SKRIPSI

PERANCANGAN DAN PEMBUATAN SISTEM PENGAMAN DATA *WORD* DENGAN KRIPTOGRAFI *CAESAR* MENGGUNAKAN MIKROKONTROLER AT89S8252

Disusun Oleh:
SYAHRIR RAMADHAN
NIM 03.17.075

JURUSAN TEKNIK ELEKTRO S-1
KONSENTRASI TEKNIK ELEKTRONIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
2009

LEMBAR PERSETUJUAN

PERANCANGAN DAN PEMBUATAN SISTEM PENGAMAN DATA *WORD* DENGAN KRIPTOGRAFI *CAESAR* MENGGUNAKAN MIKROKONTROLER AT89S8252

SKRIPSI

Disusun dan Diajukan Untuk Melengkapi dan Memenuhi Syarat-Syarat Guna Mencapai Gelar Sarjana Teknik Elektro Strata Satu (S-1)

Disusun oleh :
SYAHRIR RAMADHAN
03.17.075

Dosen Pembimbing I

Ir. Yusuf Ismail Nakhoda, MT
NIP. Y. 1018800189

Dosen Pembimbing

M. Ashar, ST, MT
NIP. Y. 1030500408

Mengetahui,
Ketua Jurusan Teknik Elektro S-1

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

JURUSAN TEKNIK ELEKTRO
KONSENTRASI TEKNIK ENERGI LISTRIK S-1
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI NASIONAL MALANG
2009

ABTSRAKSI

PERANCANGAN DAN PEMBUATAN SISTEM PENGAMAN DATA *WORD* DENGAN KRIPTOGRAFI *CAESAR* MENGGUNAKAN MIKROKONTROLER AT89S8252

Syahrir Ramadhan
Ir. Yusuf Ismail Nakhoda, ST, MT
M. Ashar, ST, MT

Perkembangan teknologi informasi yang sangat significant berdampak pada peningkatan konsumsi masyarakat akan teknologi yang pada bahasan ini adalah Personal Computer. Di sisi lain, perkembangan teknologi komputer yang sangat pesat juga berakibat pada kemudahan dalam hal pengaksesan informasi dari suatu data dalam device ini. Konsekuensi dari pernyataan di atas sekuritas dari informasi yang dimiliki oleh suatu data dalam PC akan terancam kerahasiaannya.

Berdasarkan statement yang diuraikan di atas maka penulis mencoba menghadirkan suatu sistem untuk melindungi suatu data rahasia dengan menerapkan metode Kriptografi Caesar yang dilengkapi dengan hardware sebagai mediator kunci untuk pengaman dari suatu data rahasia. Primary key dalam hardware bersama dengan private key yang dimasukan user akan dilakukan fungsi kunci dua arah. Kemudian data yang akan disamarkan (plainteks) akan dilakukan proses Caesar encode yang berakhir pada proses XOR antara hasil kunci dua arah dengan data yang telah mengalami Caesar encode

Result dari proses ini adalah suatu chiperteks dengan tingkat sekuritas yang berganda yaitu chiperteks yang diperoeh dari dua proses pengacakan.

***Kata kunci** : Data word, Kriptografi Caesar, XOR, Chiperteks, Plainteks, Primary Key, Final Key*

KATA PENGANTAR

Segala puji hanya milik Allah S.W.T. Robb semesta alam, atas rizki yang dilimpahkan kepada penyusun, sehingga makalah ini dapat diselesaikan sesuai dengan waktu yang diinginkan.

Kedua orang tua, serta adik – adik saya yang merupakan alasan terkuat untuk tetap bersemangat dalam menyelesaikan makalah yang diberi judul **"Perancangan dan Pembuatan Sistem Pengaman Data Word Dengan Kriptografi Caesar Menggunakan Mikrokontroller AT89s8252".**

Juga dalam lembaran ini penulis menyampaikan ucapan terima kasih yang sebanyak – banyaknya kepada :

1. Bapak Ir. Yusuf Ismail Nakhoda, MT, selaku dosen pembimbing I.
2. Bapak M. Ashar, ST, MT, selaku dosen pembimbing II
3. Bapak Ir. F. Yudi Limpraptono, MT, selaku Ketua Jurusan Teknik Elektro S-1, Fakultas Teknologi Industri, Institut Teknologi Nasional Malang.

Serta pihak – pihak lain yang telah membantu penyusun dalam menyelesaikan makalah skripsi ini.

Semoga skripsi ini dapat bermanfaat bagi kita semua, amin.

Malang, Maret 2009

Penulis

DAFTAR ISI

HALAMAN JUDUL i

LEMBAR PERSETUJUAN ii

ABSTRAKSI iii

KATA PENGANTAR iv

DAFTAR ISI v

DAFTAR GAMBAR viii

DAFTAR TABEL x

DAFTAR GRAFIK x

BAB I PENDAHULUAN

1.1. Latar Belakang 1

1.2. Tujuan 2

1.3. Rumusan Masalah 2

1.4. Batasan Masalah 2

1.5. Metodologi Penelitian 2

1.6. Sistematika Penulisan 3

BAB II TINJAUAN PUSTAKA

2.1. Pendahuluan 4

2.2. Mikrokontroller AT89s8252 4

2.3. Konverter RS232-USB FT232BM 9

2.4. Universal Serial Bus (USB) 13

2.4.1. Konektor USB 14
2.4.2. End Point 15
2.5. Kriptografi Caesar 16
2.5.1. Jenis – jenis Caesar Chipper 18

BAB III PERANCANGAN SISTEM

3.1. Pendahuluan 20
3.2. Desain Sistem 20
3.3. Perencanaan Perangkat Keras 22
3.3.1. Perencanaan Mikrokontroller AT89s8252 21
3.3.1.1. Rangkaian Clock 22
3.3.2. Konverter Serial to USB 23
3.3.2.1. Rangkaian Clock 24
3.4. Perencanaan Perangkat Lunak (Software) 25
3.4.1. Flowchart Sistem 25
3.4.1.1. Flowchart Identifikasi Hardware pada PC 27
3.4.1.2. Flowchart download Key Mikrokontroller 29
3.4.1.3. Flowchart Penerapan Kunci Dua Arah 29
3.4.1.4 Flowchart Enkripsi Data 30
3.4.1.5 Flowchart Dekripsi Data 31

BAB IV PENGUJIAN DAN SIMULASI SISTEM

4.1. analisa dan Pengujian 33
4.1.1. Pengujian Konverter Serial – USB 33
4.1.2. Analisa Perangkat Penyamaran Data 35
4.1.3. Analisa Penyamaran Kunci Dua Arah 38
4.1.3.1 Analisa XOR Primary key ke Private Key 38

4.2. Simulasi Program Aplikasi 44

4.2.1. Encode Data 48

4.2.2. Decode Data 50

BAB V KESIMPULAN DAN SARAN

5.1. Kesimpulan 53

5.2. Saran 54

DAFTAR PUSTAKA

LAMPIRAN

DAFTAR GAMBAR

Gambar 2-1 Blok diagram mikrokontroller AT89s8252 5

Gambar 2-2 *Special Function Register* 8

Gambar 2-3 Rangkaian *osilator external* 9

Gambar 2-4 Blok Diagram FT232BM 10

Gambar 2-5 FT232BM RS232 USB Konverter 12

Gambar 2-6 Konektor USB 13

Gambar 2-7 Fungsi *enkripsi* dan *dekripsi* sederhana 15

Gambar 3-1 Blok diagram sistem *encode/decode* 20

Gambar 3-2 Rangkaian Mikrokontroller AT89S52 22

Gambar 3-3 Rangkaian *Clock* AT89S8252 23

Gambar 3-4 Rangkaian konverter serial USB FT232BM 23

Gambar 3-5 Rangkaian *Clock* FT232BM 25

Gambar 3-6 *Flowchart* sistem *encode decode* data 26

Gambar 3-7 *Flowchart identifikasi hardware* 28

Gambar 3-8 *Flowchart download key* dari mikrokontroller 29

Gambar 3-9 *Flowchart* penerapan kunci dua arah 30

Gambar 3-10 *Flowchart encode* 31

Gambar 3-11 *Flowchart decode* 32

Gambar 4-1 Rangkaian Pengujian Serial *Interface* 33

Gambar 4-2 *Setting Port* pada PC 34

Gambar 4-3 Pengujian Serial Data pada DU-232 35

Gambar 4-4 Rangkaian Pengujian Perangkat Kode Lisensi 36

Gambar 4-5 *Setting Port* Serial pada *HyperTerminal* 37

Gambar 4-6 *Setting Parameter interface* pada *HyperTerminal* 37
Gambar 4-7 Tampilan Data *HyperTerminal* 38
Gambar 4-8 *Form About Author* 44
Gambar 4-9 Peringatan koneksi *hardware* 45
Gambar 4-10 *Form Aplikasi* 45
Gambar 4-11 Pemilihan COM Port 46
Gambar 4-12 Koneksi Dengan Hardware Sukses 46
Gambar 4-13 Tampilan Validasi Private key 47
Gambar 4-14 Visualisasi Key dari Hardware 47
Gambar 4-15 Pembentukan Key Acak (Fungsi Kunci 2 Arah) 48
Gambar 4-16 Open File untuk proses enkripsi 49
Gambar 4-17 Penyimpanan Sukses 49
Gambar 4-18 Data Hasil Encode 50
Gambar 4-19 Open File untuk proses dekripsi 50
Gambar 4-20 Messages Box Penyimpanan Sukses 51
Gambar 4-21 Data Hasil Decode 51
Gambar 4-22 Visualisasi Kapasitas data sama besar 52

DAFTAR TABEL

Tabel 2-1 Konfigurasi pin konektor FT232BM 12

Tabel 2-2 Konfigurasi / pin konektor USB 13

Tabel 2-3 Tabel Subtitusi Caesar chipper 16

BAB I
PENDAHULUAN

1.1. Latar Belakang

Perkembangan teknologi informasi yang sangat *significant* menunjukkan tingginya tingkat kebutuhan manusia akan teknologi informasi, yang berdampak pada kemudahan dalam hal penggunaan media penunjang teknologi tersebut.
Komputer sebagai salah satu media pendukung teknologi informasi memberikan banyak kemudahan untuk mendukung para pengguna dalam hal memanfaatkan teknologi informasi yang sedang berkembang.
Sejalan dengan perkembangan teknologi tersebut, melahirkan beberapa konsekuensi sebagai akibat dari mudahnya memperoleh informasi dari suatu data dalam komputer. Salah satu dampak negatif yang lahir dari problem di atas adalah sekuritas dari informasi yang dimiliki oleh suatu data dalam PC.

Pernyataan di atas merupakan problem yang menghasilkan berbagai macam metode oleh para ahli untuk mempertahankan sekuritas dari suatu data rahasia. Salah satu metode yang digunakan adalah Kriptografi *Caesar*.
Mengacu pada uraian di atas, maka penulis mencoba membuat suatu sistem sekuritas untuk data rahasia dengan menggunakan Kriptografi *Caesar* sebagai metode penyamaran data, serta memanfaatkan mikrokontroller AT89S8252 sebagai media perangkat keras yang difungsikan sebagai pendukung sekuritas dari data yang akan dirahasiakan.

