
Rev. 2487A–10/01

Configurations

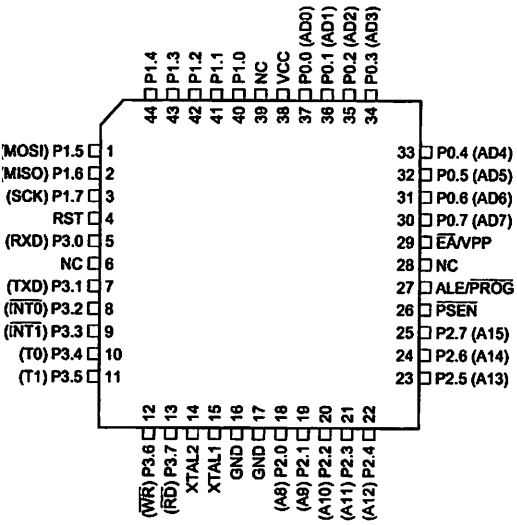
PDIP



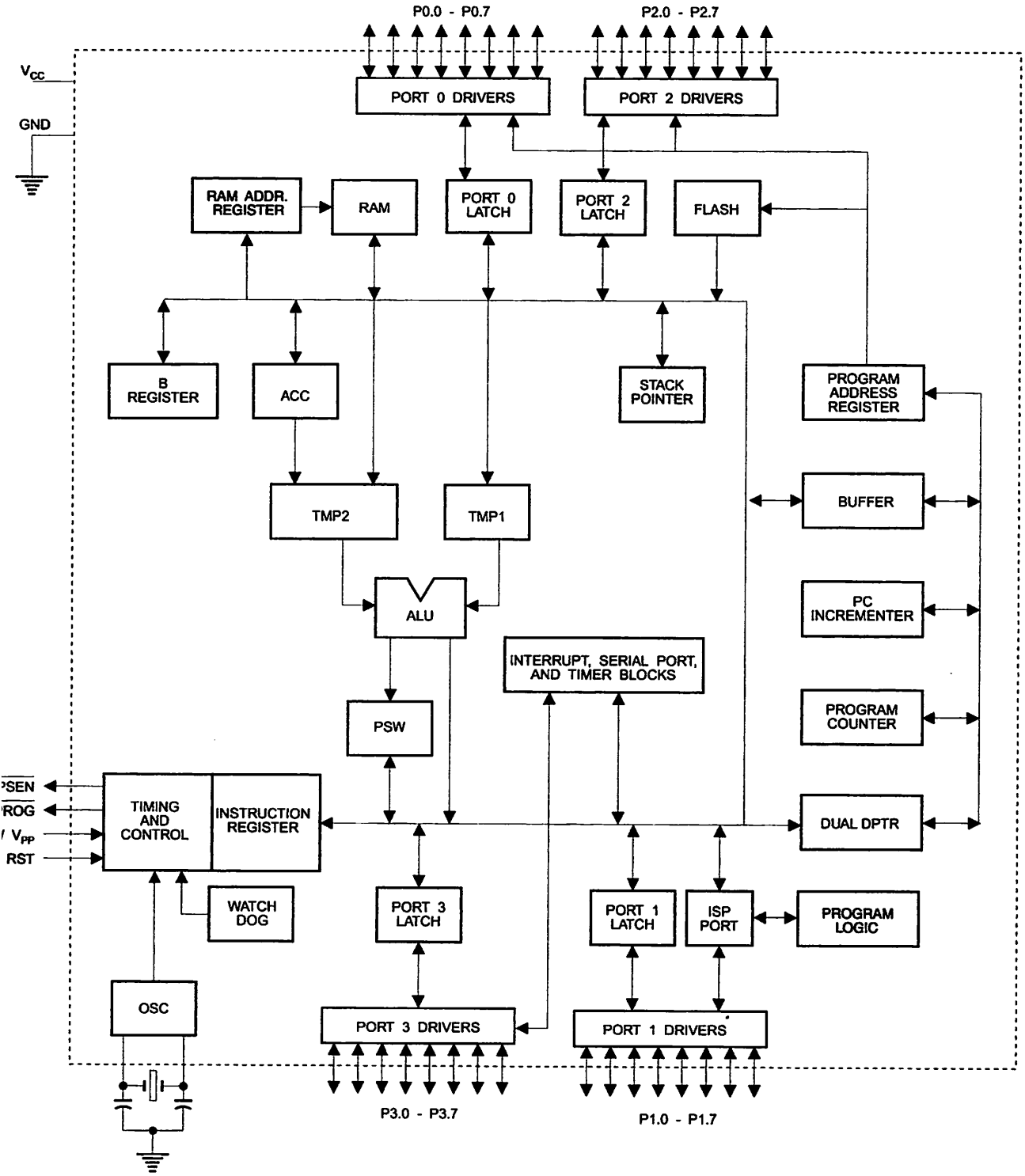
PLCC



TQFP



ck Diagram


P0.0 - P0.7
P2.0 - P2.7
V_{CC}
GND
PORT 0 DRIVERS
PORT 2 DRIVERS
RAM ADDR. REGISTER
RAM
PORT 0 LATCH
PORT 2 LATCH
FLASH
B REGISTER
ACC
STACK POINTER
PROGRAM ADDRESS REGISTER
TMP2
TMP1
BUFFER
ALU
PC INCREMENTER
INTERRUPT, SERIAL PORT, AND TIMER BLOCKS
PSW
PROGRAM COUNTER
PSEN
PROG
/ V_{PP}
RST
TIMING AND CONTROL
INSTRUCTION REGISTER
DUAL DPTR
WATCH DOG
PORT 3 LATCH
PORT 1 LATCH
ISP PORT
PROGRAM LOGIC
OSC
PORT 3 DRIVERS
PORT 1 DRIVERS
P3.0 - P3.7
P1.0 - P1.7

Description

Supply voltage.

Ground.

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.**

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.

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

Port Pin	Alternate Functions
P1.5	MOSI (used for In-System Programming)
P1.6	MISO (used for In-System Programming)
P1.7	SCK (used for In-System Programming)

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.

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 AT89S51, 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)

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.

/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.

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

When the AT89S51 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.

PP 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.

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

2 Output from the inverting oscillator amplifier

ecial
nction
gisters

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

e 1. AT89S51 SFR Map and Reset Values

8H								0FFH
0H	B 00000000							0F7H
8H								0EFH
0H	ACC 00000000							0E7H
8H								0DFH
0H	PSW 00000000							0D7H
8H								0CFH
0H								0C7H
8H	IP XX000000							0BFH
0H	P3 11111111							0B7H
8H	IE 0X000000							0AFH
0H	P2 11111111		AUXR1 XXXXXXX0				WDTRST XXXXXXXX		0A7H
3H	SCON 00000000	SBUF XXXXXXXX							9FH
0H	P1 11111111								97H
3H	TCON 00000000	TMOD 00000000	TL0 00000000	TL1 00000000	TH0 00000000	TH1 00000000	AUXR XXX00XX0		8FH
0H	P0 11111111	SP 00000111	DP0L 00000000	DP0H 00000000	DP1L 00000000	DP1H 00000000		PCON 0XXX0000	87H

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.

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

Table 2. AUXR: Auxiliary Register

AUXR Address = 8EH Reset Value = XXX00XX0B

Not Bit Addressable

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

– Reserved for future expansion

DISALE 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

DISRTO Disable/Enable Reset out

DISRTO

0 Reset pin is driven High after WDT times out

1 Reset pin is input only

WDIDLE Disable/Enable WDT in IDLE mode

WDIDLE

0 WDT continues to count in IDLE mode

1 WDT halts counting in IDLE mode

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

Power 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 up. It can be set and rest under software control and is not affected by reset.

Table 3. AUXR1: Auxiliary Register 1

AUXR1

Address = A2H

Reset Value = XXXXXXX0B

Not Bit Addressable

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

– Reserved for future expansion

DPS Data Pointer Register Select

DPS

0 Selects DPTR Registers DP0L, DP0H

1 Selects DPTR Registers DP1L, DP1H

nory anization

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.

ıram Memory

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

On the AT89S51, if $\overline{EA}$ is connected to V_{CC}, program fetches to addresses 0000H through FFFH are directed to internal memory and fetches to addresses 1000H through FFFFH are directed to external memory.

ı Memory

The AT89S51 implements 128 bytes of on-chip RAM. The 128 bytes are accessible via direct and indirect addressing modes. Stack operations are examples of indirect addressing, so the 128 bytes of data RAM are available as stack space.

chdog er e-time bled with et-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.

ıg 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.

T During ver-down 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 AT89S51 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 AT89S51 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.

RT

The UART in the AT89S51 operates the same way as the UART in the AT89C51. For further information on the UART operation, refer to the ATMEL Web site (http://www.atmel.com). From the home page, select 'Products', then '8051-Architecture Flash Microcontroller', then 'Product Overview'.

er 0 and 1

Timer 0 and Timer 1 in the AT89S51 operate the same way as Timer 0 and Timer 1 in the AT89C51. For further information on the timers' operation, refer to the ATMEL Web site (http://www.atmel.com). From the home page, select 'Products', then '8051-Architecture Flash Microcontroller', then 'Product Overview'.

rupts

The AT89S51 has a total of five interrupt vectors: two external interrupts ($\overline{INT0}$ and $\overline{INT1}$), two timer interrupts (Timers 0 and 1), and the serial port interrupt. These interrupts are all shown in Figure 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 4 shows that bit position IE.6 is unimplemented. In the AT89S51, bit position IE.5 is also unimplemented. User software should not write 1s to these bit positions, since they may be used in future AT89 products.

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

Table 4. Interrupt Enable (IE) Register

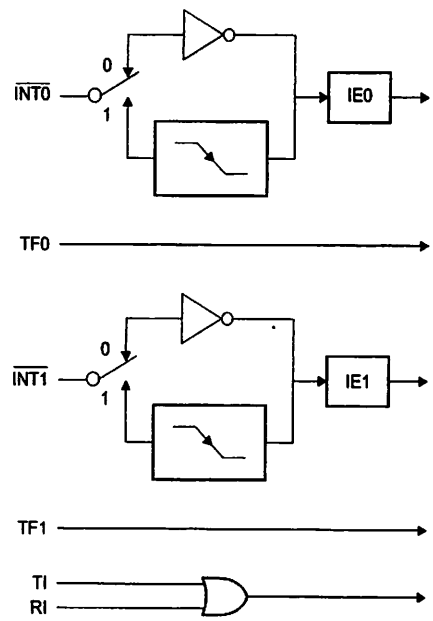
(MSB)							(LSB)
EA	–	–	ES	ET1	EX1	ET0	EX0

Enable Bit = 1 enables the interrupt.

Enable Bit = 0 disables the interrupt.

Symbol	Position	Function
EA	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
–	IE.5	Reserved
ES	IE.4	Serial Port interrupt enable bit
ET1	IE.3	Timer 1 interrupt enable bit
EX1	IE.2	External interrupt 1 enable bit
ET0	IE.1	Timer 0 interrupt enable bit
EX0	IE.0	External interrupt 0 enable bit
User software should never write 1s to reserved bits, because they may be used in future AT89 products.		

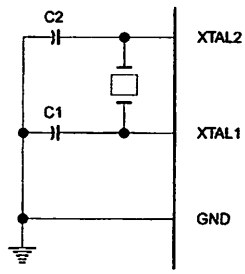
Figure 1. Interrupt Sources



illator racteristics

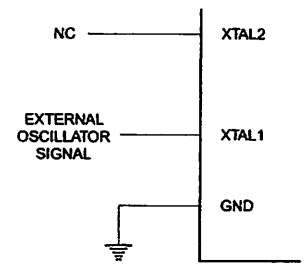
XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier that can be configured for use as an on-chip oscillator, as shown in Figure 2. Either a quartz crystal or ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected while XTAL1 is driven, as shown in Figure 3. 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.

Figure 2. Oscillator Connections



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

Figure 3. External Clock Drive Configuration



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 function 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.

er-down e

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 activation of an enabled external interrupt into $\overline{INT0}$ or $\overline{INT1}$. 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.

Table 5. 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

gram nory Lock

The AT89S51 has three lock bits that can be left unprogrammed (U) or can be programmed (P) to obtain the additional features listed in the following table.

Table 6. 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.

gramming Flash – allel Mode

The AT89S51 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 AT89S51 code memory array is programmed byte-by-byte.

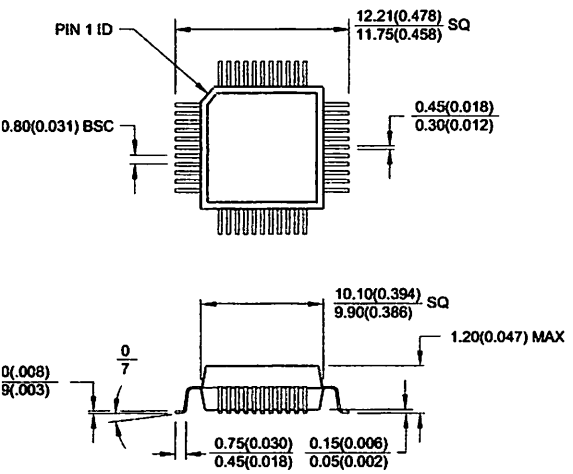
Programming Algorithm: Before programming the AT89S51, the address, data, and control signals should be set up according to the Flash programming mode table and Figures 13 and 14. To program the AT89S51, take the following steps:

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 AT89S51 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.

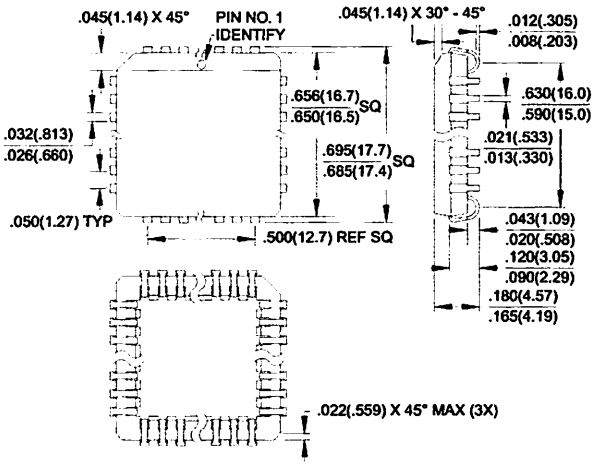
kaging Information

A, 44-lead, Thin (1.0 mm) Plastic Gull Wing Quad at Package (TQFP)
mensions in Millimeters and (Inches)*

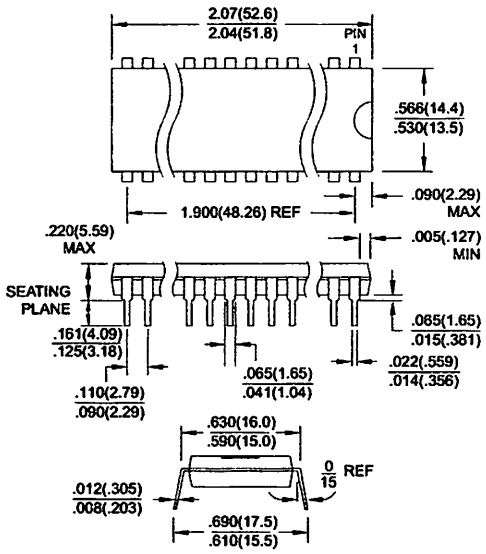


ontrolling dimension: millimeters

44J, 44-lead, Plastic J-leaded Chip Carrier (PLCC)
Dimensions in Inches and (Millimeters)



P6, 40-pin, 0.600" Wide, Plastic Dual Inline ckage (PDIP)
mensions in Inches and (Millimeters)
DEC STANDARD MS-011 AC



ıel Headquarters

ıorate Headquarters
25 Orchard Parkway
n Jose, CA 95131
L (408) 441-0311
X (408) 487-2600

pe
nel SarL
ute des Arsenaux 41
sa Postale 80
l-1705 Fribourg
itzerland
L (41) 26-426-5555
X (41) 26-426-5500

nel Asia, Ltd.
om 1219
inachem Golden Plaza
Mody Road Tsimhatsui
st Kowloon
ng Kong
L (852) 2721-9778
X (852) 2722-1369

n
nel Japan K.K.
Tonetsu Shinkawa Bldg.
4-8 Shinkawa
uo-ku, Tokyo 104-0033
ɔan
L (81) 3-3523-3551
X (81) 3-3523-7581

Atmel Product Operations

Atmel Colorado Springs
1150 E. Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906
TEL (719) 576-3300
FAX (719) 540-1759

Atmel Grenoble
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
TEL (33) 4-7658-3000
FAX (33) 4-7658-3480

Atmel Heilbronn
Theresienstrasse 2
POB 3535
D-74025 Heilbronn, Germany
TEL (49) 71 31 67 25 94
FAX (49) 71 31 67 24 23

Atmel Nantes
La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
TEL (33) 0 2 40 18 18 18
FAX (33) 0 2 40 18 19 60

Atmel Rousset
Zone Industrielle
13106 Rousset Cedex, France
TEL (33) 4-4253-6000
FAX (33) 4-4253-6001

Atmel Smart Card ICs
Scottish Enterprise Technology Park
East Kilbride, Scotland G75 0QR
TEL (44) 1355-357-000
FAX (44) 1355-242-743

e-mail
literature@atmel.com

Web Site
http://www.atmel.com

2487A–10/01/xM

mbeddedBlue™ 500

er Manual

rt Number 0000033 – Revision A

st revised on December 4, 2003 – Printed in the United States of America

Engineering, Inc.
360 C Danielson Court
way, CA 92064

ible of Contents

ible of Figures

ble of Tables

Introduction

ıgratulations on your purchase of the EmbeddedBlue 500 (eb500) module. The eb500 ɔule provides Bluetooth® connectivity for 8/16 bit microcontroller applications without ˈing to know the details of Bluetooth technology. Hobbyists, developers, and OEMs can ɜ advantage of advanced wireless connectivity with this easy to use module.