Dengan adanya sistem ini diharapkan informasi dari suatu data dapat terjamin kerahasiaannya.

1.2. Tujuan

Tujuan dari skripsi ini adalah merancang dan membuat suatu sistem pengaman dokumen *word* menggunakan metode Kriptografi *Caesar* dengan perangkat keras sebagai media penyimpan *primery key*

1.3.Rumusan Masalah

Perumusan masalah dari sistem ini adalah bagaimana merancang dan membuat suatu sistem pengaman dokumen *word* dengan metode Kriptografi *Caesar* sebagai teknik penyamaran data serta adanya perangkat keras sebagai media penyimpan kunci

1.4.Batasan Masalah

Dalam pembahasan ini dibatasi dengan beberapa batasan masalah, yaitu:

1) Media penyimpanan kunci yang digunakan adalah Mikrokontroller AT89S8252
2) Pembuatan *software* pada PC menggunakan Microsoft Visual Basic 6.0.
3) Teknik penyamaran yang digunakan adalah metode Kriptografi *Caesar*.
4) Komunikasi antara PC dengan *Hardware* melalui *line* USB.
5) Dokumen yang disamarkan adalah dokumen *word* dengan ext. (**.doc, *.rtf, *.html)*

1.5. Metodologi Penelitian

Metode pembahasan yang digunakan dalam skripsi ini adalah sebagai berikut:

1. Studi literature mengenai komponen – komponen yang terkait dengan sistem

2. Perancangan perangkat keras
3. Perancangan perangkat lunak
4. Analisa sistem pengambilan kesimpulan dari hasil analisis

1.6. Sistematika Penulisan

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

BAB I PENDAHULUAN

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

BAB II LANDASAN TEORI

Membahas teori dasar penunjang perancangan dan pembuatan alat.

BAB III PERENCANAAN DAN PEMBUATAN ALAT

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

BAB IV PENGUJIAN ALAT

Membahas tentang pengujian peralatan secara keseluruhan dan analisa hasil pengujian.

BAB V PENUTUP

Berisikan kesimpulan dan saran.

BAB II

TINJAUAN PUSTAKA

2.1. Pendahuluan

Pada bab ini akan dibahas mengenai dasar – dasar teori penunjang yang meliputi komponen pendukung, serta metode yang digunakan untuk penyamaran data.

2.2. Mikrokontroller AT 89S8252

Mikrokontroler adalah salah satu dari bagian dasar dari suatu sistem komputer. Meskipun mempunyai bentuk yang jauh lebih kecil dari suatu komputer pribadi dan komputer mainframe, mikrokontroler dibangun dari elemen-elemen dasar yang sama. Secara sederhana, komputer akan menghasilkan output spesifik berdasarkan inputan yang diterima dan program yang dikerjakan.

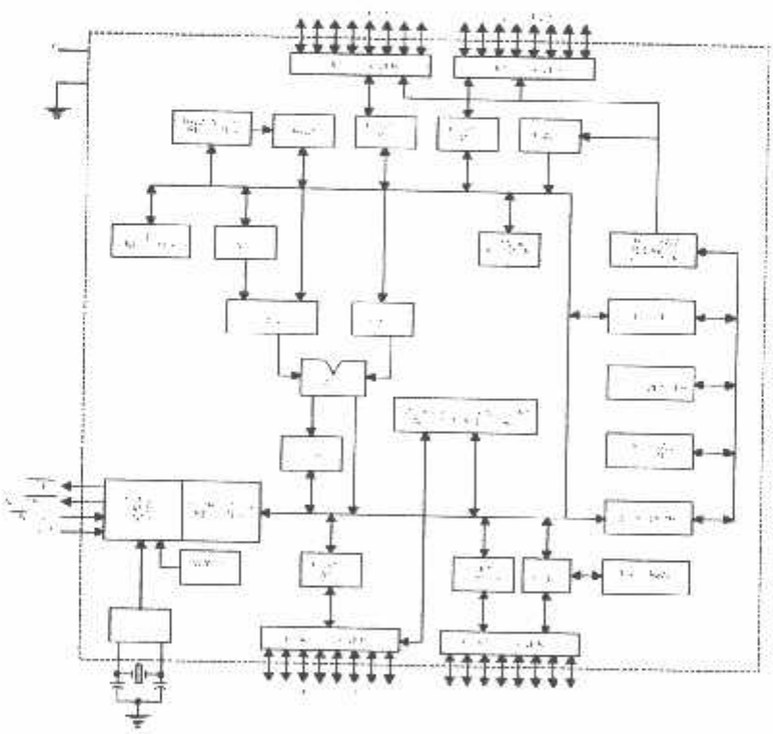
Mikrokontroler AT89S8252 merupakan mikrokontroller 8 bit kompatibel dengan standart industri MCS-51 baik atas segi pemrograman maupun atas kaki tiap pin.

Secara umum konfigurasi yang dimiliki mikrokontroller AT89S8252 adalah sebagai berikut :

- Sebuah CPU 8 bit dengan menggunakan teknologi dari Atmel
- 8 Kbyte Downloadable Flash Memory (PEROM)
- 2 Kbyte EEPROM
- 256 byte RAM internal
- 32 *Programmable I/O lines*

- Sebuah port serial dengan control *full duplex* UART (*Universal Asynchronous Receiver Transmiter*)
- SPI Serial Interface
- Programmable Watchdog Timer
- Dual Data Pointer
- Frekuensi kerja 0 sampai 24 MHz
- Tegangan operasi 2,7 sampai 6 Volt
- Kemampuan melaksanakan operasi *Boolean* (bit)

Berikut ini adalah blok diagram dari Mikrokontroller AT89s8252 :



Gambar 2.1 Blok diagram mikrokontroller AT89s8252 [5]

Pada AT89s8252 dikenal dua macam cara transmisi data serial yaitu transmisi data secara sinkron dan transmisi data secara asinkron dan untuk mengghubungkan port serial ini dibagi dalam 4 mode kerja.

Dari 4 mode kerja, ini 1 mode kerja diantaranya bekerja secara sinkron dan 3 lainnya bekerja secara asinkron, keempat mode kerja itu sebagai berikut :

1. Mode 0

Mode ini bekerja secara sinkron, data seri dikirim dan diterima melalui kaki P3.0 (RXD), dan kaki P3.1 (TXD) dipakai untuk menyalurkan clock pendorong data seri yang di bangkitkan olch AT89s8252. data dikirim/diterima 8 bit sekaligus, dimulai dari bibit yang bobotnya paling kecil (bit 0) dan diakhiri dengan bit yang bobotnya paling besar (bit 7) kecepatan pengiriman data (*baud rate*) adalah 1/12 frekuensi osilator kristal.

2. Mode 1

Mode ini dan mode-mode berikutnya bekerja secara asinkron, data dikirim melalui kaki P3.1 (TXD) dan diterima melalui kaki P3.0 (RXD). Pada Mode 1 data dikirim / diterima 10 bit sekaligus, diawal dengan 1 bit start, disusul dengan 8 bit data yang dimulai dari bit yang bobotnya paling kecil (bit 0), diakhiri dengan 1 bit stop. Pada AT89S8252 yang berfungsi sebagai penerima bit stop ditampung pada RB8 dalam register SCON. Kecepatan pengiriman data (*baud rate*) dapat disesuaikan dengan kebutuhan..

Mode inilah yang umum dikenal sebagai UART (*Universal Asynchronus Receiver/Transmitter*).

3. Mode 2

Data dikirim / diterima 11 bit sekaligus, diawali dengan 1 bit start, disusul 8 bt data yang dimulai dari bit yang bobotnya paling kecil (bit 0), kemudian bit 9 yang bisa

diatur lebih lanjut, diakhiri dengan 1 bit stop. Pada AT89S8252 yang brfungsi sebagai pengirim, bit 9 tersebut berasal dari bit TB 8 dalam register SCON. sedangkan bit stop diabaikan tidak di tampung. Kecepatan pengiriman data (*baud rate*) bisa dipilih antara 1/32 atau 1/64 frekensi osilator kristal.

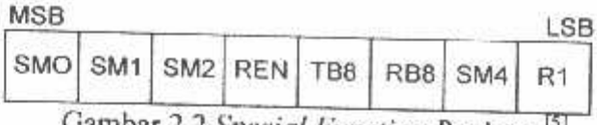
3. Mode 3

Mode ini sama dengan Mode 2, hanya saja kecepatan pengiriman data (*baud rate*) bisa diatur seduai dengan keperluan, seperti halnya mode 1. Dari keempat mode kerja diatas hanya mode asinkron yaitu mode 1, mode 2 dan mode 3 yang bisa bekerja secara *full duplex*, artinya pada saat yang sama port seri bisa mengirim dan menerima data secara bersamaan.

Register SBUF merupakan register penghubung port seri. Dalam keempat mode di atas, semua port seri mengrimkan data keluar dari 89S8252. Agar port seri bisa menerima data, bit REN dalam register SCON harus bernilai 1 Pada mode 0 proses penerimaan data dimulai dengan instruksi CLR RI, sedangkan dalam mode lainnya proses penerimaan data awali oleh bit start yang bernilai 0. Data yang diterima port seri dari luar89C51, diambil dengan instruksi MOV A, SBUF.

Mengambil data dari SBUF (*Serial Data Buffer*) dan menyimpan data ke SBUF sesungguhnya bekerja pada dua register berlainan.

Register status dan kontrol port serial pada *Special Function Register* ialah SCON yang susunannya dapat dilihat pada bagan di bawah ini. Register ini tidak hanya berisi bit untuk pengaturan mode saja, melainkan bit ke-9 untuk pengiriman dan peneriman data (TB8 dan RB8) ada di register ini, serta bit interupsi port serial (T1 dan R1).



Gambar 2.2 *Special Function Register* [5]

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

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

$$\text{Baud Rate} = \frac{(K \times \text{Frekuerisi Osilator})}{(32 \times 12 \times [256 - \text{TH1}])} \quad \text{.....(2.1)}$$

Nilai K ditentukan oleh bit SMOD dalam power control register (PCON). Bila SMOD = 0 maka K = 1, bila SMOD = 1 maka K = 2. Bila diketahui baud rate, nilai TH1 dapat dicari melalui persamaan berikut ini : [4]

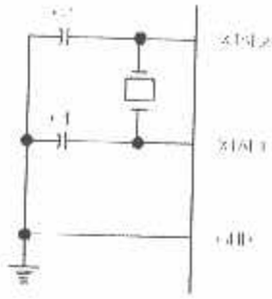
$$\text{TH1} = \frac{256 - (K \times \text{Frekuensi Osilator})}{(384 \times \text{Baud Rate})} \quad \text{.....(2.2)}$$

Nilai TH1 harus dalam bentuk integer. Pembulatan nilai TH1 pada nilai integer terdekat tidak akan menghasilkan baud rate yang dikehendaki. Dalam hal ini pemakai dapat mengganti frekuensi kristal. Karena PCON tidak dapat dialamati per bit, untuk men-set PCON dilakukan dengan mengirimkan perintah :

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

Pada port serial mode 2, baud rate memiliki nilai tetap yaitu 1/32 atau 1/64 dan frekuensi osilator tergantung pada nilai SMOD dalam register PCON. Pada mode ini tidak ada timer yang digunakan. Bila SMOD = 1, baud ratenya adalah 1/32 frekuensi osilator. Bila SMOD = 0, baud ratenya adalah 1/64 frekuensi osilator. Pada port serial

mode 3, baud rate dapat diatur seperti dalam mode 1. Berikut adalah contoh rangkaian oscilator pada mikrokontroler.