ɨ eb500 module provides a point to point connection much like a standard serial cable. ınections are made dynamically and can be established between two eb500 modules or eb500 module and a standard Bluetooth v1.1 device. Devices can be dynamically ɔovered and connected in an ad-hoc manner.

ɜnual Conventions

ɒw is a list of typographical conventions used in this manual:

`Text in this font`

- `Is used to show data that is sent to the eb500.`
- `Inside a gray box is used to show data that is sent from the eb500.`

`Text in this font`

- `Is used to show source code.`

In the eb500 Commands section of this manual

- Required parameters and placeholders appear in standard lowercase type.
- Placeholders appear in *italics*. For example, if *address* shows up in a syntax line, the actual address of the device must be entered.
- Required parameter options are separated by a vertical bar |.
- Optional parameters are enclosed in brackets [ ].

eb500 Basics

e eb500 supports two main operating modes: command mode and data mode. Upon wer up the eb500 enters command mode and is ready to accept serial commands. The tory default communication parameters are 9600 Baud, 8 Data Bits, 1 Stop Bit, No Parity, d No Flow Control. The eb500 supports commands to modify the baud rate and flow ntrol settings.

ommand Mode

this mode there are a number of commands that can be sent to change the baud rate, ate other devices that are in range, check the firmware version, etc. All commands are nt using visible ASCII characters (123 is 3 bytes "123"). Upon the successful transmission a command, the ACK string will be returned. If there is a problem in the syntax of the nsmission a NAK string is returned. After either the ACK or NAK, a carriage-return <CR> aracter is returned. When a prompt (<CR> followed by a '>') is returned, it means that the 500 radio is in the idle state and is waiting for another command. White spaces do matter d are used to separate argument parameters of the command and a carriage-return R> (ASCII 13) is used to mark the end of the command.

ata Mode

ce the eb500 radio is connected to another Bluetooth device, the eb500 automatically itches into data mode. All data transmitted while in this mode will be sent to the remote vice and, therefore, NO further commands can be sent until the eb500 radio is connected or switched back to command mode by use of the mode control I/O line or the itch to Command Mode command.

e connection status line of the eb500 module can be monitored to determine if there is an ive connection. Additionally, whenever a connection is present, the Connection Status D (Figure 1) on the eb500 module will be on.

ɔ Lines

e eb500 module features a 20 pin header for connecting to the Parallax AppMod header. ull device pinout is available in the Technical Specifications section of this manual. There ; several pins that are important when performing the exercises in the Establishing a nnection and Communications sections of this manual.

Pin 3 of the eb500 module, which aligns with the pin designated "P0" of the AppMod header, is the UART data output pin.

Pin 4 of the eb500 module, which aligns with the pin designated "P1" of the AppMod header, is the UART data input pin.

Pin 8 of the eb500 module, which aligns with the pin designated "P5" of the AppMod header, is the Connection Status pin. A BASIC Stamp application can interrogate this pin to determine the connection status of the eb500 radio.

Pin 9 of the eb500 module, which aligns with the pin designated "P6" of the AppMod header, is the Mode Control pin. A BASIC Stamp application can drive this pin high to enter Data Mode or low to enter Command Mode.

esetting the eb500 to the Factory Default Settings

e eb500 module can be reset to the factory default settings by shorting Pin 8 and Pin 9 d then applying power to the eb500 module.

witching between Data Mode and Command Mode

ıen a Connection command is issued, the eb500 attempts to establish a connection to the vice with the address specified in the command. Once a connection is established, the 500 switches into data mode. At this point all data sent to the eb500 is transmitted to the note Bluetooth device over the wireless link. It is possible to switch from data mode to mmand mode, issue commands, and then return to data mode, while maintaining a nnection. The eb500 allows you to switch between data mode and command mode by uing the Switch to Command Mode and Return to Data Mode serial commands or by ving the mode control I/O line (Pin 9) of the eb500 module.

e following BASIC Stamp application uses the Switch to Command Mode and Return to ta Mode serial commands to switch between data mode and command mode. This olication is available in electronic form on the accompanying CD in the Samples folder in file CmdModeSoft.bs2.

```
'{$STAMP BS2}
szData VAR BYTE(20)
'Wait for the eb500 radio to be ready
PAUSE 1000
'Connect to the remote device
```

```
SEROUT 1,84,["con 00:0C:84:00:07:D8",CR]
SERIN 0,84,[WAIT("ACK",CR)]
WaitForConnection:
      IF in5 = 0 THEN WaitForConnection

DEBUG "Connected",CR

SEROUT 1,84,["This text is sent in data mode",CR]

'Switch to Command Mode
PAUSE 2000
SEROUT 1,84,["+++"]
SERIN 0,84,[WAIT(CR,">")]

DEBUG "In Command Mode",CR

'Get the eb500 Bluetooth Address
SEROUT 1,84,["get addr",CR]
SERIN 0,84,[WAIT("ACK",CR)]
'Read the local address from the get command
SERIN 0,84,[STR szData\17]
SERIN 0,84,[WAIT(CR,">")]
szData(17) = 0
DEBUG STR szData\17,CR

'Return to Data Mode
SEROUT 1,84,["ret",CR]
SERIN 0,84,[WAIT("ACK",CR)]

DEBUG "In Data Mode",CR

SEROUT 1,84,["My Bluetooth address is ",STR szData,CR]

'Switch to Command Mode
PAUSE 2000
```

```
SEROUT 1,84,["+++"]
SERIN 0,84,[WAIT(CR,">")]

DEBUG "In Command Mode",CR

'Disconnect from remote device
SEROUT 1,84,["dis",CR]
SERIN 0,84,[WAIT(CR,">")]
DEBUG "Disconnected",CR
```

e following BASIC Stamp application uses the mode control I/O line of the eb500 module switch between data mode and command mode. Switching between data mode and nmand mode via the mode control I/O line is preferred, as it is faster than the serial thod. This application is available in electronic form on the accompanying CD in the nples folder in the file CmdModeHard.bs2.

```
'{$STAMP BS2}
szData VAR BYTE(20)
'Wait for the eb500 radio to be ready
PAUSE 1000
'Connect to the remote device
SEROUT 1,84,["con 00:0C:84:00:07:D8",CR]
SERIN 0,84,[WAIT("ACK",CR)]
WaitForConnection:
      IF in5 = 0 THEN WaitForConnection

DEBUG "Connected",CR

SEROUT 1,84,["This text is sent in data mode",CR]

'I/O Line 6 allows us to switch to Command Mode
OUTPUT 6
'Switch to Command Mode
LOW 6
SERIN 0,84,[WAIT(CR,">")]

DEBUG "In Command Mode",CR
```

```
'Get the eb500 Bluetooth Address
SEROUT 1,84,["get addr",CR]
SERIN 0,84,[WAIT("ACK",CR)]
'Read the local address from the get command
SERIN 0,84,[STR szData\17]
SERIN 0,84,[WAIT(CR,">")]
szData(17) = 0
DEBUG STR szData\17,CR

'Return to Data Mode
HIGH 6
PAUSE 50

DEBUG "In Data Mode",CR

SEROUT 1,84,["My Bluetooth address is ",STR szData,CR]

'Switch to Command Mode
LOW 6
SERIN 0,84,[WAIT(CR,">")]

DEBUG "In Command Mode",CR

'Disconnect from remote device
SEROUT 1,84,["dis",CR]
SERIN 0,84,[WAIT(CR,">")]
DEBUG "Disconnected",CR
```

ASIC Stamp Application Debugging

ıen debugging your BASIC Stamp application that uses an eb500 it is important that the SIC Stamp application and the eb500 are in sync. When the BASIC Stamp Editor begins downloading process the BASIC Stamp is reset; however, this reset does not reset the 500. This can cause an existing application on the BASIC Stamp to begin executing, ich can lead to a situation where the new application and the eb500 are not in sync. It is ssible that the eb500 could be in Data Mode or in an unpredictable command mode state, to the execution of the existing BASIC Stamp application. Therefore, during the lication debugging process, it is recommended that the following code be inserted at the ginning of your BASIC Stamp application, before you read or set any I/O points or issue commands to the eb500.

```
'*****************************************************************
      IF in5 = 0 THEN ClearCmd
      DEBUG "eb500 Connected (in Data Mode)",CR
      'Switch to Command Mode
      LOW 6
      SERIN 0,84,[WAIT(CR,">")]
      'Disconnect from the remote device
      SEROUT 1,84,["dis",CR]
      SERIN 0,84,[WAIT(CR,">")]
      GOTO Start
ClearCmd:
      DEBUG "eb500 in Command Mode",CR
      'Issue a carriage-return to clear any commands
      SEROUT 1,84,[CR]
      SERIN 0,84,[WAIT(">")]
'*****************************************************************
```

Hardware Connections

e eb500 module is designed to interface with a 5V CMOS signal environment. It supports ower supply of 5 – 12V and can be connected directly to boards supporting the Parallax pMod header. When inserting the eb500 module to any of the supported Parallax boards, s important that Pin 1 of the eb500 module, marked with a white dot and a square (Figure is inserted into the VSS pin of the AppMod header on the Parallax boards. A full device out is available in the Technical Specifications section of this manual.

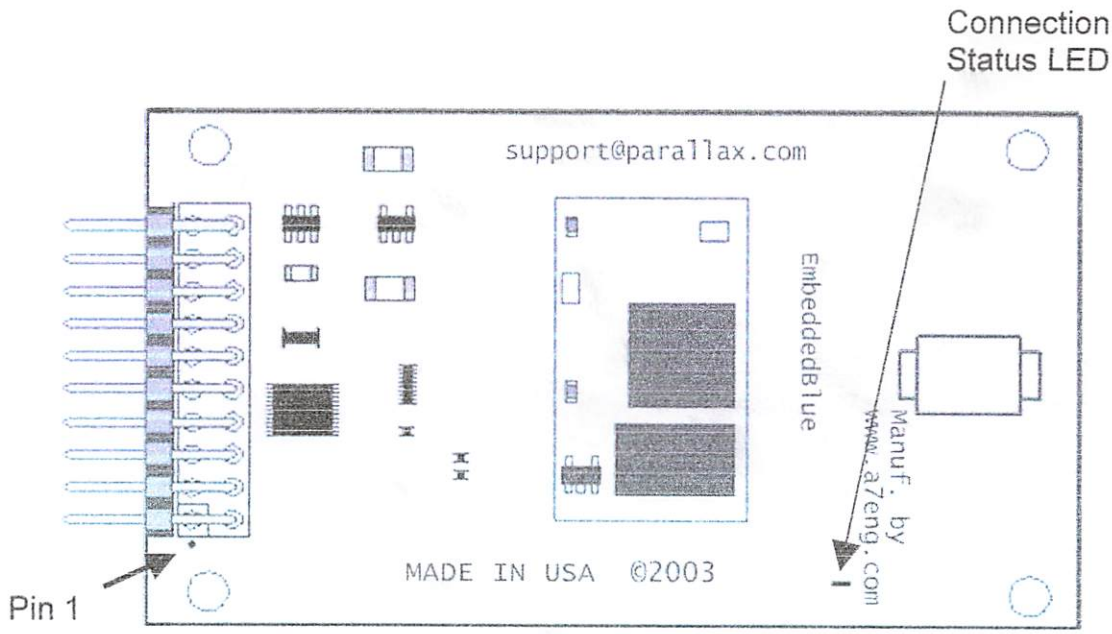


Figure 1: eb500 Module

oard Of Education

The Board Of Education (BOE) contains an AppMod header and supports a direct connection with the eb500 module. On the Board of Education, the AppMod header is labeled X1 (Figure 2). When inserting the eb500 module into the Board of Education AppMod header, assure that you insert Pin 1 of the eb500 module, marked with a white dot and a square, into the VSS pin of the AppMod header.

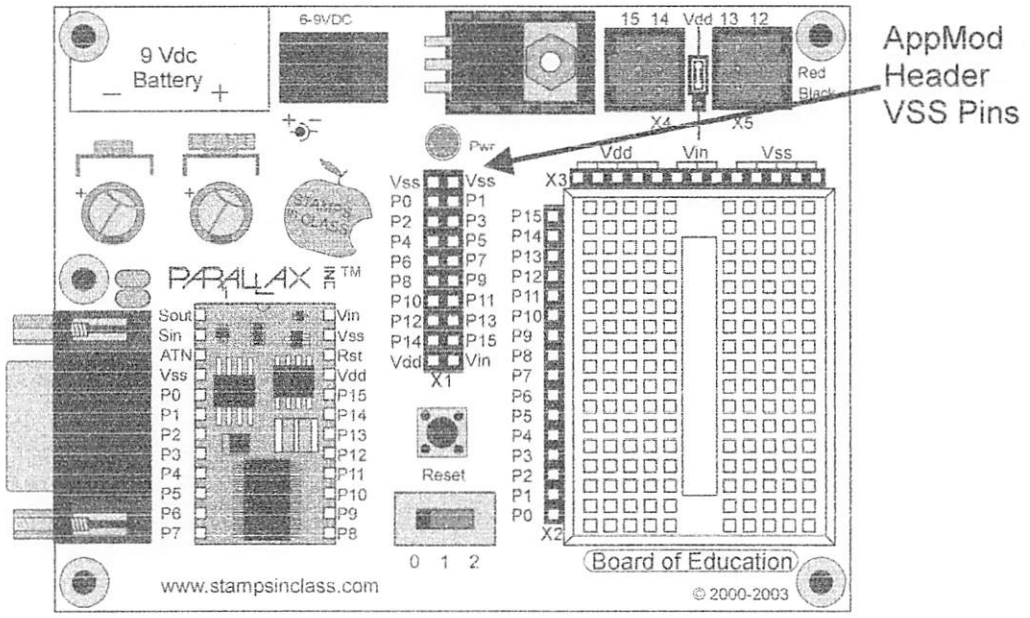


Figure 2: Board of Education Board

asic Stamp Activity Board

The Basic Stamp Activity Board contains an AppMod header and supports a direct connection with the eb500 module when using a BS2 processor. On the Basic Stamp Activity Board, the AppMod header is labeled X7 (Figure 3). When inserting the eb500 module into the Basic Stamp Activity Board AppMod header, assure that you insert Pin 1 of the eb500 module, marked with a white dot and a square, into the VSS pin (Pin 1) of the AppMod header.

pMod
ader
S Pins

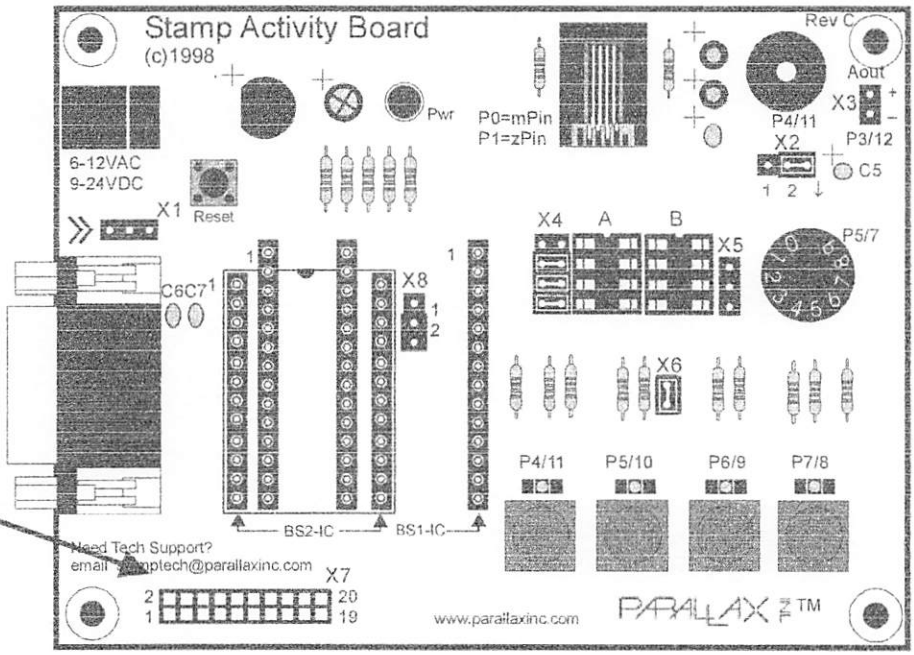


Figure 3: Basic Stamp Activity Board

S2P40 Demo Board

The BS2P40 Demo Board contains an AppMod header and supports a direct connection with the eb500 module. On the BS2P40 Demo Board, the AppMod header is labeled X1 (Figure 4). When inserting the eb500 module into the BS2P40 Demo Board AppMod header, assure that you insert Pin 1 of the eb500 module, marked with a white dot and a square, into the VSS pin of the AppMod header.

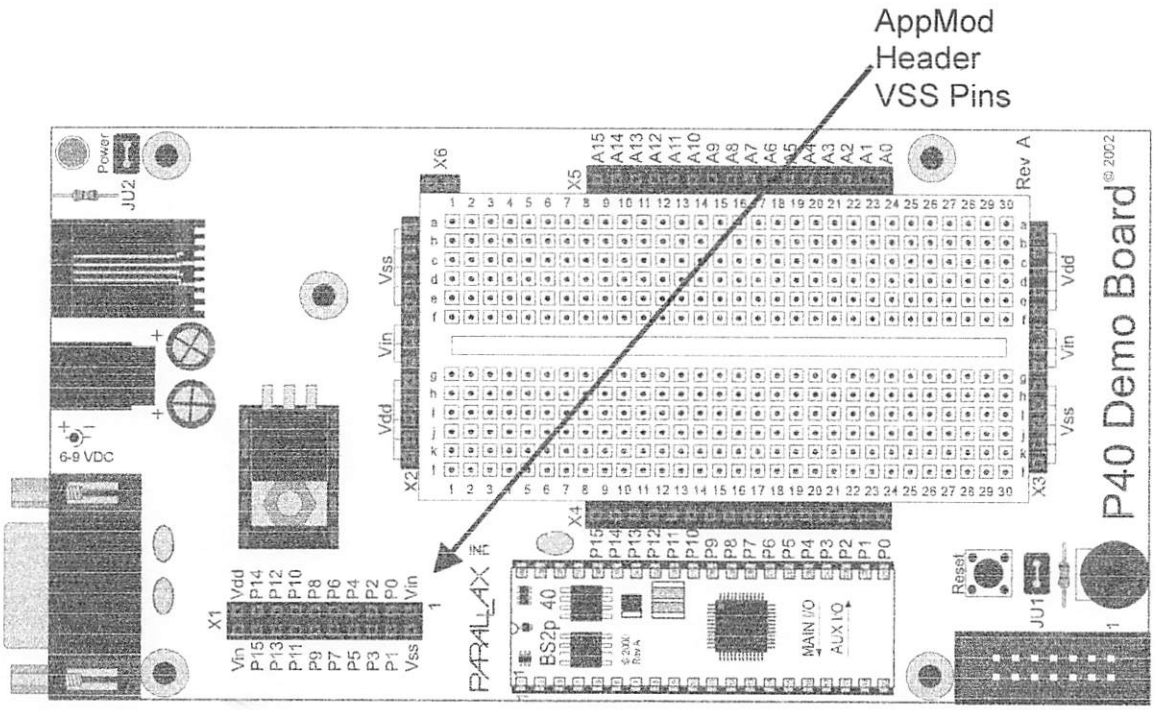


Figure 4: BS2P40 Demo Board

avelin Stamp Demo Board

The Javelin Stamp Demo Board contains an AppMod header and supports a direct connection with the eb500 module. On the Javelin Stamp Demo Board, the AppMod header is labeled X1 (Figure 5). When inserting the eb500 module into the Javelin Stamp Demo Board AppMod header, assure that you insert Pin 1 of the eb500 module, marked with a white dot and a square, into the VSS pin of the AppMod header.

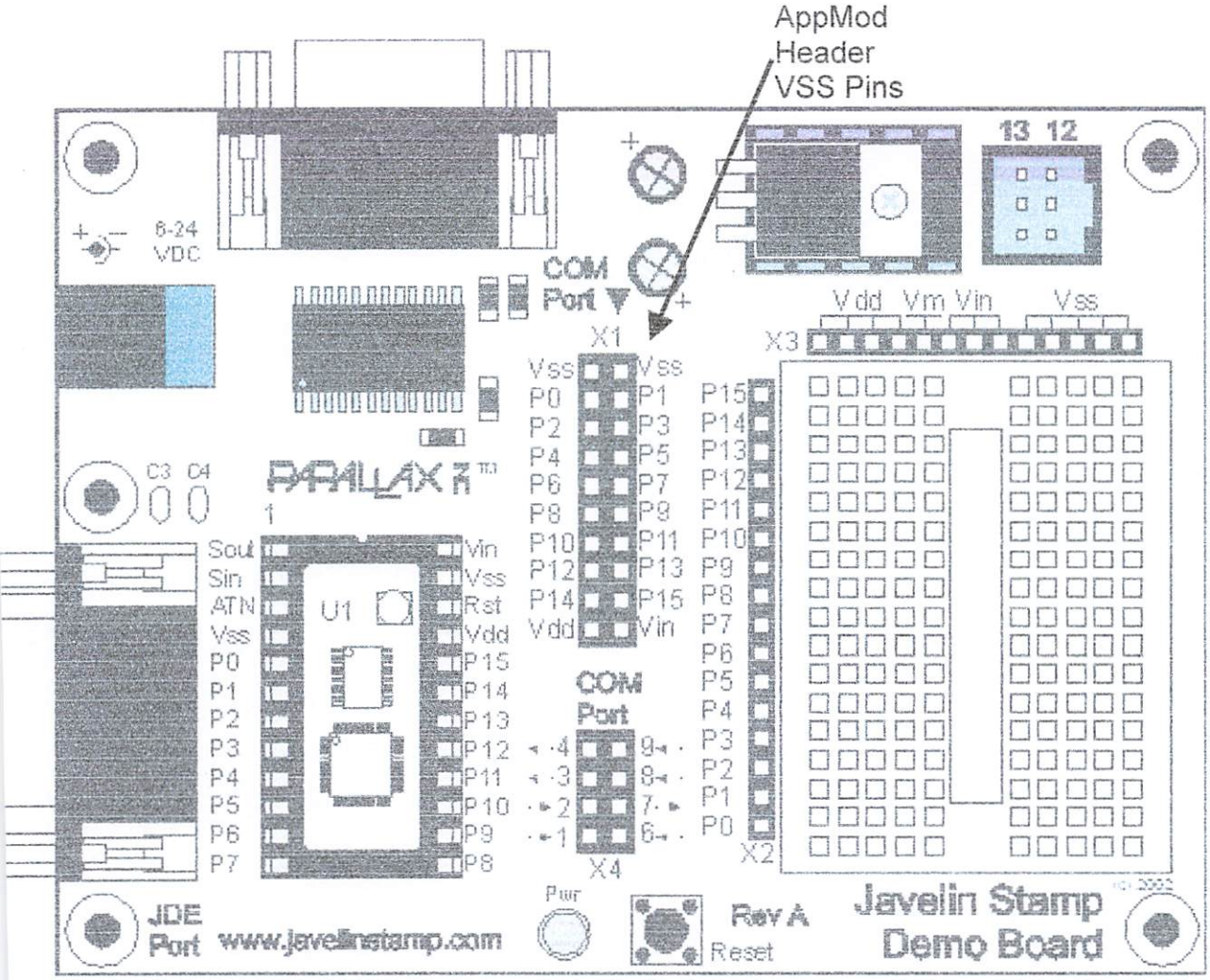


Figure 5: Javelin Stamp Demo Board

umoBoard

The SumoBoard contains an AppMod header and supports a direct connection with the eb500 module. On the SumoBoard, the AppMod header is labeled X10 (Figure 6). When inserting the eb500 module into the SumoBoard AppMod header, assure that you insert Pin 1 of the eb500 module, marked with a white dot and a square, into the VSS pin of the AppMod header.

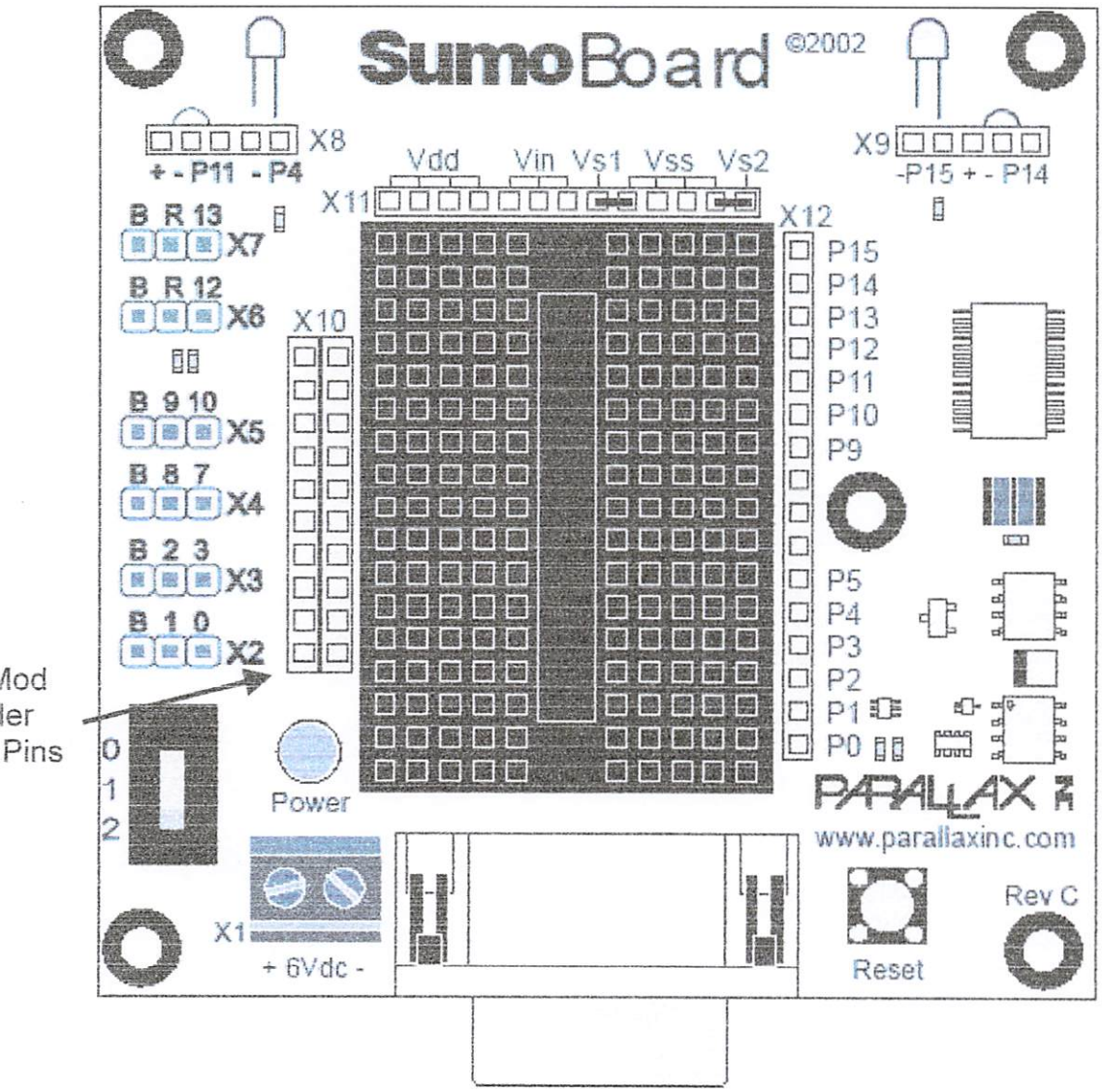


Figure 6: SumoBoard

uper Carrier Board

The Super Carrier Board contains an AppMod header and supports a direct connection with the eb500 module. On the Super Carrier Board, the AppMod header is labeled X1 (Figure 7). When inserting the eb500 module into the Super Carrier Board AppMod header, assure that you insert Pin 1 of the eb500 module, marked with a white dot and a square, into the VSS pin of the AppMod header.

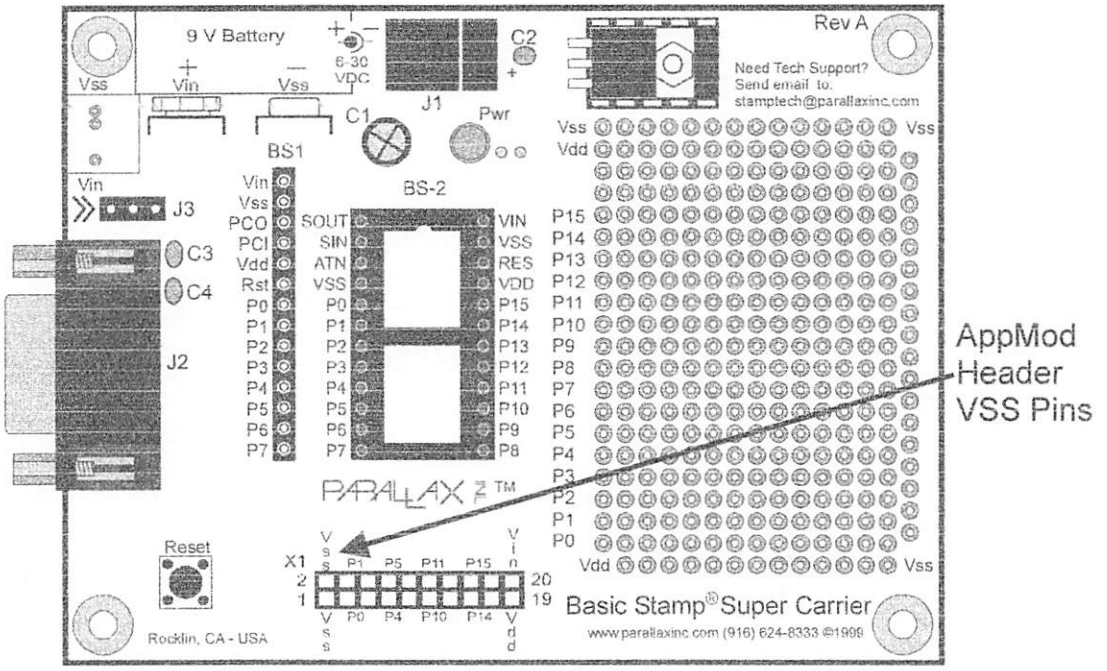


Figure 7: Super Carrier Board

Establishing a Connection

s section contains a number of exercises that demonstrate methods of establishing etooth wireless connections with the eb500. The scenarios described are not meant to n an exhaustive list, but rather illustrate a number of more common and useful ıfigurations. All source code shown in these exercises is available in electronic form on accompanying CD, in the Samples folder, using the filename used in this manual.

onnecting two eb500 Modules

his exercise we will step through the process of establishing a connection between an 500 inserted into a Board of Education board and an eb500 inserted into a SumoBoard ard.

perform this exercise, as documented, you will need a Board of Education board, a moBoard board, and two eb500 modules. If you are using any of the other supported rallax boards, you may need to make adjustments to this exercise.

ep 1: Write a BASIC Stamp Application to Get the eb500 Address

In this step we will write a BASIC Stamp application to interrogate an eb500 for its unique Bluetooth address.

1. Open the **Basic Stamp Editor**.
2. Enter the following program code into the editor.

```
'{$STAMP BS2}
szData VAR BYTE(20)
'Wait for the eb500 radio to be ready
PAUSE 1000
'Get the eb500 Bluetooth Address
SEROUT 1,84,["get addr",CR]
SERIN 0,84,[WAIT("ACK",CR)]
```

```
'Read the local address from the get command
SERIN 0,84,[STR szData\17]
SERIN 0,84,[WAIT(CR,">")]
szData(17) = 0
DEBUG STR szData\17,CR
```

The BASIC Stamp application issues an eb500 Get Address command and then reads and displays the response in the debug window. The response is the Bluetooth address of the local eb500 module.

3. On the **File** menu, click **Save As**.
4. In the **File name** box, enter a file name to which to save the program just created. For example, GetAddress.bs2.
5. Click **Save**.

Step 2: Insert the eb500 Modules into the BOE and SumoBoard boards

In this step we will insert the eb500 modules into the Board of Education (BOE) and SumoBoard boards.

1. Insert an **eb500** module into the **AppMod header** of the Board of Education board; assuring that Pin 1 of the eb500 module is inserted into the VSS pin of the AppMod header.
2. Insert an **eb500** module into the **AppMod header** of the SumoBoard board; assuring that Pin 1 of the eb500 module is inserted into the VSS pin of the AppMod header.

Step 3: Get the Address of the eb500 on the Board of Education Board

In this step we will get the Bluetooth address of the eb500 module on the Board of Education board. We will then use this address in the next step.

1. Connect the **Board of Education board serial port** to the **PC**.
2. Apply power to the Board of Education board.
3. On the **Run** menu, click **Run**.

   The Bluetooth address for the eb500 on the Board of Education board is shown in the debug window (Figure 8).
4. On the **Debug Terminal #1** dialog click **Close**.
5. Disconnect the **power** from the **Board of Education board**.
6. Disconnect the Board of Education board serial port from the PC.