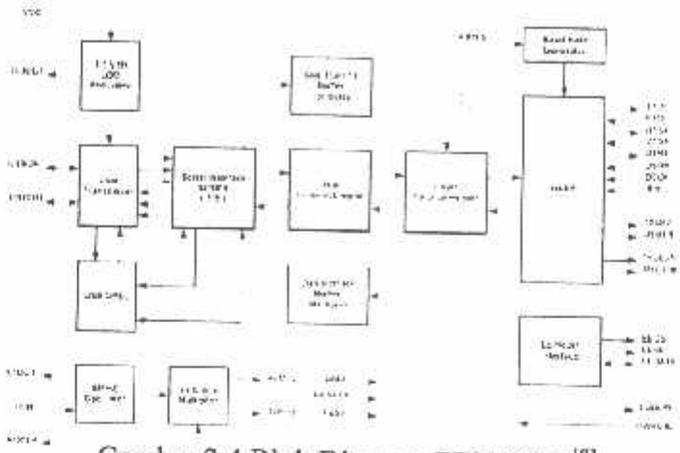


Gambar 2.3 Rangkaian *osilator external* [5]

2.3. Konverter RS232 - USB FT232BM

FT232BM adalah solusi efisien chip USB UART (U-UART) untuk mentransfer data serial dengan USB. Dengan kecepatan data transfer mencapai lebih besar dari 920kbaud untuk RS232 dan 2000k baud untuk RS422 atau RS485. FT232BM adalah solusi UART yang cukup baik untuk merancang suatu *plug n play peripheral, package* ini cukup fleksibel untuk digunakan pada berbagai aplikasi yang berbeda. IC ini menggunakan driver sebagai pengarah pada sisi software PC, dengan adanya driver maka akan terbentuk suatu virtual com port pada PC, driver untuk FT232BM dapat dijalankan pada sistem operasi Windows '98, Windows 98 SE, dan Windows 2000 dan Windows XP.

Berikut adalah blok diagram dari FT232BM.



Gambar 2.4 Blok Diagram FT232BM [8]

Berikut adalah penjelasan dari blok diagram diatas.

1) 3.3V LDO Regulator

   3.3V LDO regulator adalah sinyal yang akan membangkitkan 3.3 volt tegangan referensi untuk men-*drive* USB transceiver cell penyangga keluaran. Ini membutuhkan rangkaian eksternal berupa dekopling kapasitor yang di hubungkan pada pin 3V3OUT pin output regulator.

2) USB Transceiver

   Sinyal ini mendukung USB 1.1 *full speed* dengan antarmuka fisik yang terhubung dengan kable USB.

3) USB DPLL

   Sinyal ini adalah sinyal pengunci data USB.

4) 6 MHz Oscillator

   Sinyal ini adalah sinyal pembangkit denyut referensi sebesar 6MHz clock input untuk Clock multiplier X8 dari eksternal kristal 6MHz atau resonator keramik.

5) X8 Clock Multiplier

X8 Clock Multiplier menggunakan 6MHz yang merupakan input dari oscillator cell and dam membangkitkan denyut referensi sebesar 12MHz untuk sinyal SIE, USB Protocol Engine dan UART FIFO. Sinyal ini juga membangkitkan 48MHz denyut referensi untuk USB DPPL and the Baud Rate Generator blocks.

6) Serial Interface Engine (SIE)

SIE adalah sinyal yang berfungsi untuk mengadakan konversi data paralel-serial atau serial-paralel dari data USB.

7) USB Protocol Engine

Sinyal ini berfungsi untuk mengatur aliran data dari kontrol endpoint perangkat USB. Sinyal ini menggunakan protokol USB level rendah

8) Dual Port TX Buffer (128 bytes)

Data dari data keluaran endpoint pada USB akan disimpan sementara pada Dual port TX buffer dan akan dipindahkan dari buffer ke register UART transmit dibawah kendali UART FIFO kontroler.

9) Dual Port RX Buffer (384 bytes)

Data dari register penerima UART akan di simpan sementara pada Dual Port RX buffer sebelum akan dipindahkan oleh SIE dengan permintaan data dari USB yang berasal dari perangkat data pada endpoint.

10) UART FIFO Controller

UART FIFO controller menangani the pengiriman data antara Dual Port RX and TX buffers dan register UART transmit and receive.

11) UART

UART mendukung asynchronous 7 / 8 bit pengkonversian data Parallel-Serial dan Serial-Parallel pada antarmuka RS232 (RS422 and RS485). Sinyal kontrol didukung

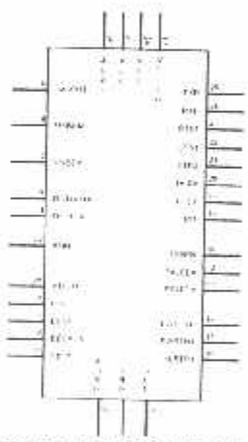
oleh UART RTS, CTS,DSR , DTR, DCD and RI. UART mengatur sinyal transmiter enable control signal (TXDEN) untuk berhubungan dengan penerima RS485.

12) Baud Rate Generator

Baud Rate Generator mendukung denyut masukan UART a x16 dari 48MHz denyut acuan dan menetapkan 14 bit resolusi dan 2 register bit dengan menyediakan sinyal baud rate (dibagi 2.5). Baud Rate dari UART dapat diproram dari 300 baud sampai 2 million baud.

13) EEPROM Interface

FT232BM dapat bekerja tanpa EEPROM yang bersifat opsional, sebuah memori eksternal 93C46 EEPROM dapat di gunakan FT232BM untuk aplikasi penyimpanan data yang bersifat permanen.



Gambar 2.5 FT232BM RS232 USB Konverter [8]

Berikut adalah keterangan dari susunan pin-pin pada FT232BM

Tabel 2.1 Konfigurasi pin konektor FT232BM [8]

PIN	Signal	Type	Description
7	USBDP	I/O	USB Data Signal Plus – Requires 1.5k pull-up to 3V3OUT
8	USBDM	I/O	USB Data Signal Minus
6	3V3OUT	OUT	3.3 volt Output from integrated regulator
27	XTIN	IN	Input to 6MHz Crystal Oscillator Cell
28	XTOUT	OUT	Output from 6MHz Crystal Oscillator Cell
31	TEST	IN	Puts device in i.c. test mode – must be tied to GND for normal operation.

31	TEST	IN	Puts device in i.c. test mode – must be tied to GND for normal operation.
4	RESET#	IN	Resets entire device using external RC network
32	EECS	I/O	Optional EEPROM – Chip Select
1	EESK	I/O	Optional EEPROM – Clock
2	EEDATA	I/O	Optional EEPROM – Data I/O
5	RSTOUT	OUT	Output of the internal Reset Generator.
25	TXD	OUT	UART – Transmit Data Output
24	RXD	IN	UART – Receive Data Input
23	RTS#	OUT	UART – Request To Send Control Output
22	CTS#	IN	UART – Clear To Send Control Input
21	DTR#	OUT	UART – Data Terminal Ready Control Output
20	DSR#	IN	UART – Data Set Ready Control Input
19	DCD#	IN	UART – Data Carrier Detect Control Input
18	RI#	IN	UART – Ring Indicator Control Input
16	TXDEN	OUT	UART – Enable Transmit Data for RS485
15	PWREN#	OUT	High after device is configured via USB
14	PWRCTL	IN	Bus Powered – Tie Low / Self Powered – Tie High
12	TXLED#	O.C.	LED Drive - Pulses Low when Transmitting Data via USB
11	RXLED#	O.C.	LED Drive - Pulses Low when Receiving Data via USB
10	SLEEP#	OUT	Goes Low during USB Suspend Mode
3,26	VCC	PWR	Device - +4.4 volt to +5.25 volt Power Supply Pins
13	VCCIO	POWER	+3.0 volt to +5.25 volt VCC to the UART interface pins 10..12, 14..16 and 18..25.
9.17	GND	PWR	Device – Ground Supply Pins
30	AVCC	PWR	Device - Analog Power Supply for the internal x8 clock multiplier
29	AGND	PWR	Analog Ground Supply for the internal clock

2.4. Universal Serial Bus (USB)

USB adalah port yang sangat diandalkan saat ini karena bentuknya yang kecil dan kecepatan transfernya yang tinggi. Anda dapat menghubungkan hingga 127 produk USB dalam satu komputer. USB mendukung dua modus kecepatan, yaitu mode kecepatan penuh (12 Mb/detik) dan kecepatan rendah (1,5 Mb/detik). USB 2.0 memiliki kecepatan 480 Mb/detik yang dikenal sebagai mode kecepatan tinggi.

2.4.1. Konektor USB

Ada dua macam konektor USB, yaitu konektor A untuk hubungan ke host, dan konektor B untuk hubungan ke perangkat USB. Kedua jenis konektor ini dapat dibedakan dengan mudah untuk menghindari kesalahan pemasangan.

Gambar 2.6 Konektor USB [1]

Tabel 2.2
Konfigurasi / pin konektor USB

Pin	Warna Kabel	Fungsi
1	Merah	V_{bus}(5 volt)
2	Putih	D-
3	Hijau	D+
4	Hitam	Ground

2.4.2. End point

Endpoint dapat dikatakan sebagai sumber data atau titik tujuan data yang merupakan ujung saluran komunikasi USB. *Endpoint* juga dilihat sebagai antarmuka fungsional perangkat keras yang berjalan di fungsi tersebut. Karena perangkat keras USB mengirim dan menerima data melalui beberapa *endpoint* yang terpasang secara seri, perangkat lunak disisi klien akan mentransfer data melalui *pipe*. *Pipe* adalah hubungan logika antara *host* dan *endpoint*. *Pipe* menentukan jenis transfer yang digunakan yaitu *control*, *bulk*, atau interupsi, serta arah dari data yang mengalir padanya, berikut ukuran maksimum paket atau ukuran buffer.

Pipe pada USB didefinisikan menjadi dua macam, yaitu:

1. Stream Pipe

   Tidak memiliki format, sehingga pemakai dapat mengirim sebarang jenis data dan menerima data melalui ujung USB. Data mengalir secara berurutan dan

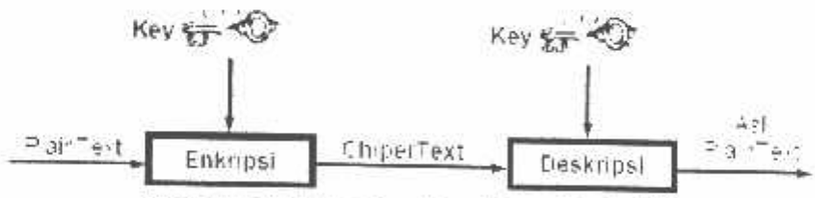
arahnya sudah ditentukan (in atau out). *Stream* dapat digunakan pada jenis transfer *bulk, isochronous,* dan interupsi. *Pipe* ini dapat dikendalikan melalui *host* atau klien.