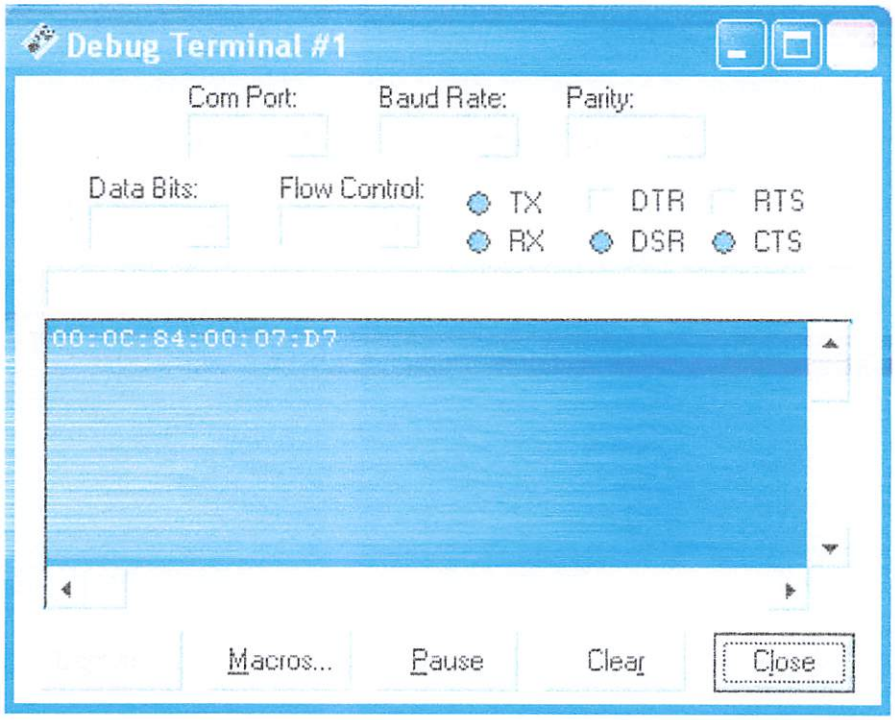


Figure 8: eb500 Bluetooth Address Output

ep 4: Connect the eb500 on the SumoBoard to the eb500 on the BOE

In this step we will develop and run a BASIC Stamp application on the SumoBoard to establish a connection with the Board of Education.

1. Using the BASIC Stamp Editor; on the **File** menu, click **New**.

   This will create a new project window within the BASIC Stamp Editor.

2. Enter the following program code into the editor, replacing the Bluetooth device address with the device address of the eb500 on the Board of Education board, which we obtained in the previous step.

```
'{$STAMP BS2}
'I/O Line 5 provides the connection status
INPUT 5
 'Wait for the eb500 radio to be ready
PAUSE 1000
'Connect to the remote device
SEROUT 1,84,["con 00:0C:84:00:07:D7",CR]
SERIN 0,84,[WAIT("ACK",CR)]
WaitForConnection:
```

```
      IF in5 = 0 THEN WaitForConnection
DEBUG "Connected",CR
'Wait for 20 seconds
PAUSE 20000
'I/O Line 6 allows us to switch to Command Mode
OUTPUT 6
'Switch to Command Mode
LOW 6
SERIN 0,84,[WAIT(CR,">")]
'Disconnect from the remote device
SEROUT 1,84,["dis",CR]
SERIN 0,84,[WAIT(CR,">")]
DEBUG "Disconnected",CR
```

The BASIC Stamp application establishes a connection with the remote Bluetooth device, waits twenty seconds, switches back to command mode and then disconnects from the remote device.

3. On the **File** menu, click **Save As**.
4. In the **File name** box, enter a file name to which to save the program just created. For example, Connect.bs2.
5. Click **Save**.
6. Apply **power** to the **Board of Education board**.
7. Apply **power** to the **SumoBoard board**.
8. On the **Run** menu, click **Run**.

   The Connection Status LED (Figure 1) on both eb500 modules will turn on when a connection is established between the two eb500 modules.
9. Disconnect the **power** from the **Board of Education board**.
10. Disconnect **power** from the **SumoBoard board**.

ɔnnecting a PC with an eb600 to a Board of Education

his exercise we will step through the process of establishing a connection between a PC t has an eb600 PC Adapter to an eb500 inserted into a Board of Education board.

perform this exercise, as documented, you will need an eb600 PC Adapter, a Board of ucation board and two eb500 modules. If you are using any of the other supported rallax boards, you may need to make adjustments to this exercise.

ep 1: eb600 PC Adapter Setup

In this step we will attach an eb500 module to the eb600 PC Adapter and apply power to the device.

1. Insert an **eb500** module into the **eb600 PC Adapter** header; assuring that Pin 1 of the eb500 module is inserted into Pin 1 of the header on the eb600 PC Adapter.
2. Connect the **eb600 PC Adapter** to a serial port on the PC using the **provided straight through serial cable**.

   The PC serial port must be available for HyperTerminal use.
3. Apply **power** to the **eb600 PC Adapter**.

ep 2: HyperTerminal Setup

In this step we will setup the Windows HyperTerminal application to establish a connection with the eb500 attached to the eb600 PC Adapter.

1. Open **HyperTerminal**.

   This will display the Connection Description dialog.
2. In the **Name** box, type the name of your connection. For example, EB600.
3. Click **OK**.

   This will display the Connect To dialog.
4. In the **Connect using** dropdown, select the **serial port** to which you have connected the eb600 PC Adapter.
5. Click **OK**.

   This will display the Properties dialog.
6. In the **Bits per second** dropdown, select **9600**.
7. In the **Data bits** dropdown, select **8**.
8. In the **Parity** dropdown, select **None**.
9. In the **Stop bits** dropdown, select **1**.

10. In the **Flow control** dropdown, select **None**.
11. Click **OK**.

    This will establish a connection with the serial port.
12. On the **Call** menu, click **Disconnect**.

    This will disconnect the connection just established, so that we can modify the connection properties in the following actions.
13. On the **File** menu, click **Properties**.

    This will display the Properties dialog.
14. On the **Settings** tab, click **ASCII Setup**.

    This will display the ASCII Setup dialog.
15. Check the **Send line ends with line feeds** checkbox.
16. Check the **Echo typed characters locally** checkbox.
17. Check the **Append line feeds to incoming line ends** checkbox.
18. Check the **Wrap lines that exceed terminal width** checkbox.
19. Click **OK**.

    This will return to the Properties dialog.
20. Click **OK**.
21. On the **Call** menu, click **Call**.

    This will establish a connection with the serial port.

ep 3: Board of Education – eb500 Setup

In this step we will attach an eb500 module to the Board of Education board and apply power to the device.

1. Insert an **eb500** module into the **AppMod header** of the Board of Education board; assuring that Pin 1 of the eb500 module is inserted into the VSS pin of the AppMod header.
2. Apply power to the **Board of Education board**.

   Power can be applied by attaching a 9 Volt battery, or the AC-Adapter provided by Parallax.

ep 4: Establish a Connection

In this step we will establish a connection between the PC and the Board of Education.

1. Using HyperTerminal, get the address of the **eb500** module that is connected to the Board of Education board by using the eb500 **LST** serial command.

   By issuing the LST command, the eb500 connected to the eb600 lists other Bluetooth devices that are in range and visible. Please note that this operation will take 30 seconds to complete.

   To obtain the address, type `lst` at the ">" prompt and press the return key.

   Example:

   ```
   >lst
   ACK
   00:0C:84:00:07:D7
   >
   ```

2. Using HyperTerminal, establish a connection with the **eb500** that is connected to the Board of Education board by using the eb500 **CON** serial command.

   To establish a connection, type `con` followed by a space, followed by the address returned in the previous action, followed by a carriage-return. The Connection Status LED (Figure 1) on both eb500 modules will turn on when a connection is established.

   Example:

   ```
   >con 00:0C:84:00:07:D7
   ACK
   >
   ```

3. Disconnect **power** from both the **eb500 PC Adapter** and the **Board of Education boards**.

   The removal of power resets the eb500 so that when power is restored the eb500 will boot into command mode.

onnecting a PC with a DBT-120 to a BOE

his exercise we will step through the process of establishing a connection from a PC that a D-Link® DBT-120 Bluetooth USB Adapter to an eb500 module inserted into a Board ducation (BOE) board.

perform this exercise, as documented, you will need a D-Link DBT-120, a Board of ication board, and an eb500 module. If you are using any of the other supported Parallax rds, you may need to make adjustments to this exercise.

the PC, the DBT-120 Bluetooth Software associates a COM port for establishing a nection from the PC to a remote Bluetooth device and a separate COM port for nections that are established from a remote Bluetooth device to the PC. This exercise nonstrates establishing a connection from the PC to a remote eb500. The next exercise demonstrate establishing a connection from a remote eb500 to the PC.

D-Link DBT-120 Bluetooth USB Adapter software must be fully installed prior to ablishing a connection. The PC settings shown in this exercise are based upon the ware provided with the D-Link DBT-120 Bluetooth USB Adapter.

p 1: DBT-120 Setup

In this step we will attach the DBT-120 USB Adapter to the PC. The software for the DBT-120 should already be setup.

1. Connect the **DBT-120** to an available **USB port** on the PC, following the instructions provided with the DBT-120 Bluetooth USB Adapter.

p 2: Board of Education – eb500 Setup

In this step we will attach an eb500 module to the Board of Education board and apply power to the device.

1. Insert an **eb500** module into the **AppMod connector** of the Board of Education board; assuring that Pin 1 of the eb500 module is inserted into the VSS pin of the AppMod header.
2. Apply **power** to the **Board of Education board**.

   Power can be applied by attaching a 9 Volt battery, or the AC-Adapter provided by Parallax.

ep 3: Establish a Connection Using the DBT-120 Bluetooth Software

In this step we will establish a connection from the PC to the eb500 module inserted into the Board of Education board.

The actions in this step need to be performed only once for the eb500. After performing the actions in this step, the connection details will be stored on the PC. Therefore, future connections can be established to an eb500 by simply opening the associated COM port.

1. Open **My Bluetooth Places** by double-clicking on the desktop icon.

   This will display the My Bluetooth Places dialog.

2. Click **View devices in range** to locate the **eb500 module** connected to the Board of Education.

   Provided the eb500 on the Board of Education is within range, eb500 will be shown in the window.

3. Select **eb500** and click **Discover services**.

   The A7 Serial Port service will be shown in the window.

4. Select **A7 Serial Port** and click **Connect to this service**.

   This will establish a connection from the PC to the eb500 on the Board of Education board and associate this connection with a specific COM port.

5. If the **A7 Serial Port** dialog is shown, click **OK**.

6. Select **A7 Serial Port** and click **Display service properties**.

   This will display the Bluetooth Properties dialog.

7. In the **Port** dropdown, which is disabled, please note the **COM port** shown.

   The DBT-120 Bluetooth software associates a specific COM port for a connection from the PC to an eb500. Applications, such as HyperTerminal, use this COM port to establish a connection and communicate with an eb500 from the PC. Remember, this COM port is used to establish a connection from the PC to the eb500. A different COM port is used when a connection is established from the eb500 to the PC.

8. Click **OK**.

9. Select **A7 Serial Port** and click **Disconnect from this service**.

   This will disconnect the connection to the eb500 on the Board of Education board.

ɜp 4: HyperTerminal Setup

In this step we will setup the Windows HyperTerminal application to establish a connection with the eb500 on the Board of Education board.

1. Open **HyperTerminal**.

   This will display the Connection Description dialog.

2. In the **Name** box, type the name of your connection. For example, eb500-BOE.

3. Click **OK**.

   This will display the Connect To dialog.

4. In the **Connect using** dropdown, select the **serial port** to which the DBT-120 Bluetooth software associated with the connection from the PC to the eb500 on the Board of Education board.

   The COM port associated with the connection was discovered in the previous step.

5. Click **OK**.

   This will display the Properties dialog.

6. In the **Bits per second** dropdown, select **9600**.

7. In the **Data bits** dropdown, select **8**.

8. In the **Parity** dropdown, select **None**.

9. In the **Stop bits** dropdown, select **1**.

10. In the **Flow control** dropdown, select **None**.

11. Click **OK**.

    This will establish a connection with the eb500 on the Board of Education board.

12. On the **Call** menu, click **Disconnect**.

    This will disconnect the connection just established, so that we can modify the connection properties in the following actions.

13. On the **File** menu, click **Properties**.

    This will display the Properties dialog.

14. On the **Settings** tab, click **ASCII Setup**.

    This will display the ASCII Setup dialog.

15. Check the **Send line ends with line feeds** checkbox.

16. Check the **Echo typed characters locally** checkbox.

17. Check the **Append line feeds to incoming line ends** checkbox.

18. Check the **Wrap lines that exceed terminal width** checkbox.
19. Click **OK**.

    This will return to the Properties dialog.
20. Click **OK**.

ep 5: Establish a Connection Using HyperTerminal

In this step we will establish a connection from the PC to the eb500 on the Board of Education, using HyperTerminal. This step relies on the connection information created previously in Step 3.

1. On the **Call** menu, click **Call**.

   This will establish a connection with the eb500 on the Board of Education board. The Connection Status LED (Figure 1) on the eb500 module will turn on when a connection is established.
2. On the **Call** menu, click **Disconnect**.

   This will close the connection with the eb500 on the Board of Education.

onnecting a BOE to a PC with a DBT-120

his exercise we will step through the process of establishing a connection from an eb500 dule inserted into a Board of Education (BOE) board to a PC that has a D-Link® DBT- Bluetooth USB Adapter.

perform this exercise, as documented, you will need a D-Link DBT-120, a Board of ication board, and an eb500 module. If you are using any of the other supported Parallax rds, you may need to make adjustments to this exercise.

the PC, the DBT-120 Bluetooth Software associates a COM port for establishing a nection from the PC to a remote Bluetooth device and a separate COM port for nections that are established from a remote Bluetooth device to the PC. This exercise nonstrates establishing a connection from a remote eb500 to the PC. When a remote etooth device establishes a connection with the PC, the connection is established with DBT-120 Bluetooth USB Adapter software running on the PC. To gain access to the a, an application, such as HyperTerminal, must open the COM port associated with the nection established from the remote device. In the Communications section, we will step ugh this process.

D-Link DBT-120 Bluetooth USB Adapter software must be fully installed prior to ablishing a connection. The PC settings shown in this exercise are based upon the ware provided with the D-Link DBT-120 Bluetooth USB Adapter.

p 1: DBT-120 Setup

In this step we will attach the DBT-120 USB Adapter to the PC. The software for the DBT-120 should already be setup.

1. Connect the **DBT-120** to an available **USB port** on the PC, following the instructions provided with the DBT-120 Bluetooth USB Adapter.

p 2: Obtain the Bluetooth Address of the PC

In this step we will obtain the Bluetooth address of the DBT-120 USB Adapter attached to the PC.

1. Open **My Bluetooth Places** by double-clicking on the desktop icon.

   This will display the My Bluetooth Places dialog.

2. Click **View or modify configuration**.

   This will display the Bluetooth Configuration dialog.

3. Select the **Hardware** tab and note the **Device Address** shown in the Device Properties section of the dialog.

   The device address will be used in the BASIC Stamp application developed in the next step.

4. Click **Cancel**

   This will close the Bluetooth Configuration dialog.

ep 3: Write a BASIC Stamp Application to Connect to the PC

In this step we will attach an eb500 module to the Board of Education board and develop a BASIC Stamp application to establish a connection with the PC.

1. Insert an **eb500** module into the **AppMod connector** of the Board of Education board; assuring that Pin 1 of the eb500 module is inserted into the VSS pin of the AppMod header.
2. Connect the **Board of Education board serial port** to the **PC**.
3. Open the **BASIC Stamp Editor**.
4. Enter the following **program code** into the editor, replacing the Bluetooth device address with the device address of the PC, which we obtained from the Hardware tab of the Device Properties section of the Bluetooth Configuration dialog in the previous step.

```
'{$STAMP BS2}
'I/O Line 5 provides the connection status
INPUT 5
'Wait for the eb500 radio to be ready
PAUSE 1000
'Connect to the remote device
SEROUT 1,84,["con 00:80:C8:35:2C:B8",CR]
SERIN 0,84,[WAIT("ACK",CR)]
WaitForConnection:
      IF in5 = 0 THEN WaitForConnection
DEBUG "Connected",CR
'Wait for 20 seconds
PAUSE 20000
'I/O Line 6 allows us to switch to Command Mode
OUTPUT 6
'Switch to Command Mode
LOW 6
SERIN 0,84,[WAIT(CR,">")]
'Disconnect from the remote device
SEROUT 1,84,["dis",CR]
```

```
SERIN 0,84,[WAIT(CR,">")]
DEBUG "Disconnected",CR
```

The BASIC Stamp application establishes a connection with the remote Bluetooth device, waits twenty seconds, switches back to command mode and then disconnects from the remote device.

5. On the **File** menu, click **Save As**.
6. In the **File name** box, enter a file name to which to save the program just created. For example, ConnectPC.bs2.
7. Click **Save**.

;p 4: Connect the eb500 on the Board of Education to the PC

1. Apply **power** to the **Board of Education board**.

   Power can be applied by attaching a 9 Volt battery, or the AC-Adapter provided by Parallax.

2. On the **Run** menu, click **Run**.

   The Connection Status LED (Figure 1) on the eb500 module will turn on when a connection is established. Additionally, on the My Bluetooth Places window, in the Additional Information column, the text "Connected" will be shown while a connection exists between the eb500 and the PC.

onnecting an iPAQ h1940 to a Board of Education

his exercise we will step through the process of establishing a connection from an iPAQ 940, which has integrated Bluetooth, to an eb500 module inserted into a Board of ucation board.

perform this exercise, as documented, you will need an iPAQ h1940, a Board of ucation board, and an eb500 module. If you are using a different model of Pocket PC, n integrated Bluetooth, or any of the other supported Parallax boards, you may need to ke adjustments to this exercise.

ep 1: Board of Education – eb500 Setup

In this step we will attach an eb500 module to the Board of Education board and apply power to the device.

1. Insert an **eb500** module into the **AppMod connector** of the Board of Education board; assuring that Pin 1 of the eb500 module is inserted into the VSS pin of the AppMod header.
2. Apply **power** to the **Board of Education board**.

   Power can be applied by attaching a 9 Volt battery, or the AC-Adapter provided by Parallax.

ep 2: iPAQ h1940 Setup

In this step we will setup the iPAQ for connecting to the eb500.