2. Message type

   Formatnya sudah tertentu dan hanya dapat dikontrol oleh *host*. Data ditransfer dengan arah yang sudah ditentukan oleh permintaan pengirim dan dapat ditransfer dengan dua arah, serta hanya mendukung jenis transfer kontrol..

2.5. Kriptografi Caesar

Kriptografi *(cryptography)* merupakan ilmu dan seni penyimpanan pesan, data, atau informasi secara aman. Kriptografi *(cryptography)* berasal dari bahasa yunani dari kata *"crypto"* berarti *"secret"* (rahasia) dan *"grapia"* berarti *"writing"* (tulisan), sehingga kriptografi berarti penulisan rahasia. Kriptografi merupakan bagian dari suatu cabang ilmu matematika yang disebut *cryptology*.



Gambar 2.7 Fungsi *enkripsi* dan *dekripsi* [3]

Secara umum operasi *enkripsi* dan *dekripsi* dapat diterangkan secara matematis dinyatakan dalam persamaan berikut ini :

$Y = E_{KE}(X)$ proses *enkripsi*

$X = D_{KD}(Y)$ proses *dekripsi*

Dimana:

X adalah *plaintext* (informasi atau pesan asli/jelas)

Y adalah *chippertext* (informasi atau pesan yang tidak jelas atau rahasia)

KE adalah kunci *enkripsi*

KD adalah kunci *dekripsi*

Sebelum komputer ada, kriptografi dilakukan dengan suatu algoritma berbasis karakter. Caesar adalah algoritma kriptografi yang mula – mula digunakan. Dan orang yang menggunakannya adalah Kaisar Romawi (Julius Caesar) sehingga dinamakan *Caesar chipper*. Metode yang digunakan adalah dengan mengganti (*substitute*) setiap karakter dengan karakter lain dalam abjad (*alfabet*). Karena metode yang digunakan adalah substitusi, maka metode ini juga dikenal dengan sebutan *chipper substitute*.

Tiap huruf disubtitusikan dengan huruf ketiga berikutnya dari susunan abjad. Dalam hal ini kuncinya adalah pergeseran huruf (yaitu $k = 3$).

Tabel 2.3
Tabel Subtitusi *Caesar chipper*

Pi	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Ci	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C

Secara matematis *Caesar chipper* menyandikan *plainteks (Pi)* menjadi *chipperteks (Ci)* mengikuti aturan :

$$Ci = E(Pi) = (Pi + 3) \bmod 26$$

Dan dekripsi *chipperteks (Ci)* menjadi *plainteks (Pi)* adalah:

$$Pi = D(Ci) = (Ci - 3) \bmod 26$$

Karena hanya ada 26 huruf abjad, maka pergeseran huruf yang mungkin dilakukan adalah dari 0 sampai 25. Secara umum, pergeseran huruf sejauh k (dalam hal

ini k adalah kunci enkripsi atau dekripsi). Untuk pergeseran 0 maka tidak ada perubahan yang terjadi. Artinya *plainteks* sama dengan *chipperteks*. Pergeseran lain intuk $k > 25$ dapat juga dilakukan, yang hasilnya akan *kongruen* dengan bilangan bulat dengan *modulo* 26. Karena ada operasi penjumlahan dalam beberapa persamaan di atas, maka *Caesar chipper* kadang – kadang dinamakan juga *additive chipper*.

2.5.1. Jenis – jenis *Caesar Chipper*

a. *Chipper* abjad tunggal (*mono alphabetic chipper*)

Satu karakter di *plainteks* diganti dengan satu karakter yang bersesuaian. Yang berarti fungsi *chippering* adalah satu ke satu. Jika *plainteks* terdiri dari huruf – huruf abjad, maka jumlah kemungkinan penyusunan karakter dari *chipperteks* yang dapat dibuat adalah :

26! = 403.291.461.126.605.635.584.000.000 kemungkinan.

Caesar chiper adalah kasus khusus dari *chiper* abjad tunggal di mana susunan huruf *chipperteks* diperoleh dengan menggeser huruf-huruf alfabet sejauh 3 karakter.

ROT13 adalah program *enkripsi* sederhana yang ditemukan pada sistem UNIX. *ROT13* menggunakan *chiper* abjad-tunggal dengan pergeseran k = 13 (jadi, huruf A diganti dengan N, B diganti dengan O, dan seterusnya).

Enkripsi arsip dua kali dengan ROT13 menghasilkan arsip semula:

P = ROT13(ROT13(P))

b. *Chipper substitute homophonic*

Seperti *chipper* abjad tunggal, kecuali bahwa setiap karakter dalam *plainteks* dapat dipetakan ke dalam salah satu dari karakter *chipperteks* yang mungkin. Misalnya huruf A dapat berkoresponden dengan 7, 9, atau 16, huruf B dapat berkoresponden

dengan 5, 10, atau 23 dan seterusnya. Fungsi *chipering*-nya memetakan satu-ke-banyak (*one-to-many*).

Chiper substitusi homofonik digunakan pertama kali pada tahun 1401 oleh wanita bangsawan Mantua. Chiper substitusi homofonik lebih sulit dipecahkan daripada *chipper* abjad-tunggal. Namun, dengan *known-plaintext attack*, *chipper* ini dapat dipecahkan, sedangkan dengan *chipertext-only attack* lebih sulit.

c. Chiper abjad-majemuk (*Polyalpabetic substitution chiper*)

Merupakan *chipper* substitusi-ganda (*multiple-substitution chiper*) yang melibatkan penggunaan kunci berbeda.*Chiper* abjad-majemuk dibuat dari sejumlah *chipper* abjad-tunggal, masing-masing dengan kunci yang berbeda. Kebanyakan *chipper* abjad-majemuk adalah *chipper* substitusi periodik yang didasarkan pada periode *m*.

Misalkan plainteks *P* adalah

$$P = p_1 p_2 \ldots p_m p_{m+1} \ldots p_{2m} \ldots$$

maka chiperteks hasil enkripsi adalah

$$E_k(P) = f_1(p_1)\, f_2(p_2) \ldots f_m(p_m)\, f_{m+1}(p_{m+1}) \ldots f_{2m}(p_{2m}) \ldots$$

yang dalam hal ini p_i adalah huruf-huruf di dalam *plainteks.*

Untuk $m = 1$, *chipper*-nya ekivalen dengan *chipper* abjad-tunggal.

Contoh *chiper* substitusi periodik adalah *chipper Vigenere* yang ditemukan oleh kriptologi Perancis, Blaise de Vigenere pada abad 16. Misalkan *K* adalah deretan kunci

$$K = k_1\, k_2 \ldots k_m$$

yang dalam hal ini k_i untuk $1 \leq i \leq m$ menyatakan jumlah pergeseran pada huruf ke-*i*. Maka, karakter *chipperteks* $y_i(p)$ adalah :

$$y_i(p) = (p + k_i) \bmod n$$

Misalkan periode m = 20, maka 20 karakter pertama dilakukan prose *enkripsi* dengan persamaan ($yi(p) = (p + ki) \bmod n$), dimana setiap karakter ke-i menggunakan kunci k_i. Untuk 20 karakter berikutnya, kembali menggunakan pola *enkripsi* yang sama. *Chipper* abjad-majemuk ditemukan pertama kali oleh Leon Battista pada tahun 1568.

Metode ini digunakan oleh tentara AS selama Perang Sipil Amerika. Meskipun *chipper* abjad-majemuk dapat dipecahkan dengan mudah (dengan bantuan komputer), namun anehnya banyak program keamanan komputer (*computer security*) yang menggunakan *chipper* jenis ini.

d. Chiper substitusi poligram (*Polygram substitution chiper*)

Blok karakter disubstitusi dengan blok chiperteks. Misalnya ABΛ diganti dengan RTQ, ABB diganti dengan SLL, dan lain-lain. *Playfair chiper*, ditemukan pada tahun 1854, termasuk ke dalam *chipper* substitusi poligram dan digunakan oleh negara Inggris selama Perang DuniaI.

Pada dasarnya algoritma untuk melakukan suatu proses *enkripsi* dan *dekripsi* adalah sama. Yang membedakan dari kedua jenis proses tersebut adalah penjadualan kunci. Penjadualan kunci pada proses *enkripsi* akan terbalik dengan *dekripsi*, misalnya pada *enkripsi* suatu data dilakukan proses A1, A2, A3,...An. Maka pada proses *dekripsi* akan terjadi proses yang berlawanan yaitu An,...,A3, A2, A1.

BAB III
PERANCANGAN SISTEM

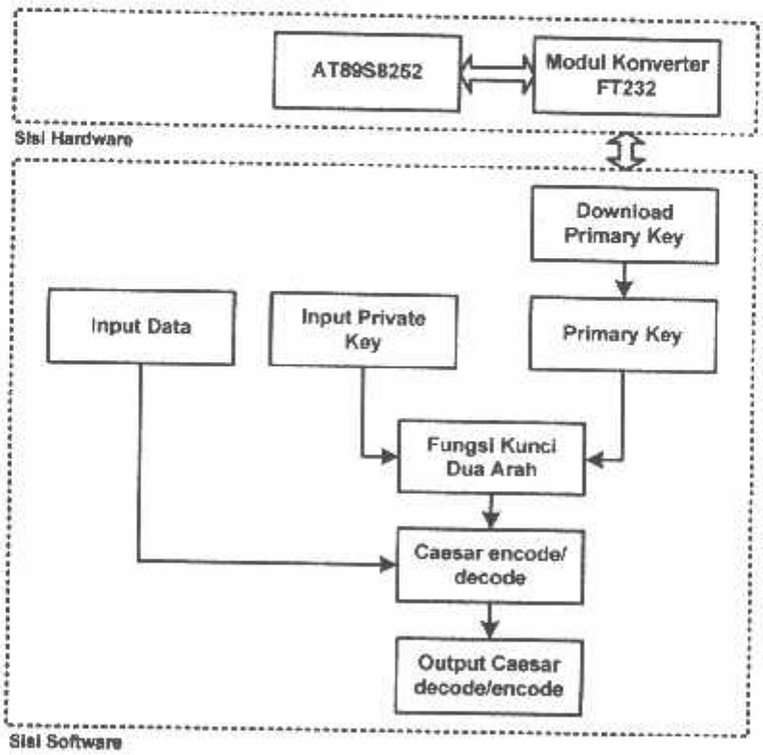
3.1. Pendahuluan

Pada bab ini akan dibahas mengenai perancangan sistem yang meliputi perancangan perangkat keras (*harware*) dan perancangan perangkat lunak (*software*) dari sistem pengaman data *word*

3.2. Desain Sistem

Sistem pengaman data *word* ini adalah suatu sistem yang menggunakan mikrokontroller AT89S8252 sebagai media penyimpan *primary key*, serta menggunakan metode *cryptography Caesar* sebagai teknik penyamaran data.

Di bawah ini adalah blok diagram sistem secara keseluruhan :



Gambar 3.1 Blok diagram sistem *encode/decode*

Dari blok diagram di atas, dapat dijabarkan:

a. Proses penyamaran kunci:

a. Proses penyamaran kunci:

Pada tahap awal *download primary key* dari *hardware* dilakukan oleh program aplikasi pada PC. Setelah itu *software* akan meminta masukan dari *user* berupa *private key,* yang nantinya *software* pada PC akan melakukan proses fungsi kunci dua arah antara *primary key* dan *private key*

b. Proses penyamaran data

Proses selanjutnya adalah tahap penerapan fungsi kunci dua arah untuk menghasilkan *final key,* sebagai persiapan proses *encode/decode* dokumen. *Final key* bersama dengan dokumen akan di parameterisasikan pada fungsi *Caesar encode/decode* data .

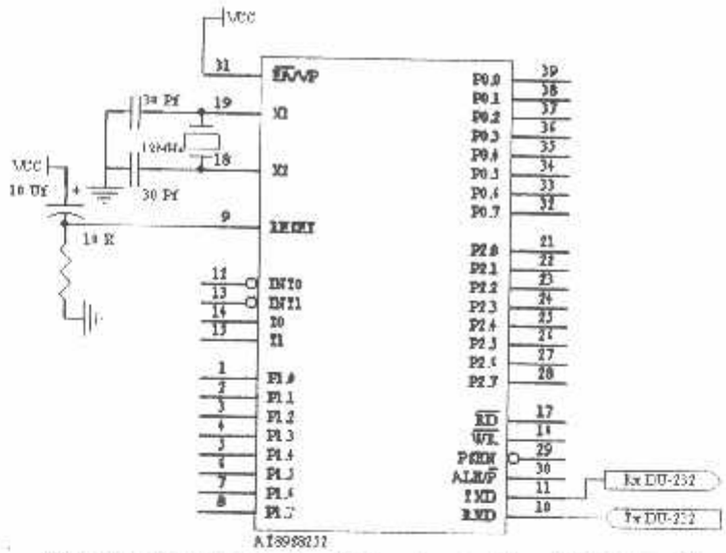
3.3. Perencanaan perangkat keras

Pada perancangan perangkat keras, terdapat beberapa modul yang akan dirangkai menjadi satu, yang nantinya bersama dengan *software* pada PC akan menjadi suatu sistem pengaman data *word.* Modul – modul tersebut adalah mikrokontroller, dan rangkaian konverter serial *to* USB.

3.3.1. Perencanaan Mikrokontroller AT89S52

Pada perencanaan ini mikrokontroler digunakan sebagai media yang akan mengirimkan data berupa *primery key* yang telah tersimpan pada memori internalnya kepada PC. Data yang akan dikirimkan mikrokontroler menggunakan komunikasi serial yang akan dikonversikan menjadi USB untuk di terima PC.

Berikut adalah skematik dari konfigurasi dari AT89S8252 pada sistem ini.



Gambar 3.2 Rangkaian Mikrokontroller AT89S52 [5]

Penjelasan dari pin-pin yang dipergunakan:

- Pin 9 sebagai reset.
- Pin 10 dan 11 dihubungkan modul konverter DU-232.
- Pin 18 dan 19 dihubungkan dengan Kristal 12 MHz.
- Pin 20 dihubungkan ke *Ground.*
- Pin 31 dihubungkan dengan sumber tegangan 5 Volt.
- Pin 40 dihubungkan dengan Vcc.

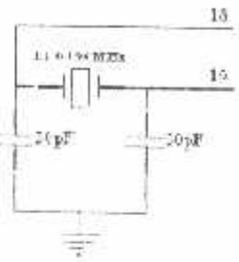
3.3.1.1. Rangkaian Clock

Mikrokontroller AT89S8252 ini memiliki rangkaian *internal clock* generator yang berfungsi sebagai sumber *clock*, tetapi masih diperlukan rangkaian tambahan untuk membangkitkan *clock* yang diperlukan.

Rangkaian ini terdiri dari dua buah kapasitor dan sebuah kristal, dengan ketentuan sebagai berikut:

- 6 – 14 MHz untuk besarnya nilai kristal.
- 27 – 33 pf untuk besarnya nilai kapasitor.

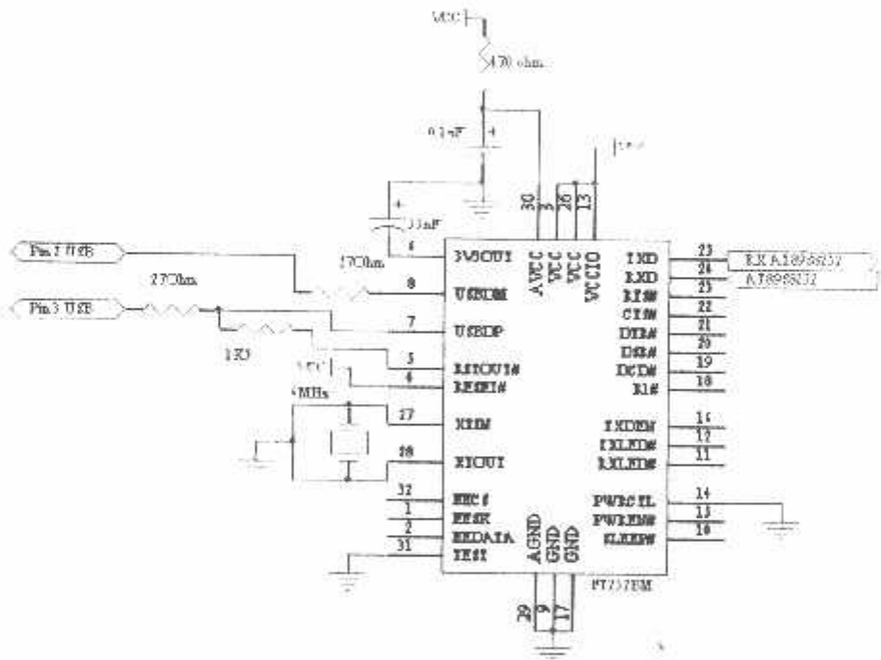
Gambar dari rangkaian clock adalah sebagai berikut:



Gambar 3.3 Rangkaian Clock AT89S8252 [5]

3.3.2. Konverter serial *to* USB

Komponen utama pada modul rangkaian ini adalah IC FT232BM. IC ini berfungsi sebagai pengubah data serial yang berasal dari mikrokontroler menjadi data USB yang akan diterima PC. Berikut adalah skematik dari rangkaian ini.



Gambar 3.4 Rangkaian *converter* serial USB FT232BM [8]

Konfigurasi pin yang digunakan pada rangkaian ini adalah sebagai berikut:

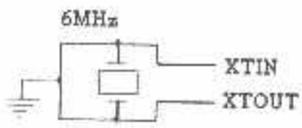
- Pin 30 dihubungkan dengan kapasitor sebesar 33nF dan *ground* merupakan *output* dari *regulator* internal.

Konfigurasi pin yang digunakan pada rangkaian ini adalah sebagai berikut:

- Pin 30 dihubungkan dengan kapasitor sebesar 33nF dan *ground* merupakan *output* dari *regulator* internal.
- Pin 8 dihubungkan dengan pin 2 USB pada PC sebagai sinyal data *negatif* yang terhubung paralel dengan tahanan sebesar 27 ohm dan pin 5 yang merupakan *output* dari internal reset generator.
- Pin 7 dihubungkan dengan pin 3 USB pada PC sebagai sinyal data *positif*
- Pin 5 dihubungkan dengan tahanan sebesar 15Kohm yang digunakan sebagai *output* reset dari FT232.
- Pin 27 dihubungkan dengan sinyal input rangkaian internal oscilator 6MHz.
- Pin 28 dihubungkan dengan sinyal output rangkaian internal oscilator 6MHz
- Pin 25 dihubungkan port RX dari mikrokontroler.
- Pin 24 dihubungkan dengan port TX dari mikrokontroler.
- Pin 29, 9, 17 terhubung dengan *ground*.
- Pin 3, 26, 13 terhubung dengan Vcc.
- Pin 14 terhubung dengan *ground* untuk mengeset mode *powered* pada rangkaian ini

3.3.2.1. Rangkaian Clock

IC FT232BM ini memiliki rangkaian internal clock dengan sinyal pembangkit denyut referensi sebesar 6MHz clock input untuk clock multiplier X8 dari eksternal kristal 6MHz atau resonator keramik, dan membangkitkan 12MHz clock referensi untuk SIE (*Serial Interface Enggine).USB protocol Enggine*, dan UART FIFO controller block.



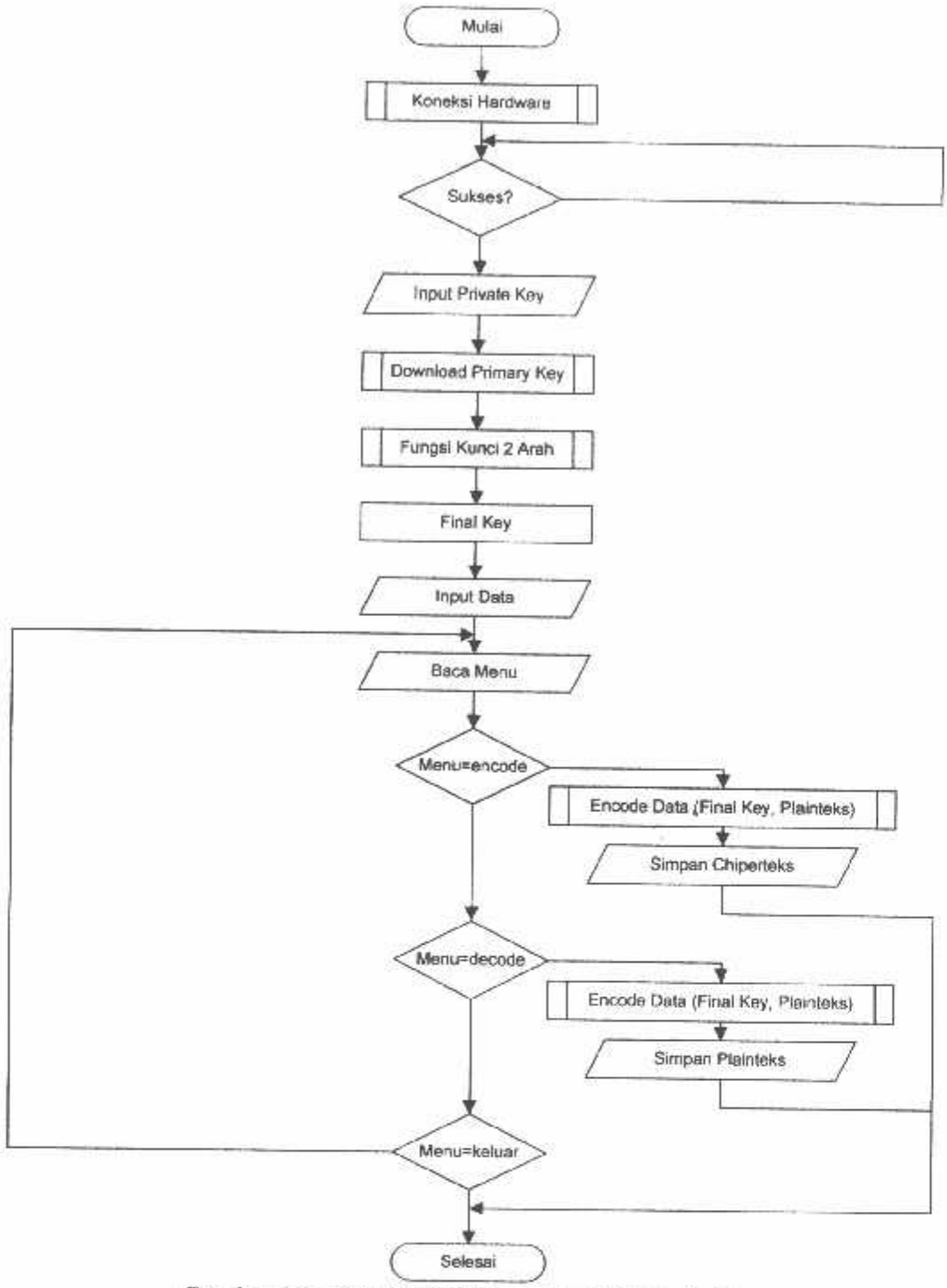
Gambar 3.5 Rangkaian *Clock* FT232BM [8]