1. Tap the **Bluetooth icon** in the system tray on the Today screen and select **Bluetooth Manager**.

   This will display the Bluetooth Manager dialog.
2. On the **New** menu, select **Connect**.

   This will display the first page of the Connection Wizard.
3. Select **Explore a Bluetooth device** and tap **Next**.

   This will display the next page of the Connection Wizard.
4. Tap in the **Device** box.

   This will display the Connection Wizard Bluetooth Browser dialog containing a list of found devices.
5. Tap **eb500**.

   This will display the next page of the Connection Wizard.
6. In the **Service Selection** box, select **A7 Serial Port**.

7. Tap **Next**.

   This will create a shortcut for the service.

8. Tap **Finish**.

   This will display the Bluetooth Manager dialog with the shortcut created in the window, eb500: A7 Serial Port.

ep 3: Establish a Connection

In this step we will establish a connection from the iPAQ to the Board of Education.

1. Tap-and-hold the shortcut created in the previous step, **eb500: A7 Serial Port**.
2. Select **Connect**.

   This will establish a connection with the eb500 on the Board of Education board. The Connection Status LED (Figure 1) on the eb500 module will turn on when a connection is established.

3. Tap **Active Connections**.

   This will display the Bluetooth Manager Active Connections page showing the status of your active Bluetooth connections.

4. Tap **My Shortcuts**.
5. Tap-and-hold the shortcut created in the previous step, **eb500: A7 Serial Port**.
6. Select **Disconnect**.

   This will close the connection with the eb500 on the Board of Education board. The Connection Status LED on the eb500 module will turn off.

onnecting a Board of Education to an iPAQ h1940

his exercise we will step through the process of establishing a connection from an eb500 dule inserted into a Board of Education board to an iPAQ h1940, which has integrated etooth.

perform this exercise, as documented, you will need an iPAQ h1940, a Board of ucation board, and an eb500 module. If you are using a different model of Pocket PC, n integrated Bluetooth, or any of the other supported Parallax boards, you may need to ke adjustments to this exercise.

ep 1: Obtain the Bluetooth Address of the iPAQ

In this step we will obtain the Bluetooth address of the iPAQ.

1. Tap the **Bluetooth icon** in the system tray on the Today screen and select **Bluetooth Settings**.

   This will display the Settings dialog.

2. Tap the **Accessibility** tab and note the **Address** shown in the Device Identification section of the dialog.

   The device address will be used in the BASIC Stamp application developed in the next step.

3. Tap **OK** to close the dialog.

ep 2: Write a BASIC Stamp Application to Connect to the iPAQ

In this step we will attach an eb500 module to the Board of Education board and develop a BASIC Stamp application to establish a connection with the iPAQ.

1. Insert an **eb500** module into the **AppMod connector** of the Board of Education board; assuring that Pin 1 of the eb500 module is inserted into the VSS pin of the AppMod header.
2. Connect the **Board of Education board serial port** to the **PC**.
3. Open the **BASIC Stamp Editor**.
4. Enter the following **program code** into the editor, replacing the Bluetooth device address with the device address of the iPAQ, which we obtained from the iPAQ in the previous step.

```
'{$STAMP BS2}
'I/O Line 5 provides the connection status
INPUT 5
'Wait for the eb500 radio to be ready
```

```
PAUSE 1000
'Connect to the remote device
SEROUT 1,84,["con 00:04:3E:62:FE:01",CR]
SERIN 0,84,[WAIT("ACK",CR)]
WaitForConnection:
      IF in5 = 0 THEN WaitForConnection
DEBUG "Connected",CR
'Wait for 20 seconds
PAUSE 20000
'If there is no connection just exit
IF in5 = 0 THEN Exit
'I/O Line 6 allows us to switch to Command Mode
OUTPUT 6
'Switch to Command Mode
LOW 6
SERIN 0,84,[WAIT(CR,">")]
'Disconnect from the remote device
SEROUT 1,84,["dis",CR]
SERIN 0,84,[WAIT(CR,">")]
Exit:
      DEBUG "Disconnected",CR
```

The BASIC Stamp application establishes a connection with the remote Bluetooth device, waits twenty seconds, switches back to command mode and then disconnects from the remote device.

On the h1940 model of the iPAQ, the Bluetooth software closes the connection after a short period of time if there is not an application running on the iPAQ to receive the data over the connection. Therefore, after the twenty second wait, the BASIC Stamp application checks if there is still a valid connection before switching to Command Mode. If there is no connection, the eb500 is already in Command Mode.

5. On the **File** menu, click **Save As**.
6. In the **File name** box, enter a file name to which to save the program just created. For example, ConnectPPC.bs2.
7. Click **Save**.

ep 3: Establish a Connection

In this step we will establish a connection from the Board of Education to the iPAQ.

1. Turn on the **iPAQ**.
2. Tap the **Bluetooth icon** and select **Bluetooth Manager**.

   This will display the Bluetooth Manager dialog.
3. Tap the **Active Connections** tab.
4. Apply **power** to the **Board of Education board**.

   Power can be applied by attaching a 9 Volt battery, or the AC-Adapter provided by Parallax.
5. Using the Basic Stamp Editor, on the **Run** menu, click **Run**.

   Depending on your current iPAQ Bluetooth configuration, the Authorization Requested Dialog may appear (Figure 9). If this dialog appears, tap Accept to accept the connection. The Connection Status LED (Figure 1) on the eb500 module will turn on when a connection is established. On the iPAQ the connection will be shown in the Incoming Connections section of the Active Connections tab on the Bluetooth Active Connections dialog.

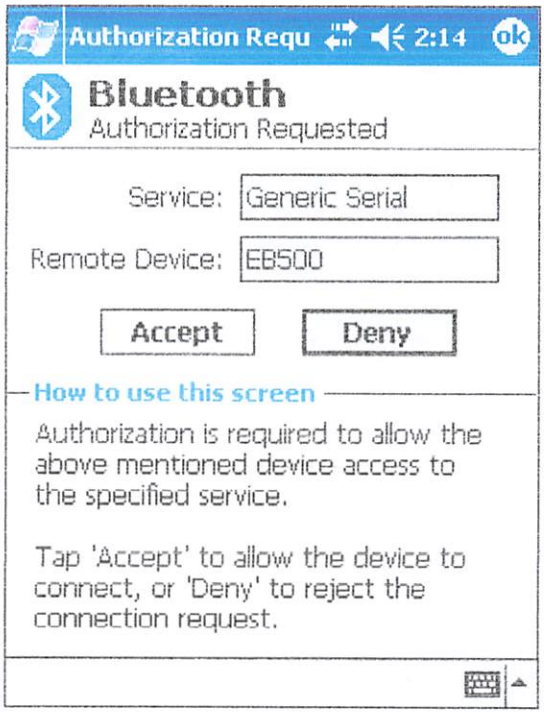


Figure 9: iPAQ Bluetooth Authorization Request Dialog

Communications

s section contains a number of exercises that demonstrate methods of communicating
ır a Bluetooth wireless connection with the eb500. The scenarios described are not
ant to form an exhaustive list, but rather illustrate a number of more common and useful
ıfigurations. All source code shown in these exercises are available in electronic form on
accompanying CD, in the Samples folder, using the filename used in this manual.

ɔmmunicating between Two eb500 Modules

this exercise we will step through the process of communicating between two eb500
dules, one inserted into a Boe-Bot robot and the other inserted into a SumoBot robot. We
program the SumoBot to use its infrared sensors to follow an object and then transmit its
vements to the Boe-Bot. The Boe-Bot will use the received information to mimic the
vements of the SumoBot.

perform this exercise, as documented, you will need a Boe-Bot, a SumoBot, and two
00 modules. If you are using any of the other supported Parallax robots, you may need
nake adjustments to this exercise.

ıp 1: Create a Monkey-See Application for the SumoBot

In this step we will create a BASIC Stamp application that will use the infrared sensors of the SumoBot to follow an object and transmit its movements to a remote eb500.

1. Open the **BASIC Stamp Editor**.
2. Enter the following **program code** into the editor, replacing the Bluetooth device address with the device address of the eb500 inserted into the Boe-Bot robot.

```
'{$STAMP BS2}
'I/O Line 5 provides the connection status
INPUT 5
'-----[I/O Definitions]-------------------------------------
```

```
LMotor      CON   13
RMotor      CON   12
LfIrOut     CON   4
LfIrIn      VAR   In11
RtIrOut     CON   15
RtIrIn      VAR   In14

'-----[Constants]-------------------------------------------
LFwdFast    CON   1000
LRevFast    CON   500
RFwdFast    CON   500
RRevFast    CON   1000

'-----[Variables]-------------------------------------------
irBits      VAR   NIB
irLeft      VAR   irBits.Bit1
irRight     VAR   irBits.Bit0
lastIr      VAR   NIB
bBuffer     VAR   BYTE(4)
bErrorCode  VAR   BYTE

'-----[Initialization]--------------------------------------
'Wait for the eb500 radio to be ready
PAUSE 1000

Connect:
      'Connect to Monkey-Do
      SEROUT 1,84,["con 00:0C:84:00:07:D7",CR]
      SERIN 0,84,[WAIT("ACK",CR)]
      'Either an Err #<CR> or a ">" will be received
      SERIN 0,84,[STR bBuffer\6\">"]
      IF bBuffer(0) = "E" THEN ErrorCode

WaitForConnection:
      IF in5 = 0 THEN WaitForConnection
```

```
'-----[Main Code]-----------------------------------------------
Main:
      'Verify the connection is still up before each loop
      IF in5 = 0 THEN Connect
      GOSUB Read_IR_Sensors
      BRANCH irBits,[Hold, Turn_Right, Turn_left, Move_Fwd]

Move_Fwd:
      SEROUT 1,84,["3"]
      PULSOUT LMotor,LFwdFast
      PULSOUT RMotor,RFwdFast
      GOTO Main

Turn_Right:
      SEROUT 1,84,["1"]
      PULSOUT LMotor,LFwdFast
      PULSOUT RMotor,RRevFast
      GOTO Main

Turn_Left:
      SEROUT 1,84,["2"]
      PULSOUT LMotor,LRevFast
      PULSOUT RMotor,RFwdFast
      GOTO Main

Hold:
      GOTO Main

'-----[Subroutines]----------------------------------------------
Read_IR_Sensors:
      FREQOUT LfIrOut,1,38500
      irLeft = ~LfIrIn
      FREQOUT RtIrOut,1,38500
      irRight = ~RtIrIn
```

```
        RETURN

BadCommand:
        DEBUG "A bad command was received."
        END

ErrorCode:
        bErrorCode = bBuffer(4)
        DEBUG "An error was received: ",STR bErrorCode,CR
        END
```

3. On the **File** menu, click **Save As**.
4. In the **File name** box, enter a file name to which to save the program just created. For example, MonkeySee.bs2.
5. Click **Save**.

ɜp 2: Create a Monkey-Do Application for the Boe-Bot

In this step we will create a BASIC Stamp application that will receive information from the remote SumoBot and perform movements based on that information.

1. On the **File** menu, click **New**.

   This will create a new project window within the BASIC Stamp Editor.
2. Enter the following **program code** into the editor.

```
'{$STAMP BS2}
'-----[I/O Definitions]-------------------------------------
LMotor      CON   15
RMotor      CON   14

'-----[Constants]-------------------------------------------
LFwdFast    CON   1000
LRevFast    CON   500
RFwdFast    CON   500
RRevFast    CON   1000

'-----[Variables]-------------------------------------------
CmdData     VAR   BYTE
```

```
'-----[Initialization]---------------------------------------
Initialize:
      'Wait for the eb500 radio to be ready
      PAUSE 1000
      'Set the initial state to hold
      CmdData = 3

'-----[Main Code]---------------------------------------------
Main:
      'Wait for a command
      SERIN 0,84,[DEC1 CmdData]

      'Process the command
      BRANCH CmdData,[Hold, Turn_Right, Turn_left, Move_Fwd]

      'If the command was invalid just loop again
      GOTO Main

Move_Fwd:
      PULSOUT LMotor,LFwdFast
      PULSOUT RMotor,RFwdFast
      GOTO Main

Turn_Right:
      PULSOUT LMotor,LFwdFast
      PULSOUT RMotor,RRevFast
      GOTO Main

Turn_Left:
      PULSOUT LMotor,LRevFast
      PULSOUT RMotor,RFwdFast
      GOTO Main

Hold:
```

```
GOTO Main
```

3. On the **File** menu, click **Save As**.
4. In the **File name** box, enter a file name to which to save the program just created. For example, MonkeyDo.bs2.
5. Click **Save**.

ep 3: Download the Applications to the Robots

In this step we will download the applications we just created to the respective robots.

1. Click the **MonkeySee.bs2** tab in the BASIC Stamp Editor.
2. Connect the **SumoBoard board serial port** to the **PC**.
3. Apply **power** to the **SumoBoard board**.
4. On the **Run** menu, click **Run**.
5. On the **Debug Terminal #1** dialog click **Close**.
6. Disconnect the **power** from the **SumoBoard board**.
7. Disconnect the **SumoBoard board serial port** from the **PC**.
8. Click the **MonkeyDo.bs2 tab** in the BASIC Stamp Editor.
9. Connect the **Board of Education board serial port** to the **PC**.
10. Apply **power** to the **Board of Education board**.
11. On the **Run** menu, click **Run**.
12. Disconnect the **Board of Education board serial port** from the **PC**.

ep 4: Run the Monkey-See / Monkey-Do Applications

In this step we will run the Monkey-See / Monkey-Do applications.

1. Apply **power** to the **SumoBoard board**.
2. Make the **Boe-Bot** robot **mimic the movements** of the SumoBot by putting your hand in front of the SumoBot IR sensors.

   As you move your hand left, right and forward, the SumoBot will follow your hand and the Boe-Bot will mimic the same movements.

ommunicating between a PC with an eb600 and a BOE

this exercise we will step through the process of communicating between an eb500 idule inserted into a Board of Education (BOE) board and a PC that has an eb600 PC apter.

perform this exercise, as documented, you will need to have two serial ports available on ur PC, an eb600 PC Adapter, a Board of Education board, and two eb500 moduels. One ial port will be used to connect the PC to the Board of Education serial port. The other ial port will be used to connect to the eb600 PC Adapter. If you are using any of the other pported Parallax boards, you may need to make adjustments to this exercise.

ep 1: Transmit Data from the PC to the BASIC Stamp

In this step we will create a BASIC Stamp application to read data from the eb500 and display the data in the BASIC Stamp Editor Debug window. We will then download and run the application.

1. Connect the **Board of Education board serial port** to the **PC**.
2. Open the **BASIC Stamp Editor**.
3. Enter the following **program code** into the editor.

```
'{$STAMP BS2}
bData VAR BYTE
'Wait for the eb500 radio to be ready
PAUSE 1000
Main:
      SERIN 0,84,[STR bData\1]
      DEBUG STR bData\1
      GOTO Main
```

The application waits for an individual byte of data to arrive and then displays the byte in the debug window and then repeats this process.

4. On the **File** menu, click **Save As**.
5. In the **File name** box, enter a file name to which to save the program just created. For example, Receive.bs2.
6. Click **Save**.
7. Apply **power** to the **Board of Education board**.
8. On the **Run** menu, click **Run**.

9. Establish a connection from the **PC** to the **Board of Education**.

   Please see the section titled Connecting a PC with an eb600 to a Board of Education for information on establishing the connection.

10. Using HyperTerminal, type a series of **characters**.

    These characters will be transmitted over the wireless link, read by the BASIC Stamp application, and then displayed in the debug window (Figure 10).

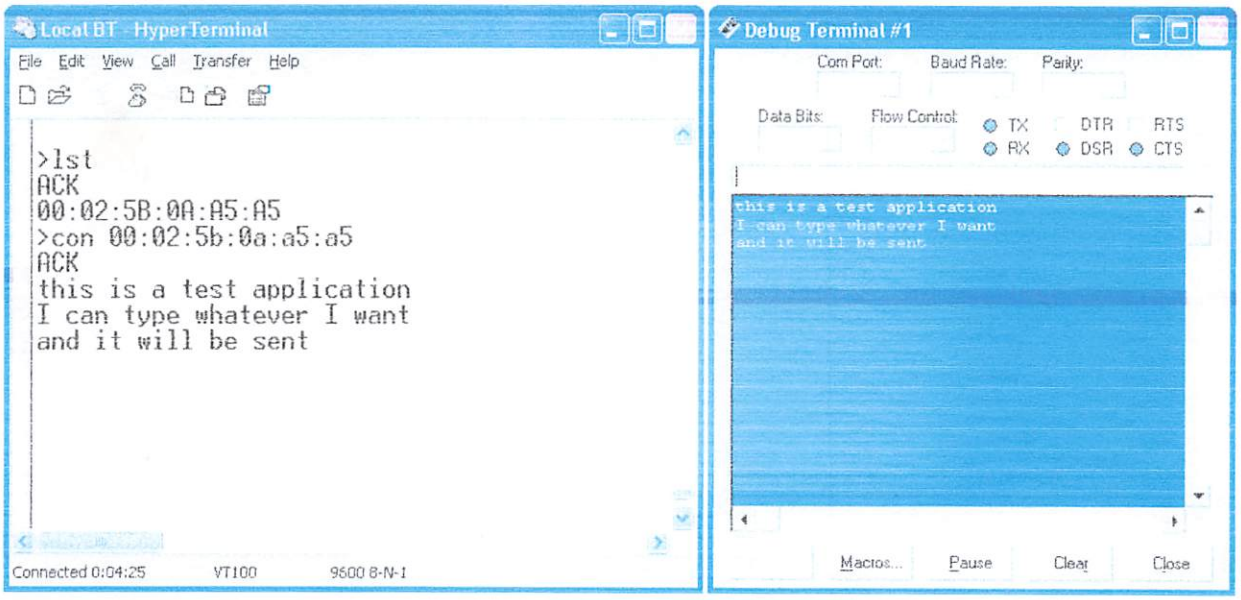


Figure 10: HyperTerminal Input and Debug Output

ep 2: Transmit Data from the BASIC Stamp to the PC

In this step we will create a BASIC Stamp application to send data out the eb500 to the PC where we will use HyperTerminal to display the data received by the eb500 module attached to the eb600 on the PC.