3.4. Perencanaan Perangkat Lunak (*Software*)

Tahap awal dari perencanaan perangkat lunak adalah perancangan algoritma dari *software* yang akan dibuat.Perencanaan algoritma dilakukan agar pada pembuatan *listing* pada program lebih sistematis dan lebih mudah dalam hal penanganan kesalahan. Dalam perencanaan perangkat lunak ini, *compiler* yang digunakan adalah *Microsoft Visual Basic 6.0.*

3.4.1. *Flowchart* system

Flowchart dari keseluruhan sistem ditunjukan pada gambar di bawah ini:



Gambar 3.6 *Flowchart* sistem *encode/decode* data

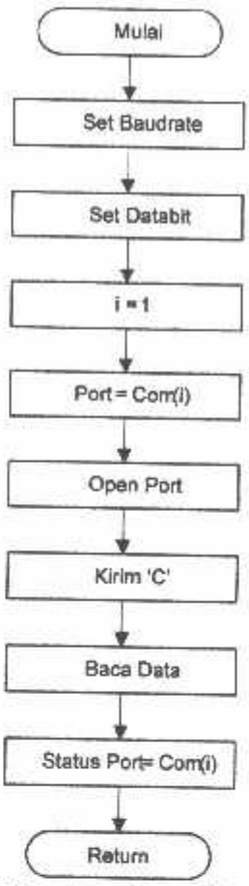
Algoritma dari *flowchart* di atas :

1. mulai
2. identifikasi *hardware*
3. apakah identifikasi sukses?
    a. ya, lakukan langkah 4

5. input (*private key*)
6. fungsi kunci dua arah
7. final key
8. baca menu
9. jika menu = *encode*
    a. ya, lakukan langkah 10
    b. tidak, lakukan langkah 13
10. load data
11. *caesar encode* (*Final key*, data asli)
12. simpan data acak
13. jika menu = *decode*
    a. ya, lakukan langkah 14
    b. tidak, lakukan langkah 17
14. load dokumen acak
15. *caesar decode* (*Final Key*, data acak)
16. simpan data asli
17. jika menu = selesai
    a. ya, lakukan langkah 18
    b. tidak, kembali kelangkah 8
18. selesai

3.4.1.1. *Flowchart* Identifikasi *Hardware* pada PC

Proses ini menjelaskan tentang antarmuka *hardware* dengan PC. Sehingga PC dapat mengetahui pada COM keberapa *hardware* terkoneksi.



Gambar 3.7 *Flowchart* identifikasi *hardware*

Pemaparan dari diagram alir di atas:

1. Mulai.
2. PC menginisialisasikan nominal baudrate yang digunakan.
3. PC menginisialisasikan databits yang digunakan.
4. Set pencacah i = 0
5. PC menginisalisasikan port koneksi COM ke-i.
6. Open port.
7. Kirim karakter "C".
8. Baca data.
9. Status Port = COM ke-i.
10. Lanjutkan ke proses *download primary key hardware*

3.4.1.2. *Flowchart download* kunci dari mikrokontroller ke PC

Proses pengiriman kunci dari mikrokontroller ke PC dilakukan pada saat akan dilakukan proses penyamaran data *word* oleh *software*.

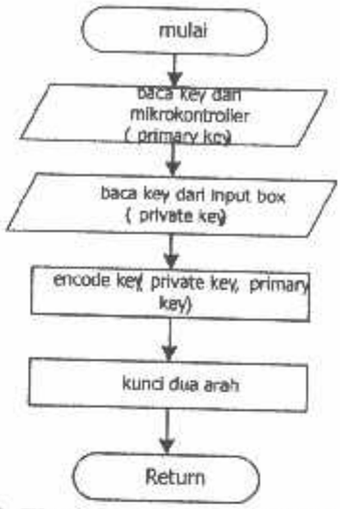
Gambar 3.8 *Flowchart download key* dari mikrokontroller

Penjelasan *linguistic* dari *flowchart*:

1. mulai.
2. kirim karakter "C" ke mikrokontroller
3. mikrokontroller mengirim *primary key* ke PC
4. kunci primer dari nikrokontroller
5. Lanjutkan ke proses *input private key* oleh *user*

3.4.1.3. *Flowchart* penerapan kunci dua arah

Penerapan kunci dua arah dilakukan pada saat akan melakukan proses manipulasi *file word*. Proses ini terjadi antara *primary key* dari mikrokontroller dan *private key* dari *input box* pada *software*.



Gambar 3.9 *Flowchart* penerapan kunci dua arah

Penjelasan dari *flowchart* di atas:

1. mulai
2. proses pembacaan kunci dari mikrokontroler oleh *software* pada PC
3. proses selanjutnya adalah pembacaan kunci dari *input box* pada *software*
4. *software* melakukan fungsi kunci dua arah antara *primary key* dan *private key*
5. hasil fungsi kunci dua arah
6. Menampilkan karakter kunci dua arah (*Final Key*)

3.4.1.4. *Flowchart enkripsi* data dengan metode *cryptography caesar*

Proses ini dilakukan pada saat melakukan *enkripsi* dari data yang akan diamankan.

Gambar 3.10 *Flowchart encode*

Penjelasan dari bagan alir di atas:

1. mulai
2. persiapan fungsi kunci dua arah untuk melakukan *Caesar enkripsi*
3. persiapan data yang akan dimanipulasi
4. kunci dua arah dan data dienkripsi menggunakan *Caesar chipper*
5. hasil dari dokumen yang telah dimanipulasi
6. Lanjutkan ke proses selanjutnya

3.4.1.5. *Flowchart dekripsi* data dengan metode *cryptography Caesar*

Pada dasarnya proses ini merupakan kebalikan dari proses sebelumya. Artinya jika tahap sebelumnya dianggap (N + 1, N + 2, ..., N + n), maka proses ini dapat dianggap (N + n, ..., N + 2, N + 1).

Gambar 3.11 *Flowchart decode*

Pemaparan dari diagram alir di atas:

1. mulai
2. fungsi kunci dua arah
3. *download* data yang telah tersamarkan
4. Caesar decode (kunci dua arah, data acak)
5. Data asli
6. Lakukan proses pada menu selanjutnya.

BAB IV
PENGUJIAN DAN SIMULASI SISTEM

4.1 Analisa dan pengujian

4.1.1 Pengujian Konverter Serial - USB

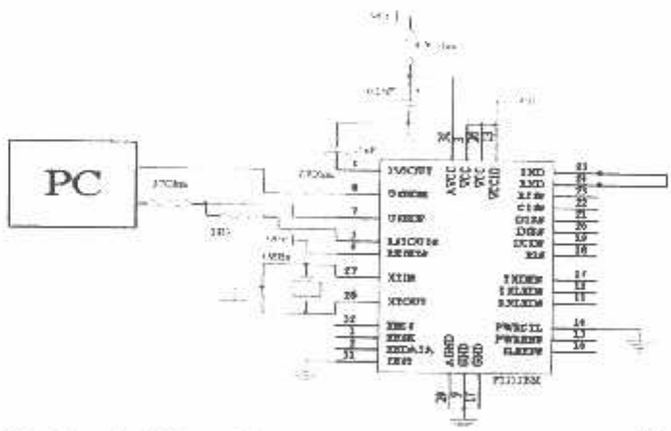
Untuk mengetahui fungsi pin Tx (pengiriman data) dan Rx(penerimaan data) pada rangkaian FT-232BM yang terdapat pada modul rangkaian konverter serial to USB, dilakukan dengan cara mengirimkan data dari PC ke FT232BM melalui USB, dan pada pin24 dan 25 (Rx dan Tx) dijumper, sehingga data yang akan dikirimkan oleh PC ke Mikrokontroller akan dibalikkan lagi ke PC.

Peralatan yang digunakan pada pengujian ini adalah senagai berikut:

- Rangkaian FT232BM
- Konektor USB
- Jumper

a. Langkah-langkah Pengujian

1.Merangkai rangkaian driver seperti pada gambar dibawah ini:



Gambar 4 -1 Rangkaian Pengujian *Serial Interface* [8]

2. Pada gambar di atas, output FT232BM (Pout T_{XD}) dihubungkan dengan input (Pin R_{XD}), dengan demikian semua data yang dikirim melalui PC akan diumpan-balikkan ke PC lagi.

b. Analisa

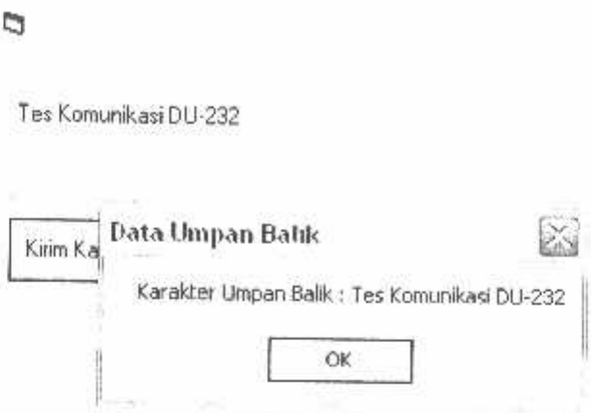
Pada pengujian ini menggunakan beberapa parameter pada sistem jabat tangannya, yaitu:

1. *Port*, *port* yang digunakan adalah *virtual com* dengan alamat yang terdapat pada *COM8*.
2. *Baudrate*, *baudrate* yang digunakan 9600 *bit per second*.
3. *Data bits*, *data bits* yang digunakan memiliki ukuran panjang *frame* 8 bit.
4. *Parity*, tidak menggunakan bit *parity*.
5. *Flow control*, tidak menggunakan *flow control*.

Berikut adalah *screen shot* dari setting port pada pengujian ini:

Gambar 4 -2
Setting Port pada PC

Data yang dikirim Tes Komunikasi DU-232 maka data yang diterima oleh PC pun sama Tes Komunikasi DU-232.



Gambar 4 -3
Pengujian *Serial Data* pada DU-232

4.1.2 Analisa Perangkat Penyamaran Data

a. Tujuan

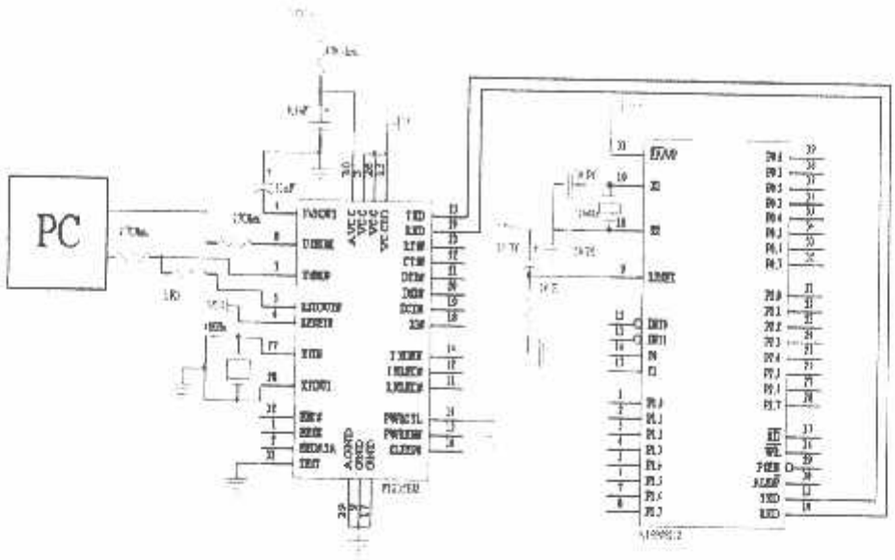
Untuk mengetahui sistem jabat tangan PC dengan mikrokontroler apakah dapat berjalan dengan baik. Data akan dikirimkan PC kepada mikrokontroler melalui USB konverter.

b. Peralatan yang digunakan

- Rangkaian FT232BM
- Mikrokontroler
- PC

c. Langkah-langkah Pengujian

1.Merangkai rangkaian driver seperti pada gambar dibawah ini:



Gambar 4–4 Rangkaian Pengujian Perangkat Kode Lisensi [5]

2. Pada gambar di atas, output FT232BM (Pin T_{XD}) dihubungkan dengan Rx mikrokontroler dan Rx FT232BM di hubungkan dengan pin Tx mikrokontroler, dengan demikian semua data yang dikirim melalui PC akan diterima oleh mikrokontroler atau sebaliknya.

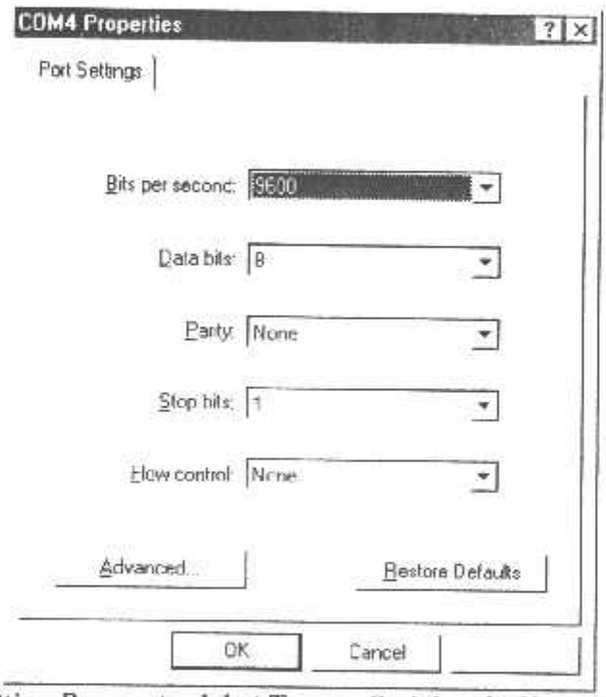
d. Analisa

Pada pengujian ini perangkat lunak yang digunakan adalah program aplikasi *Hyper Terminal* pada sistem operasi windows. Pada pengujian ini menggunakan beberapa parameter pada sistem jabat tangannya, yaitu:

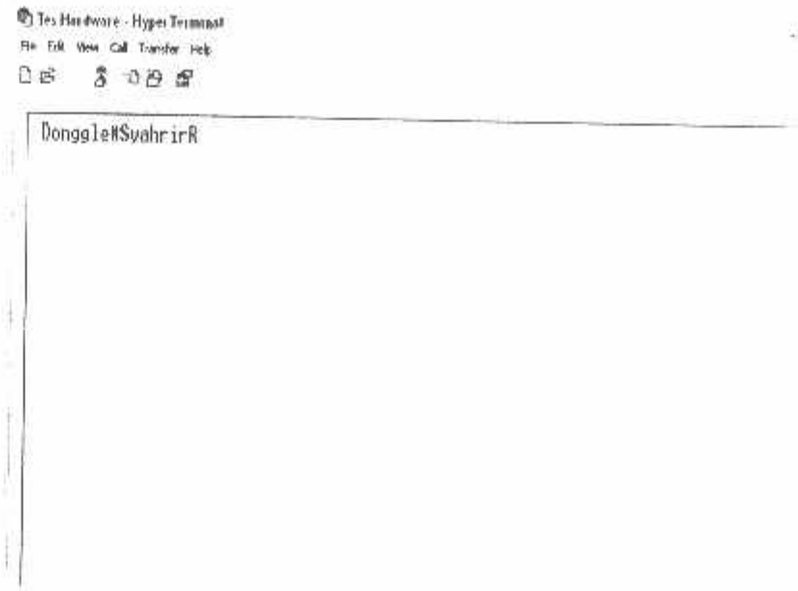
1. *Port*, *port* yang digunakan adalah *virtual com* dengan alamat yang terdapat pada *COM7*.
2. *Baudrate*, *baudrate* yang digunakan 9600 *bit per second*.
3. *Data bits*, *data bits* yang digunakan memiliki ukuran panjang *frame* 8 bit.
4. *Parity*, tidak menggunakan bit *parity*.
5. *Flow control*, tidak menggunakan *flow control*.

Berikut adalah *screen shot* dari *setting port* pada pengujian ini:

Gambar 4 -5 *Setting* Port Serial pada *HyperTerminal*



Gambar 4 -6 *Setting Parameter* Jabat Tangan Serial pada *HyperTerminal*



Gambar 4 -7 Tampilan Data *HyperTerminal*

Pada gambar diatas mikrokontroler akan memberikan data berupa ID hardware yang digunakan sebagai *primary key* jika PC akan mengirimkan karakter 'C' melalui USB.

4.1.3 Analisa Penyamaran Kunci Dua Arah

Analisa ini dilakukan untuk melihat sejauh mana tingkat kesamaran dan keakuratan data yang telah dilakukan proses *enkripsi* dengan melakukan beberapa percobaan menggunakan *primary key* dengan *private key* dalamhal ini proses *enkripsi* yang dilakukan adalah penggunaan logika XOR untuk menghasilkan kunci dus arah.

4.1.3.1 Analisa XOR Satu *Primary key* ke Banyak *Private key*

Pada proses ini tiap satu byte (8 bit) dari *primary key* di-XOR dengan setiap karakter dari *private key* sampai semua karakter pada *private key* terpenuhi. Dapat dikatakan, pada tahap ini penyamaran kunci dilakukan dengan prinsip dasar *one to many*.

- *Primary key* : "SyahrirR",dengan biner dan ordinal dari kunci adalah sebagai berikut:

Char	S	y	a	h	r	i	r	R
ASCII	83	121	97	104	114	105	114	82

Maka nilai biner pada ASCII tersebut akan dirangkaiakan menjadi,

01010011 | 01111001 | 01100001 | 01101000 | 01110010 | 01101001 | 01110010 | 01010010

- *Private key* : "Kunci", dengan biner dan ordinal dari kunci adalah sebagai berikut:

Char	K	u	N	c	i
ASCII	75	117	110	99	105

Dengan nilai biner pada ASCII, yaitu :

01001011 | 01110101 | 01101110 | 01100011 | 01101001

maka akan menghasilkan karakter kunci acak (*final key*), yag akan dijabarkan berikut ini :

4.1.3.1.1 Karakter pertama *primary key* (S)

Pada proses pertama ini akan menghasilkan suatu karakter yang dihasilkan dalam beberapa kali proses XOR antara karakter pertama dari *primary key* dengan semua karakter dari *private key*.

Karakter "S" XOR karakter "K"

01010011 Xor 01001011 = 00011000

hasil dari XOR di atas XOR dengan karakter "u"

00011000 XOR 01110101 = 001101101

hasil XOR di atas XOR lagi dengan karakter "n"

001101101 XOR 01101110 = 00000011

hasil dari XOR di atas Xor dengan karakter "c"

00000011 XOR 01100011 = 01100000

Hasil dari proses di atas adalah karakter 01100000

4.1.3.1.2 Karakter kedua dari *primary key* (y)

Pada proses ini akan menghasilkan suatu karakter yang dihasilkan dalam beberapa kali proses XOR antara karakter kedua dari *primary key* dengan semua karakter dari *private key*.

Karakter "y" XOR dengan karakter "K"

01111001 XOR 01001011 = 00110011

hasil dari XOR di atas Xor dengan karakter "u"

00110011 XOR 01110101 = 01000110

hasil dari XOR di atas XOR dengan karakter "n"

01000110 XOR 01101110 = 00101000

hasil dari XOR di atas XOR dengan karakter "c"

00101000 XOR 01100011 = 01001011

hasil dari XOR di atas XOR dengan karakter "i"

01001011 XOR 01101001 = 00100010

Proses di atas menghasilkan karakter 00100010

4.1.3.1.3 Karakter ketiga dari *primary key* (a)

Proses ini akan menghasilkan suatu karakter yang dihasilkan dalam beberapa kali proses XOR antara karakter "a" dari *primary key* dengan semua karakter dari *private key*

Karakter "a" XOR dengan karakter "K"

01100001 XOR 01001011 = 00101010

hasil dari XOR di atas XOR dengan karakter "u"

00101010 XOR 01110101 = 01011111

hasil dari XOR di atas XOR dengan karakter "n"

01011111 XOR 01101110 = 00110001

hasil dari XOR di atas XOR dengan karakter "c"

00110001 XOR 01100011 = 01010010

hasil dari XOR di atas XOR dengan karakter "i"

01010010 XOR 01101001 = 00111011

Karakter yang dihasilkan adalah 00111011

4.1.3.1.4 Karakter keempat dari *primary key* (h)

Proses ini akan menghasilkan suatu karakter yang dihasilkan dalam beberapa kali proses XOR antara karakter "h" dari *primary key* dengan semua karakter dari *private key*

Karakter "h" XOR dengan karakter "K"

01101000 XOR 01001011 = 00100011

hasil dari XOR di atas XOR dengan karakter "u"

00100011 XOR 01110101 = 01010110

hasil dari XOR di atas XOR dengan karakter "n"

01010110 XOR 01101110 = 00111000

hasil dari XOR di atas XOR dengan karakter "c"

00111000 XOR 01100011 = 01011011

hasil dari XOR di atas XOR dengan karakter "i"