1. Reset the **eb500** attached to the **eb600 PC Adapter** to place the eb500 into command mode.

   To reset the eb500 attached to the eb600 PC Adapter, disconnect the power, wait a couple of seconds, and then reconnect the power.

2. Reset the **eb500** attached to the **Board of Education board** to place the eb500 into command mode.

   To reset the eb500 attached to the Board of Education board, disconnect the power, wait a couple of seconds, and then reconnect the power. The Reset push button on the Board of Education board will NOT reset the eb500.

3. Using HyperTerminal, acquire the **device address** of the **eb500** connected to the eb600 PC Adapter by using the eb500 GET ADDR serial command.

   Please note the device address as it will be used in the BASIC Stamp application developed in the following actions.

   By issuing the GET ADDR command, the eb500 connected to the eb600 will return its own device address.

   To obtain the device address, type `get addr` at the ">" prompt and press the return key.

   Example:

   ```
   >get addr
   ACK
   00:0C:84:00:07:D8
   >
   ```

4. Using the BASIC Stamp Editor, on the **File** menu, click **New**.

   This will create a new project window within the BASIC Stamp Editor.

5. Enter the following **program code** into the editor, replacing the device address with the device address obtained from the GET ADDR command issued above.

```
'{$STAMP BS2}
nCount VAR BYTE
'I/O Line 5 provides the connection status
INPUT 5
'Wait for the eb500 radio to be ready
PAUSE 1000
'Connect to the remote device
SEROUT 1,84,["con 00:0C:84:00:07:D8",CR]
SERIN 0,84,[WAIT("ACK",CR)]
WaitForConnection:
      IF in5 = 0 THEN WaitForConnection
DEBUG "Connected",CR
FOR nCount = 1 to 10
      SEROUT 1,84,["Hello World",CR]      'sending data
      PAUSE 1000                          'wait for 1 second
NEXT
'I/O Line 6 allows us to switch to command mode.
```

```
OUTPUT 6
'Switch to Command Mode
LOW 6
SERIN 0,84,[WAIT(CR,">")]
'Disconnect from the remote device
SEROUT 1,84,["dis",CR]
SERIN 0,84,[wait(CR,">")]
DEBUG "Disconnected",CR
```

The application establishes a connection with the remote eb500 device, transmits "Hello World" ten times, switches back to command mode and then disconnects from the remote device.

The first call to SEROUT is used when the eb500 is in command mode and instructs the eb500 to establish a connection with the device specified. Once a connection is established the eb500 is in data mode, which causes further calls to SEROUT to be sent to the remote device.

6. On the **File** menu, click **Save As**.

   This will display the Save As dialog.

7. In the **File name** box, enter a file name to which to save the program just created. For example, HelloWorld.bs2.

8. Click **Save**.

9. On the **Run** menu, click **Run**.

   This will display the Download Program dialog while downloading the program to the BASIC Stamp. After the download is complete the BASIC Stamp application will transmit "Hello World" over the wireless link, and HyperTerminal will display the received data (Figure 11).

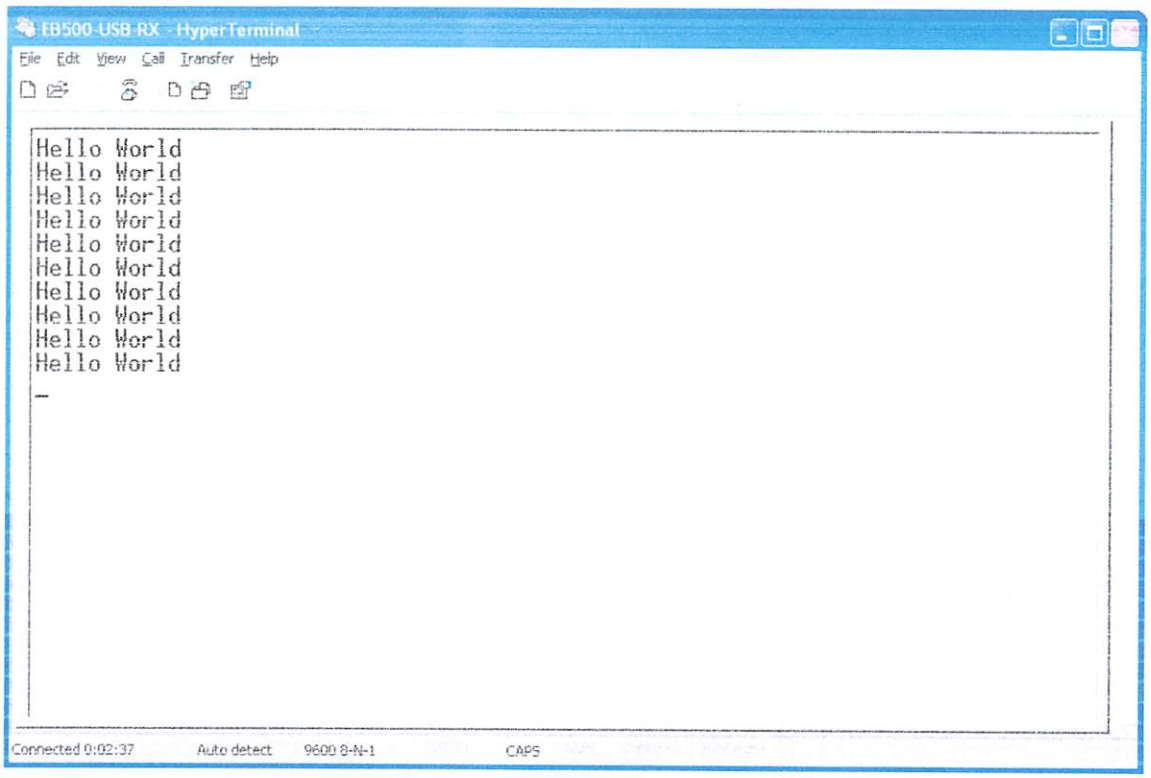


Figure 11: HyperTerminal Output - Hello World

ɔmmunicating between a PC with a DBT-120 and a BOE

:his exercise we will step through the process of communicating between a PC that has a _ink® DBT-120 Bluetooth USP Adapter and an eb500 module inserted into a Board of ucation board.

perform this exercise, as documented, you will need a D-Link DBT-120, a Board of ucation board, and an eb500 module. If you are using any of the other supported Parallax ards, you may need to make adjustments to this exercise.

the PC, the DBT-120 Bluetooth Software associates a COM port for establishing a ınection from the PC to a remote Bluetooth device and a separate COM port for ınections that are established from a remote Bluetooth device to the PC. Step 1 of this ercise demonstrates establishing a connection from the PC to the eb500 on the Board of ucation and then transmitting data over the connection. Step 2 of this exercise monstrates establishing a connection from the eb500 on the Board of Education to the PC l then transmitting data over the connection.

e D-Link DBT-120 Bluetooth USB Adapter software must be fully installed prior to ablishing a connection. The PC settings shown in this exercise are based upon the tware provided with the D-Link DBT-120 Bluetooth USB Adapter.

ep 1: Transmit Data from the PC to the BASIC Stamp

In this step we will create a BASIC Stamp application to read data from the eb500 and display the data in the BASIC Stamp Editor Debug window. We will then download and run the application.

1. Connect the **Board of Education board serial port** to the **PC**.
2. Open the **BASIC Stamp Editor**.
3. Enter the following **program code** into the editor.

```
'{$STAMP BS2}
bData VAR BYTE
'Wait for the eb500 radio to be ready
PAUSE 1000
Main:
      SERIN 0,84,[STR bData\1]
      DEBUG STR bData\1
      GOTO Main
```

The application waits for an individual byte of data to arrive and then displays the byte in the debug window and then repeats this process.

4. On the **File** menu, click **Save As**.

5. In the **File name** box, enter a file name to which to save the program just created. For example, Receive.bs2.
6. Click **Save**.
7. Apply **power** to the **Board of Education board**.
8. On the **Run** menu, click **Run**.

   This will display the Download Progress dialog while downloading the program to the BASIC Stamp. After the download is complete, the Debug Terminal #1 dialog will be shown.
9. Establish a connection from the **PC** to the **Board of Education**.

   Please see the section titled Connecting a PC with a DBT-120 to a BOE for information on establishing a connection.
10. Using HyperTerminal, type a series of **characters**.

    These characters will be transmitted over the wireless link, read by the BASIC Stamp application, and then displayed in the debug window.

ep 2: Transmit Data from the BASIC Stamp to the PC

In this step we will create a BASIC Stamp application to send data out the eb500 to the PC where we will use HyperTerminal to display the data received by the DBT-120 Bluetooth USB Adapter.

1. Reset the **eb500** attached to the **Board of Education board** to place the eb500 into command mode.

   To reset the eb500 attached to the Board of Education board, disconnect the power, wait a couple of seconds, and then reconnect the power. The Reset push button on the Board of Education board will NOT reset the eb500.
2. Close **HyperTerminal**.
3. Close the **BASIC Stamp Editor Debug** dialog.
4. Open **My Bluetooth Places** by double-clicking on the desktop icon.

   This will display the My Bluetooth Places dialog.
5. Click **View or modify configuration**.

   This will display the Bluetooth Configuration dialog.

6. Select the **Local Services** tab and note the **COM Port** for the Bluetooth Serial Port service. You may have to scroll to the right to see the COM Port column of the table.

   This COM port is the serial communications port that the DBT-120 Bluetooth software has associated for connections that are established from a remote Bluetooth device. This COM port can be used to communicate with the eb500 from applications, such as HyperTerminal, when connections are established from remote Bluetooth devices to the PC.

7. Select the **Hardware** tab and note the **Device Address** shown in the Device Properties section of the dialog.

   The device address will be used in the BASIC Stamp application developed in later actions.

8. On the **Bluetooth Configuration** dialog, click **Cancel**.

   This will close the Bluetooth Configuration dialog.

9. Close the **My Bluetooth Places** window.

10. Open **HyperTerminal**.

    This will display the Connection Description dialog.

11. In the **Name** box, type the name of your connection. For example, EB500-USB-RX.

12. Click **OK**.

    This will display the Connect To dialog.

13. In the **Connect using** dropdown, select the **serial port** to which the DBT-120 Bluetooth software associated with the connection from the eb500 on the Board of Education board to the PC.

    This is the COM port that we previously noted as being the COM port that is used to communicate with the eb500 when connections are established from remote Bluetooth devices to the PC.

14. Click **OK**.

    This will display the Properties dialog.

15. In the **Bits per second** dropdown, select **9600**.

16. In the **Data bits** dropdown, select **8**.

17. In the **Parity** dropdown, select **None**.

18. In the **Stop bits** dropdown, select **1**.

19. In the **Flow control** dropdown, select **None**.

20. Click **OK**.

    This will establish a connection with the DBT-120 Bluetooth USB Adapter software. This does NOT establish a connection with the remote Bluetooth device. The remote Bluetooth device establishes a connection with the DBT-120 Bluetooth USB Adapter software and applications, such as HyperTerminal, establish a connection to the DBT-120 Bluetooth USB Adapter software using the COM port that is used when connections are established from remote Bluetooth devices to the PC to gain access to the data being transmitted from the remote device.

21. On the **Call** menu, click **Disconnect**.

    This will disconnect the connection just established, so that we can modify the connection properties in the following actions.

22. On the **File** menu, click **Properties**.

    This will display the Properties dialog.

23. On the **Settings** tab, click **ASCII Setup**.

    This will display the ASCII Setup dialog.

24. Check the **Send line ends with line feeds** checkbox.

25. Check the **Echo typed characters locally** checkbox.

26. Check the **Append line feeds to incoming line ends** checkbox.

27. Check the **Wrap lines that exceed terminal width** checkbox.

28. Click **OK**.

    This will return to the Properties dialog.

29. Click **OK**.

30. On the **Call** menu, click **Call**.

    This will establish a connection with the DBT-120 Bluetooth USB Adapter Software.

31. Using the BASIC Stamp Editor, on the **File** menu, click **New**.

    This will create a new project window within the BASCIC Stamp Editor.

32. Enter the following **program code** into the editor, replacing the device Bluetooth address with the device address obtained from the Hardware tab of the Device Properties section of the Bluetooth Configuration dialog on the PC.

```
'{$STAMP BS2}
nCount VAR BYTE
'I/O Line 5 provides the connection status
INPUT 5
```

```
'Wait for the eb500 radio to be ready
PAUSE 1000
'Connect to the remote device
SEROUT 1,84,["con 00:80:C8:35:2C:B8",CR]
SERIN 0,84,[WAIT("ACK",CR)]
WaitForConnection:
      IF in5 = 0 THEN WaitForConnection
DEBUG "Connected",CR
FOR nCount = 1 to 10
      SEROUT 1,84,["Hello World",13]      'sending data
      PAUSE 1000                          'wait for 1 second
NEXT
'I/O Line 6 allows us to switch to command mode
OUTPUT 6
'Switch to Command Mode
LOW 6
SERIN 0,84,[WAIT(CR,">")]
'Disconnect from the remote device
SEROUT 1,84,["dis",CR]
SERIN 0,84,[WAIT(CR, ">")]
DEBUG "Disconnected",CR
```

The BASIC Stamp application establishes a connection with the remote eb500 device, waits for the connection to be established, then transmits "Hello World" ten times, switches back to command mode, and then disconnects from the remote device.

The first call to SEROUT is used when the eb500 is in command mode and instructs the eb500 to establish a connection with the device specified. Once a connection is established the eb500 is in data mode, which causes further calls to SEROUT to be sent to the remote device.

33. On the **File** menu, click **Save As**.

This will display the Save As dialog.

34. In the **File name** box, enter a file name to which to save the program just created. For example, HelloWorld.bs2.

35. Click **Save**.

36. On the **Run** menu, click **Run**.

This will display the Download Program dialog while downloading the program to the BASIC Stamp. After the download is complete the BASIC Stamp application will transmit "Hello World" over the wireless link, and HyperTerminal will display the received data (Figure 11).

Communicating between an iPAQ h1940 an a BOE

n this exercise we will step through the process of communicating between an iPAQ h1940, which has integrated Bluetooth, and an eb500 module inserted into a Board of Education oard.

o perform this exercise, as documented, you will need an iPAQ h1940, a Board of ducation board, and an eb500 module. If you are using a different model of Pocket PC, ith integrated Bluetooth, or any of the other supported Parallax boards, you may need to ake adjustments to this exercise.

tep 1: Transmit Data from the iPAQ to the BASIC Stamp

In this step we will create a BASIC Stamp application to read data from the eb500 and display the data in the BASIC Stamp Editor Debug window. We will then download and run the application. The application for the iPAQ is too verbose to include in this manual; therefore, the application, along with the source code, is available on the accompanying CD in the Samples folder. To modify the Pocket PC application you will need eMbedded Visual C++ 4.0 with Service Pack 2 and the SDK for Windows Mobile™ 2003-based Pocket PCs.

1. Connect the **Board of Education board serial port** to the **PC**.
2. Open the **BASIC Stamp Editor**.
3. Enter the following **program code** into the editor.

```
'{$STAMP BS2}
szData VAR BYTE(20)
'Wait for the eb500 radio to be ready
PAUSE 1000
Main:
      SERIN 0,84,[STR szData\20\CR]
      DEBUG STR szData,CR
      GOTO Main
```

4. On the **File** menu, click **Save As**.
5. In the **File name** box, enter a file name to which to save the program just created. For example, ReceivePPC.bs2.
6. Click **Save**.
7. Apply **power** to the **Board of Education board**.

8. On the **Run** menu, click **Run**.

   This will display the Download Progress dialog while downloading the program to the BASIC Stamp. After the download is complete, the Debug Terminal #1 dialog will be shown.

9. On the iPAQ, tap the **Bluetooth icon** in the system tray on the Today screen and select **Bluetooth Settings**.

   This will display the Settings dialog.

10. Scroll to the right, tap the **Serial Port** tab and note the **Outbound COM port**.

    The Outbound COM port will be used in the iPAQ application later in this step.

11. Download the **Hello World** Pocket PC application to the **iPAQ**.

12. Run the **Hello World** Application (Figure 12).

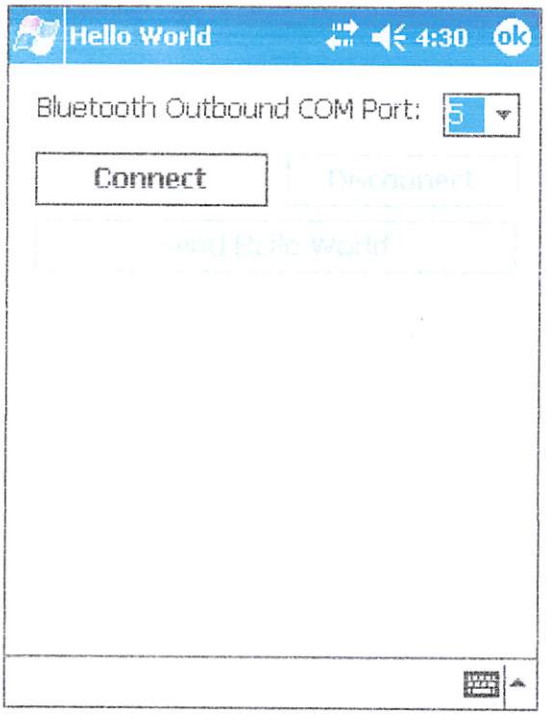


Figure 12: Hello World Pocket PC Application

13. In the **Bluetooth Outbound COM Port** dropdown, select the **COM port number** that matches the Bluetooth Outbound COM Port, which we discovered in a previous action.

14. Tap the **Connect** button.

    This will display the Bluetooth Browser dialog (Figure 13).

15. Tap **eb500** in the Bluetooth Browser dialog to establish a connection with the eb500 on the Board of Education.

    If there are no devices shown in the Bluetooth Browser dialog, tap the refresh icon to search for your Bluetooth device.

Figure 13: iPAQ Bluetooth Brower Dialog

16. Tap the **Send Hello World** button.

    This will transmit the ASCII text "Hello World" over the wireless link. The BASIC Stamp application will then receive these characters and display them in the BASIC Stamp Editor Debug window.

17. Tap the **Disconnect** button to close the Bluetooth connection.

ep 2: Transmit Data from the BASIC Stamp to the iPAQ

In this step we will create a BASIC Stamp application to send data out the eb500 to the iPAQ. We will then download and run the application. The application for the iPAQ is too verbose to include in this manual; therefore, the application, along with the source code, is available on the accompanying CD in the Samples folder. To modify the Pocket PC application you will need eMbedded Visual C++ 4.0 with Service Pack 2 and the SDK for Windows Mobile™ 2003-based Pocket PCs.

1. Using the BASIC Stamp Editor, on the **File** menu, click **New**.

   This will create a new project window within the BASIC Stamp Editor.

2. Enter the following **program code** into the editor, replacing the device address with the device address obtained from the iPAQ.

```
'{$STAMP BS2}
nCount VAR BYTE
'I/O Line 5 provides the connection status
INPUT 5
'Wait for the eb500 radio to be ready
PAUSE 1000
'Connect to the remote device
SEROUT 1,84,["con 00:04:3E:62:FE:01",CR]
SERIN 0,84,[WAIT("ACK",CR)]
WaitForConnection:
      IF in5 = 0 THEN WaitForConnection
DEBUG "Connected",CR
FOR nCount = 1 to 10
      SEROUT 1,84,["Hello World",CR]      'sending data
      PAUSE 1000                          'wait for 1 second
NEXT
'I/O Line 6 allows us to switch to Command Mode
OUTPUT 6
'Switch to Command Mode
LOW 6
SERIN 0,84,[WAIT(CR,">")]
'Disconnect from the remote device
SEROUT 1,84,["dis",CR]
SERIN 0,84,[WAIT(CR,">")]
DEBUG "Disconnected",CR
```

The BASIC Stamp application establishes a connection with the iPAQ, transmits "Hello World" ten times, switches back to command mode, and then disconnects from the remote device.

The first call to SEROUT is used when the eb500 is in command mode and instructs the eb500 to establish a connection with the device specified. Once a connection is established the eb500 is in data mode, which causes further calls to SEROUT to be sent to the remote device.

3. On the **File** menu, click **Save As**.

4. In the **File name** box, enter a file name to which to save the program just created. For Example, HelloWorld.bs2.
5. Click **Save**.
6. On the iPAQ, tap the **Bluetooth icon** in the system tray on the Today screen and select **Bluetooth Settings**.

   This will display the Settings dialog.
7. Scroll to the right, tap the **Serial Port** tab and note the **Inbound COM port**.

   The Inbound COM port will be used in the iPAQ application later in this step.
8. Download the **RXEB500** Pocket PC application to the **iPAQ**.
9. Run the **RXEB500** Application (Figure 14).

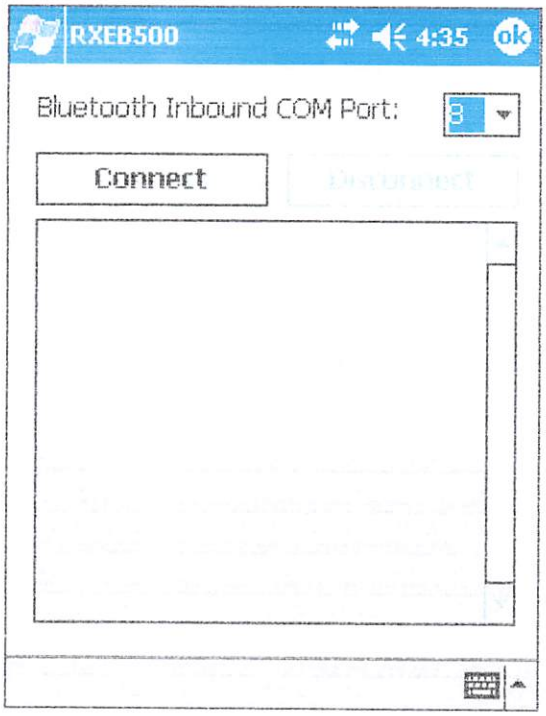


Figure 14: RXEB500 Pocket PC Application

10. In the **Bluetooth Inbound COM Port** dropdown, select the **COM port number** that matches the Bluetooth Inbound COM Port, which we discovered in a previous action.
11. Tap the **Connect** button.
12. Apply **power** to the Board of Education board.

13. Using the BASIC Stamp Editor, on the **Run** menu, click **Run**.

    This will display the Download Progress dialog while downloading the program to the BASIC Stamp. After the download is complete the BASIC Stamp application will establish a connection with the iPAQ. Depending on your current iPAQ Bluetooth configuration, the Authorization Request Dialog may appear (Figure 9). If this dialog appears, tap Accept to accept the connection. Once the connection is established, the BASIC Stamp application will transmit "Hello World" over the wireless link, and the iPAQ Pocket PC application will display the received data.

14. On the iPAQ, tap the **Disconnect** button to close the connection.

eb500 Commands

e eb500 command set is comprised of visible ASCII characters. Therefore, a command n be issued from a terminal application, such as HyperTerminal, or directly from a custom plication program, written in a programming language such as C++ or Visual Basic, ıning on a PC, using the eb600 PC Adapter. From a BASIC Stamp application, these mmands can be issued by using the PBASIC™ SERIN and SEROUT commands.

ommand Basics

mmands may only be sent to the eb500 when the module is in Command Mode. White aces are used to separate parameters of the command and a carriage-return is used to ırk the end of the command. Upon receipt of a command the eb500 begins to parse the rameters. If the syntax of the command is correct the eb500 returns an ACK string, not ACK character (0x06); otherwise, a NAK string is returned. Following the ACK or NAK ing is a carriage-return (0x0D) character. If an error occurs while processing the mmand an error string is returned followed by a carriage-return followed by the prompt (>) aracter. If the command executed successfully the eb500 will issue the prompt (>) aracter. Please see the eb500 Error Codes section for a description of the error codes.

e following example shows the basic structure of a command. A prompt (>) is issued by eb500. A command followed by a carriage-return is sent to the eb500. The eb500 ponds with either an ACK or NAK string followed by a carriage-return. If an error occurs, eb500 responds with an Err string followed by a space followed by an ASCII string meric value followed by a carriage-return. A prompt (>) is then issued by the eb500.

>*command*<CR>

ACK | NAK<CR>

Err *number*<CR>

>

ASIC Stamp Application eb500 Command Error Handling

e BASIC Stamp has a software based UART; meaning it does not buffer incoming serial ta. Therefore, the checking of errors from the issuing of an eb500 command must be rformed immediately after the issuing of the command; otherwise, the data may be lost. low is a sample of BASIC Stamp code that issues an eb500 Connect command, waits for ACK<CR> response from the eb500, then waits for the error string or the prompt (>) to returned from the eb500. It then checks the first bye of the data returned to determine if error has occurred. If an error has occurred, the code jumps to the error handler code, ere an error string along with the error number is shown in the debug window of the Basic amp Editor.

```
        'Connect to remote Bluetooth device
        SEROUT 1,84,["con 00:0C:84:00:07:D8",CR]
        SERIN 0,84,[WAIT("ACK",CR)]
        'Either an Err #<CR> or a ">" will be received
        SERIN 0,84,[STR bBuffer\6\">"]
        IF bBuffer(0) = "E" THEN ErrorCode
        … Progam Logic …
ErrorCode:
        bErrorCode = bBuffer(4)
        DEBUG "Error: ",STR bErrorCode,CR
        END
```

onnect

The *connect* command establishes a connection to another Bluetooth device. The *connect* command may be canceled before a connection is established by issuing a carriage-return to the eb500. The *connect* command may be canceled before the timeout is reached by issuing a carriage-return to the eb500. It can take up to four seconds to cancel the connection request.

Syntax

con *address* [*timeout*]<CR>

Parameters

address The Bluetooth address of the remote device. The Bluetooth device address is the 48-bit IEEE address which is unique for each Bluetooth unit. The format of a Bluetooth device address is a series of six hexadecimal byte values separated by colons, i.e., 00:0C:84:00:07:D7.

timeout An optional parameter used to abort the connection request after the specified number of seconds. The maximum value is 120 seconds.

Example

```
>con 00:0C:84:00:07:D7<CR>
ACK<CR>
>
```

isconnect

The *disconnect* command closes the connection with the remote Bluetooth device.

Syntax

dis<CR>

Example

```
>dis<CR>
ACK<CR>
>
```

et Address

The *get address* command returns the address of the local eb500 module.

Syntax

get addr<CR>

Returns

The unique address of the local eb500 module used to identify the module when making connections. In Bluetooth terminology this is the Bluetooth Device Address.

Example

```
>get addr<CR>
ACK<CR>
00:0C:84:00:07:D7<CR>
>
```

et Connectable Mode

The *get connectable mode* command returns the connectable mode setting of the local eb500 module.

Syntax

get con<CR>

Returns

The current connectable mode setting of the local eb500 module. In Bluetooth terminology, the returned value reflects the current setting for page scan.

on	The eb500 will accept connections.
off	The eb500 will NOT accept connections.

Example

```
>get con<CR>
ACK<CR>
on<CR>
>
```

et Discoverable Mode

The *get discoverable mode* command returns the discoverable mode setting of the local eb500 module.

Syntax

get dis<CR>

Returns

The current discoverable mode setting of the local eb500 module. In Bluetooth terminology, the returned value reflects the current setting for inquiry scan.

on	The eb500 is visible to other devices.
off	The eb500 is NOT visible to other devices.

Example

```
>get dis<CR>
ACK<CR>
on<CR>
>
```

et Escape Character

The *get escape character* command returns the current character used in the Switch to Command Mode command to instruct the eb500 to leave Data Mode and enter Command Mode.

Syntax

get escchar<CR>

Example

```
>get escchar<CR>
ACK<CR>
+<CR>
>
```

et Flow Control

The *get flow control* command returns the flow control setting of the local eb500 module.

Syntax

get flow<CR>

Returns

none	The eb500 is configured for no flow control and both the RTS and CTS lines are configured as high Z inputs. A BASIC Stamp application is free to use these lines as normal I/O without regard to the eb500.
hardware	The eb500 is configured for hardware flow control and the RTS line is used for receive flow control and the CTS line is used for transmit flow control.

Example

```
>get flow<CR>
ACK<CR>
none<CR>
>
```

et Link Timeout

The *get link timeout* command returns the amount of time it takes for the local eb500 module to notice that the connection has been broken, if the remote device disappears. This timeout also has an effect on how robust the communications link is to interference. If this value is set very low, the link may be lost if interference picks up for several seconds, such as when a heavy burst of 802.11 traffic is encountered.

Syntax

get linktimeout<CR>

Example

```
>get linktimeout<CR>
ACK<CR>
5<CR>
>
```

elp

The *help* command returns a listing of the eb500 commands and a brief description of each command.

Syntax

hlp [*command*]<CR>

Parameters

command	The eb500 command name (con, dis, get, lst, set and ver) for which to return help.

Examples

```
>hlp<CR>
ACK<CR>
... Help Information ...
<CR>
>

>hlp con<CR>
ACK<CR>
... Help Information on the Connect Command ...
<CR>
>
```

lst

The *list* command returns a listing of other Bluetooth devices that are in range and visible. The *list* command may be canceled before the timeout is reached by sending an additional carriage-return to the eb500.

Syntax

lst [*timeout*]<CR>

Parameters

timeout	An optional parameter used to abort the list request after the specified number of seconds. The default value is 30. The maximum value is 120 seconds.

Returns

The addresses of the Bluetooth devices that are in range and visible.

Example

```
>lst<CR>
ACK<CR>
00:0C:84:00:07:D7<CR>
00:80:C8:35:2C:B8<CR>
>
```

eturn to Data Mode

The *return to data mode* command instructs the eb500 to enter Data Mode when there is an active connection.

Syntax

ret<CR>

Example

```
>ret<CR>
ACK<CR>
>
```

et Baud Rate

The *set baud rate* command sets the baud rate for communications with the local eb500 module.

Syntax

set baud rate [*]<CR>

Parameters

rate	The baud rate value. Valid baud rates are 9600 (default), 19200, 38400, 57600, 115200, and 230400. Once the baud rate has been set, applications, such as HyperTerminal, must also be configured to the same baud rate to continue communicating with the eb500.
*	An optional parameter used to persist the new setting when the module is powered down.

Example

```
>set baud 19200<CR>
ACK<CR>
>
```

et Connectable Mode

The *set connectable mode* command provides control over whether the local eb500 module will accept connections from other Bluetooth devices. In Bluetooth terminology, this command controls the setting for page scan.

Syntax

set con on | off [*]<CR>

Parameters

on	Configures the eb500 so that other Bluetooth devices may establish a connection.
off	Configures the eb500 so that other Bluetooth devices may not establish a connection.
*	An optional parameter used to persist the new setting when the module is powered down.

Example

```
>set con off<CR>
ACK<CR>
>
```

et Discoverable Mode

The *set discoverable mode* command provides control over whether the local eb500 module is visible to other Bluetooth devices. In Bluetooth terminology, this command controls the setting for inquiry scan.

Syntax

set dis on | off [*]<CR>

Parameters

on	Configures the eb500 so that other Bluetooth devices may detect the presence of this eb500.
off	Configures the eb500 so that other Bluetooth devices may not detect the presence of this eb500.
*	An optional parameter used to persist the new setting when the module is powered down.

Example

```
>set dis on<CR>
ACK<CR>
>
```

et Escape Character

The *set escape character* command provides control over the character used in the Switch to Command Mode command to instruct the eb500 to leave Data Mode and enter Command Mode. The factory default escape character is the plus sign (+).

Syntax

set escchar character [*]<CR>

Parameters

character The character the eb500 should recognize as the escape character used in the Switch to Command Mode command.

Example

```
>set escchar & *<CR>
ACK<CR>
>
```

et Flow Control

The *set flow control* command provides control over the flow control setting of the local eb500 module.

Syntax

set flow none | hardware [*] <CR>

Parameters

none	Configures the eb500 for no flow control and both the RTS and CTS lines are configured as high Z inputs. This allows a BASIC Stamp application to use these lines a normal I/O without regard to the eb500.
hardware	Configures the eb500 for hardware flow control. The RTS line is used for receive flow control and the CTS line is used for transmit flow control.
*	An optional parameter used to persist the new setting when the module is powered down.

Example

```
>set flow none<CR>
ACK<CR>
>
```

et Link Timeout

The *set link timeout* command sets the amount of time it takes for the local eb500 module to notice that the connection has been broken, if the remote device disappears. This timeout also has an effect on how robust the communications link is to interference. If this value is set very low, the link may be lost if interference picks up for several seconds, such as when a heavy burst of 802.11 traffic is encountered. In Bluetooth terminology, this command controls the setting for link supervisor timeout.

Syntax

set linktimeout timeout [*]<CR>

Parameters

timeout	The time it takes for the eb500 to notice that a connection has been broken. The default value is 5. The maximum value is 40 seconds.
*	An optional parameter used to persist the new setting when the module is powered down.

Example

```
>set linktimeout 10<CR>
ACK<CR>
>
```

witch to Command Mode

The *switch to command mode* command instructs the eb500 to enter Command Mode.

Syntax

<2 second pause>*esc sequence*<2 second pause>

Parameters

esc sequence Three consecutive instances of the escape character. The factory default escape character is the plus sign (+). A different escape character can be set in the eb500 by using the Set Escape Character command.

Example


```
>con 00:0C:84:00:07:D7<CR>
ACK<CR>
>This text is sent in data mode<CR>
<2 second pause>+++<2 second pause><CR>
>get addr<CR>
ACK<CR>
00:0C:84:00:07:D8<CR>
>ret<CR>
ACK<CR>
>This text is sent in data mode<CR>
<2 second pause>+++<2 second pause><CR>
>dis<CR>
ACK<CR>
>
```

ersion

The *version* command returns the current firmware version of the local eb500 module.

Syntax

ver [all] <CR>

Parameters

all	An optional parameter used to return the build number, model number, serial number, and manufacturer.

Example

```
>ver all<CR>
ACK<CR>
Firmware Version: 1.0<CR>
Firmware Build: 189<CR>
Model Number: eb500<CR>
Serial Number: 8<CR>
Manufacturer: A7 Engineering<CR>
>
```

eb500 Error Codes

ıile using the eb500 you may encounter an error. Below is a listing of all eb500 error ıdes with a description of what causes the error to occur.

rror Code	Description
	General connection failure. This error occurs if the remote Bluetooth device is not configured properly. For example, this error may occur if the remove device requires Bluetooth security for a serial port connection.
	Connection attempt failed. This error occurs when attempting to connect with an invalid Bluetooth address or a device that is not available.
	Command not valid while active. This error occurs when there is an active connection and a command is issued that is not valid while connected with a remote device.
	Command only valid while active. This error occurs when there is not an active connection and a command is issued that is only valid while connected with a remote device.

Table 1: eb500 Error Codes

Technical Specifications

perating Parameters

e operating parameters of the eb500 are shown below in Table 2.

ransmit Power	4dBm (max) class 2 operation
pen Field Range	More than 100 meters (328 feet)
eceiver Sensitivity	-85dBm
perating Temperature	0° to 70°C
upply Power	5 to 12VDC
urrent Consumption	115.2kbps data transfer: 35mA 38.4kbps data transfer: 30mA 9.6kbps data transfer: 25mA connected and idle: 8mA no connection: 3mA
terfaces	5V logic level UART or RS232 serial w/ optional eb600 adapter Baud rate 9.6k – 230.4k Flow control: RTS/CTS or none
onnector	10x2 AppMod compatible 20 pin header with 0.1" spacing
ntenna	Matched internal surface mount
uetooth Support	Version 1.1 compliant with profiles L2CAP, RFCOMM, SDP, SPP
rmware	Upgradeable with optional eb600 adapter

Table 2: eb500 Operating Parameters

imensions

ne dimensions of the eb500 are shown below in Table 3. Please reference Figure 15 to cate the referenced dimension on the eb500.

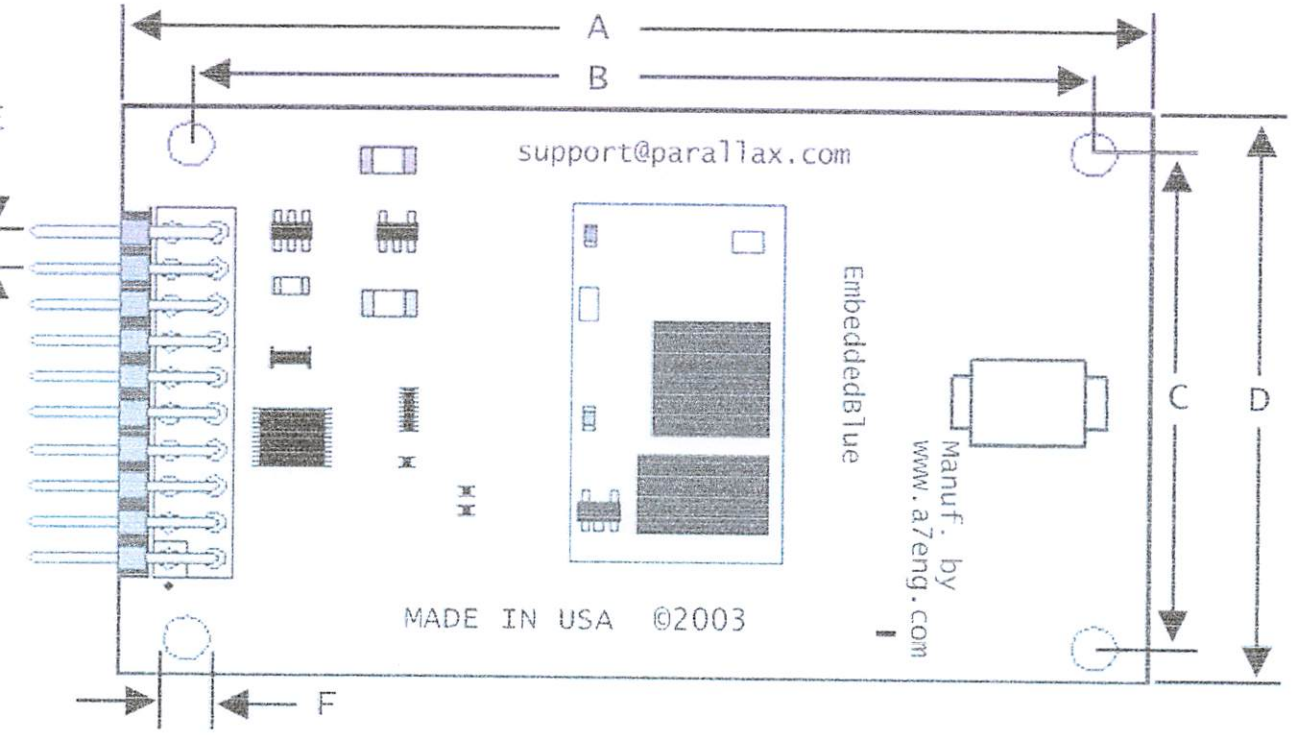


Figure 15: eb500 Dimensions

Dimension	inches	mm
A	2.7	68
B	2.4	61
C	1.3	33
D	1.6	40.2
E	0.1	2.5
F	0.125	3.2

Table 3: eb500 Dimensions

inout Diagram

e eb500 module features a 20 pin header with 0.1" spacing for connection to the AppMod ader. Currently, nine of the pins are in use (seven when flow control is set to none). The ıer pins are reserved for future use.

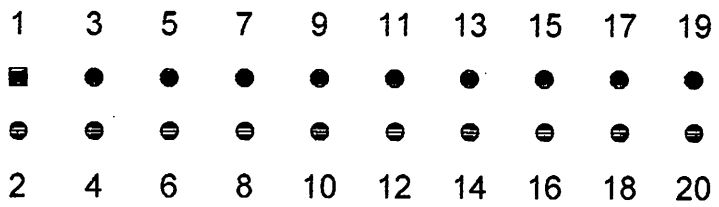


Figure 16: eb500 Pinout Diagram

in Name	Pin	Type	Description
SS	1, 2	GND	Ground
X	3	TTL output	UART data output
X	4	CMOS/TTL input	UART data input
X Flow (RTS)	5	CMOS/TTL input, weak pull down	Signaled high to stop module data transmission
X Flow (CTS)	6	TTL output	Signaled high to stop host data transmission
	7	Reserved	Reserved for future use.
onnection tatus	8	TTL output	High when there is an active wireless connection
ode Control	9	CMOS/TTL input, weak pulled down	Low for command mode / High for data mode
	10 - 19	Reserved	Reserved for future use.
N	20	VCC	Module supply, 5 to 12V DC

Table 4: eb500 Pinout Description

Frequently Asked Questions

uestion: My robots appear to be operating normally, but why will my eb500 not connect?

nswer: Check the battery power to the robot. Although the motors may still run, the voltage from the batteries may have fallen well below 5V preventing the eb500 from operating normally.

uestion: I changed the baud rate on my eb500 to 115200 and persisted the setting. I am unable to communicate at that speed from my BASIC Stamp. How can I reset the value to 9600 baud?

nswer: You can reset the eb500 module to its factory default settings by shorting Pin 8 and Pin 9 and applying power to the eb500 module. Alternatively, if you have an eb600 PC Adapter you can modify and persist a new setting using your PC, since the PC is capable of communicating at 115200 baud.

uestion: How do I obtain eMbedded Visual C++ 4.0 to develop Pocket PC applications?

nswer: The eMbedded Visual C++ 4.0 development tool is available from Microsoft. In addition, you will need eMbedded Visual C++ 4.0 SP2 and the SDK for Windows Mobile™ 2003-based Pocket PCs. These tools can be downloaded free of charge from the Microsoft Windows Mobile web site: http://www.microsoft.com/windowsmobile.

uestion: Why is my eb500 not displayed when I try to discover it from my PC or iPAQ?

nswer: Verify that the eb500 module is properly powered. Check the battery power to the robot. It is likely you will discover the eb500 on the first attempt; however, because Bluetooth discovery is not deterministic, discovery on the first attempt is not guaranteed. On the PC or iPAQ, use the refresh option to search for devices again. Verify that the discoverable mode setting in the eb500 is set to on.

uestion: I can discover my eb500, but why am I unable to establish a connection?

nswer: Verify that the connectable mode setting in the eb500 is set to on.

uestion: I have Bluetooth Authentication Security enabled on my iPAQ. Why am I unable to connect my iPAQ to the eb500 on my Board of Education board?

nswer: Currently, the eb500 does not support Bluetooth Authentication Security. However, you can achieve a measure of security for receiving connections on your iPAQ by enabling Bluetooth Authorization.

Contact Information

arallax provides technical support both through email, an online newsgroup, and by ephone. It is recommended that you use email as the first line of questioning because mmon questions can be answered quickly and in greater detail in this manner

ebsite: www.parallax.com

pport Email: support@parallax.com

les Email: sales@parallax.com

rallax, Inc

9 Menlo Drive, Suite 100

cklin, Ca 95765

8.512.1024 main

6.624.8003 fax

Engineering has created the EmbeddedBlue product line of easy to use wireless utions for 8 and 16 bit embedded systems. In addition, A7 provides several levels of pport for OEM product integration, certification, and even custom solutions.

ebsite: www.a7eng.com

les Email: sales@a7eng.com

Engineering Incorporated

360 Danielson Court, Suite C

way, Ca 92064

.679.7708 main

.391.5616 fax

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

Malang, 12 Juni 2008

Nomor : ITN-052/1.TA/7/08
Lampiran : -
Perihal : BIMBINGAN SKRIPSI

Kepada : Yth. Sdr. **Ir. F. Yudi Limpraptono, MT**
Dosen Institut Teknologi Nasional

Dosen Pembimbing
Jurusan Teknik Elektro S-1
di
Malang

Dengan hormat
Sesuai dengan permohonan dan persetujuan dalam Proposal Skripsi
Untuk Mahasiswa :

Nama	: I made Yudi Ariyanto
Nim	: 0612905
Fakultas	: Teknologi Industri
Jurusan	: Teknik Elektro S-1
Konsentrasi	: Teknik Elektronika

Maka dengan ini pembimbingan tersebut kami serahkan sepenuhnya kepada Saudara/I selama masa waktu (enam) bulan, terhitung mulai tanggal :

9 Juni 2008 s-d 9 Desember 2008

Sebagai satu syarat untuk menempuh Ujian Sarjana Teknik,
Jurusan Teknik Elektro S-1
Demikian agar maklum atas perhatian serta bantuannya kami sampaikan terima kasih.

Ketua Jurusan
Teknik Elektro S-1

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

Tindasan Kepada Yth :
1. Mahasiswa yang bersangkutan
2. Arsip
3. * coret yang tidak perlu

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

Malang, 12 Juni 2008

Nomor : ITN-053/1.TA/7/08
Lampiran : -
Perihal : BIMBINGAN SKRIPSI

Kepada : Yth. Sdr **Joseph Dedy Irawan, ST, MT**
Dosen Institut Teknologi Nasional

Dosen Pembimbing
Jurusan Teknik Elektro S-1
di
Malang

Dengan hormat
Sesuai dengan permohonan dan persetujuan dalam Proposal Skripsi
Untuk Mahasiswa :

Nama	: I made Yudi Ariyanto
Nim	: 0612905
Fakultas	: Teknologi Industri
Jurusan	: Teknik Elektro S-1
Konsentrasi	: Teknik Elektronika

Maka dengan ini pembimbingan tersebut kami serahkan sepenuhnya kepada Saudara/I selama masa waktu (enam) bulan, terhitung mulai tanggal :

9 Juni 2008 s-d 9 Desember 2008

Sebagai satu syarat untuk menempuh Ujian Sarjana Teknik,
Jurusan Teknik Elektro S-1
Demikian agar maklum atas perhatian serta bantuannya kami sampaikan terima kasih.

Ketua Jurusan
Teknik Elektro S-1

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

Tindasan Kepada Yth :
1. Mahasiswa yang bersangkutan
2. Arsip
3. *) coret yang tidak perlu

INSTITUT TEKNOLOGI NASIONAL MALANG
Kampus I : Jl. Bendungan Sigura-gura No. 2, Malang
Kampus II : Jl. Raya Karang Ploso Km. 2, Malang

FORMULIR BIMBINGAN SKRIPSI

Nama : I Made Yudi Ariyanto
Nim : 06.12.905
Masa Bimbingan : 9 Juni 2008 s/d 9 Desember 2008
Judul : Downloader Mikrokontroler AT89S51 Menggunakan Teknologi Bluetooth

NO	TANGGAL	URAIAN	PARAF PEMBIMBING
1	4/8/2008	Revisi Bab I	
2	7/8/2008	Revisi Bab II	
3	26/8/2008	Revisi Bab III	
4	3/9/2008	Revisi Bab IV	
5	4/9/2008	Revisi Bab V	
6	4/9/2008	Acc Akhir	
7			
8			
9			
10			

Malang, September 2008
Dosen Pembimbing I

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

INSTITUT TEKNOLOGI NASIONAL MALANG
Kampus I : Jl. Bendungan Sigura-gura No. 2, Malang
Kampus II : Jl. Raya Karang Ploso Km. 2, Malang

FORMULIR BIMBINGAN SKRIPSI

Nama : I Made Yudi Ariyanto
Nim : 06.12.905
Masa Bimbingan : 9 Juni 2008 s/d 9 Desember 2008
Judul : Downloader Mikrokontroler AT89S51 Menggunakan Teknologi Bluetooth

NO	TANGGAL	URAIAN	PARAF PEMBIMBING
1	27/8/2008	Revisi Bab III	
2	4/9/2008	Revisi Bab IV, Flowchart & Pengujian Software	
3	6/9/2008	Revisi Bab V	
4	6/9/2008	Acc Akhir	
5			
6			
7			
8			
9			
10			

Malang, September 2008
Dosen Pembimbing II

Yoseph Dedy Irawan, ST, MT
NIP. 132315178

INSTITUT TEKNOLOGI NASIONAL MALANG
Kampus I : Jl. Bendungan Sigura-gura No. 2, Malang
Kampus II : Jl. Raya Karang Ploso Km. 2, Malang

LEMBAR BIMBINGAN SKRIPSI

Nama	: I Made Yudi Ariyanto
Nim	:06.12.905
Jurusan	: Teknik Elektro Strata -1
Konsentrasi	: Elektronika
Judul Skripsi	: Downloader Mikrokontroler AT89S51 Menggunakan Teknologi Bluetooth
Tanggal Pengajuan	: 9 Juni 2008
Tanggal Penyelesaian	: 23 September 2008
Dosen Pembimbing I	: Ir. F. Yudi Limpraptono, MT
Dosen Pembimbing II	: Yoseph Dedy Irawan, ST, MT
Telah Dievaluasi Dengan Nilai I	: 90 (sembilan puluh)
Telah Dievaluasi Dengan Nilai II	: 86 (delapan puluh enam)

Malang, Oktober 2008
Mengetahui,

Dosen Pembimbing I	Dosen Pembimbing II
Ir F. Yudi Limpraptono, ST, MT	Joseph Dedy Irawan, ST, MT

INSITUT TEKNOLOGI NASIONAL MALANG
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO S-1

LEMBAR REVISI SKRIPSI

Nama : **I MADE YUDI ARIYANTO**
NIM : **06.12.905**
Jurusan : **TEKNIK ELEKTRONIKA STRATA- I**
Judul Skripsi : **DOWNLOADER MIKROKONTROLLER AT89S51 MENGGUNAKAN TEKNOLOGI BLUETOOTH**

No	Tanggal	Keterangan	Paraf
1.	25 September 2008	Daftar Pustaka	
2.	25 September 2005	Tata Tulis	

Dosen Pembimbing I

(Ir. F. Yudi Limpraptono, MT)

Dosen Pembimbing II

(Joseph Dedy Irawan ST, MT)

Dosen Penguji I

(Komang Somawirata,ST, MT)

Dosen Penguji II

(Irmalia Suryani Faradisa, ST)

INSTITUT TEKNOLOGI NASIONAL MALANG
Kampus I : Jl. Bendungan Sigura-gura No. 2, Malang
Kampus II : Jl. Raya Karang Ploso Km. 2, Malang

BERITA ACARA UJIAN SKRIPSI
FAKULTAS TEKNOLOGI INDUSTRI

Nama Mahasiswa	: I Made Yudi Ariyanto
NIM	: 06.12.905
Jurusan	: Teknik Elektro S-I
Konsentrasi	: Elektronika
Judul Skripsi	: Downloader Mikrokontroler AT89S51 Menggunakan Teknologi Bluetooth

Dipertahankan dihadapan Team Penguji Skripsi Jenjang Strata I pada :

Hari	: Selasa
Tanggal	: 23-09-2008
Dengan Nilai	: 84,1 (A)

Panitia Ujian Skripsi

INSTITUT TEKNOLOGI NASIONAL FAKULTAS TEKNOLOGI INDUSTRI DEKAN

Ir. Mochtar Asroni, MSME
Ketua

Ir.F. Yudi Limpraptono, MT
Sekretaris

Anggota Penguji

Komang Somawirata, ST, MT
Penguji Pertama

Irmalia Suryani F, ST, MT
Penguji Kedua