01011011 XOR 01101001 = 00110010

Hasil dari proses XOR di atas adalah karakter 00110010

4.1.3.1.5 Karakter kelima *primary key* (r)

Proses ini akan menghasilkan suatu karakter yang dihasilkan dalam beberapa kali proses XOR antara karakter "r" dari *primary key* dengan semua karakter dari *private key*

Karakter "r" XOR dengan karakter "K"

01110010 XOR 01001011 = 00111001

hasil dari XOR di atas XOR dengan karakter "u"

00111001 XOR 01110101 = 01001100

hasil dari XOR di atas dilakukan XOR lagi dengan karakter "n"

01001100 XOR 01101110 = 00100010

hasil dari XOR di atas XOR dengan karakter "c"

00100010 XOR 01100011 = 01000001

hasil dari XOR di atas XOR dengan karakter "i"

01000001 XOR 01101001 = 00101000

00101000 adalah hasil dari beberapa kali proses XOR di atas

4.1.3.1.6 Karakter keenam dari *primary key* (i)

Sama seperti proses – proses di atas, proses ini akan menghasilkan suatu karakter yang dihasilkan dalam beberapa kali proses XOR antara karakter "r" dari *primary key* dengan semua karakter dari *private key*

Karakter "i" XOR dengan karakter "K"

01101001 XOR 01001011 = 00100010

hasil dari XOR di atas XOR dengan karakter "u"

00100010 XOR 01110101 = 01010111

hasil dari XOR di atas XOR dengan karakter "n"

01010111 XOR 01101110 = 00111001

hasil dari XOR di atas XOR dengan karakter "c"

00111001 XOR 01100011 = 01011010

hasil dari XOR di atas XOR dengan karakter "i"

01011010 XOR 01101001 = 00110011

Result dari proses di atas adalah karakter 00110011

4.1.3.1.7 Karakter kedelapan dari *primary key* (R)

Sama seperti proses – proses di atas, proses terakhir ini akan menghasilkan suatu karakter yang dihasilkan dalam beberapa kali proses XOR antara karakter "R" dari *primary key* dengan semua karakter dari *private key*

Karakter "R" XOR dengan karakter "K"

01010010 XOR 01001011 = 00011001

hasil dari XOR di atas XOR dengan karakter "u"

00011001 XOR 01110101 = 01101100

hasil dari XOR di atas XOR dengan karakter "n"

01101100 XOR 01101110 = 00000010

hasil dari XOR di atas XOR dengan karakter "c"

00000010 XOR 01100011 = 01100001

hasil dari XOR di atas XOR dengan karakter "i"

01100001 XOR 01101001 = 00001000

Hasil dari beberapa kali prose XOR di atas adalah karakter 00001000

Final result dari semua proses kunci dua arah di atas adalah

01100000 | 00100010 | 00111011 | 00110010 | 00101000 | 00110011 |

00101000 | 00001000

Char	#	;	2	(	□	(	□	Z
HEXA	60	22	3B	32	28	33	28	8

4.2. Simulasi Program Aplikasi.

Sub bab ini akan membahas tentang proses yang terjadi dari sisi *software*, dimulai dari tampilan program aplikasi sampai dengan proses penyamaran data (*encode*) dan pengembalian data yang telah disamarkan (*decode*) dari folder tempat data disimpan.

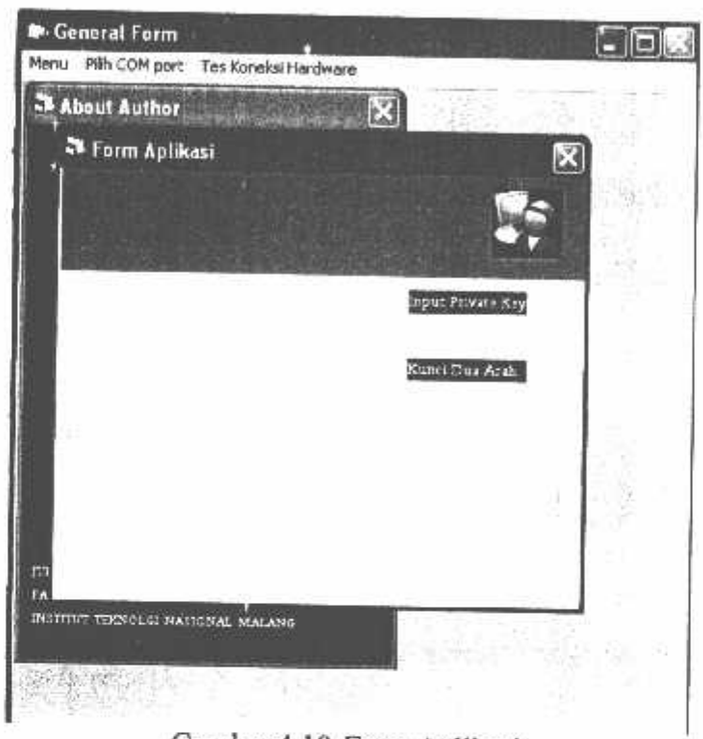
1. Tampilan awal dari program aplikasi adalah *form About Author* yang berada dalam MDIForm General. Visualisasi dari *form* ini adalah

Gambar 4.8 *Form About Author*

2. Setelah *form About Author* (click OK), atau pada file Menu dipilih sub menu form Aplikasi maka sebelum program aplikasi menampilkan *form* utama dari *project* ini akan ada peringtan pada *user* bahwa *hardware* belum terkoneksi

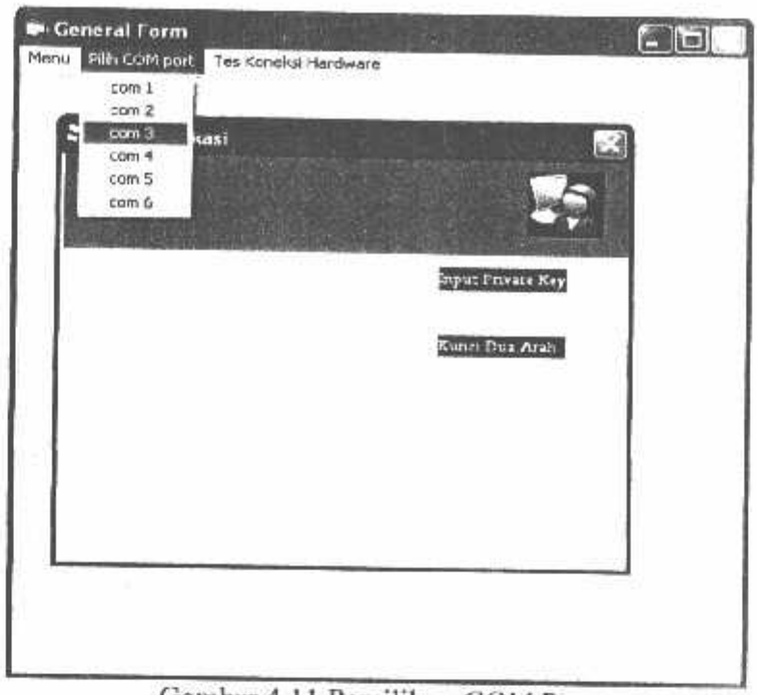
Gambar 4.9 Peringatan koneksi *hardware*

3. Visualisasi dari *form* Aplikasi adalah sebagai berikut:

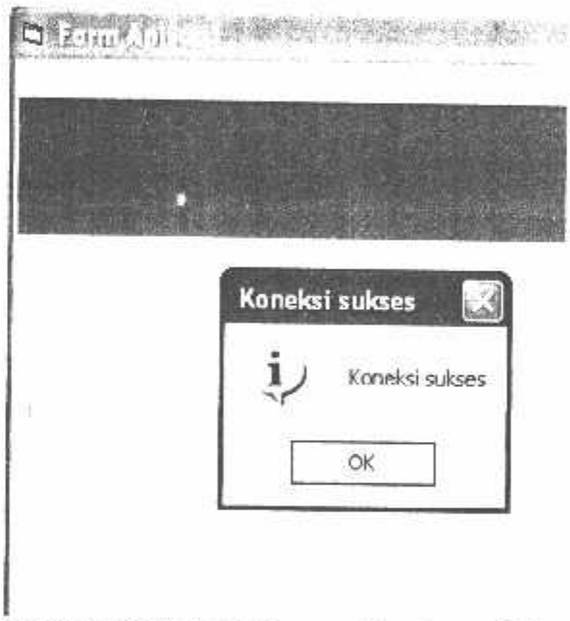


Gambar 4.10 *Form* Aplikasi

4. Setelah *form* aplikasi ditampilkan, maka proses koneksi *hardware* dari sisi *software* dapat dilakukan. Langkah – langkah yang dilakukan adalah *Pilih COM Port*- setelah itu lakukan tes koneksi pada menu *Tes Koneksi Hardware*



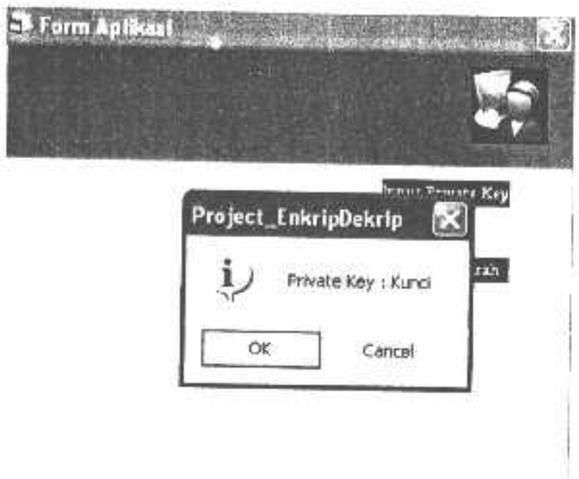
Gambar 4.11 Pemilihan *COM Port*



Gambar 4.12 Koneksi Dengan *Hardware* Sukses

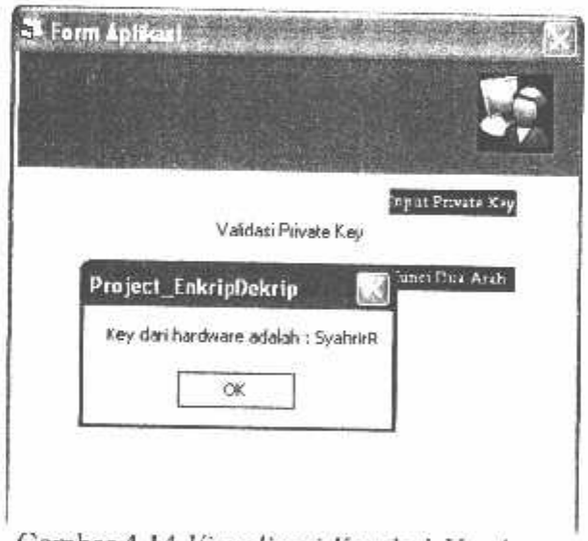
5. Setelah proses konensi sukses dilakukan, maka secara otomatis kursor akan berada dalam teks box (input box *private key*). Proses validasi akan dilakukan, yang nantinya akan menampilkan penegasan karakter yang digunakan sebagai *private key.*

Tampilan dari proses ini adalah:



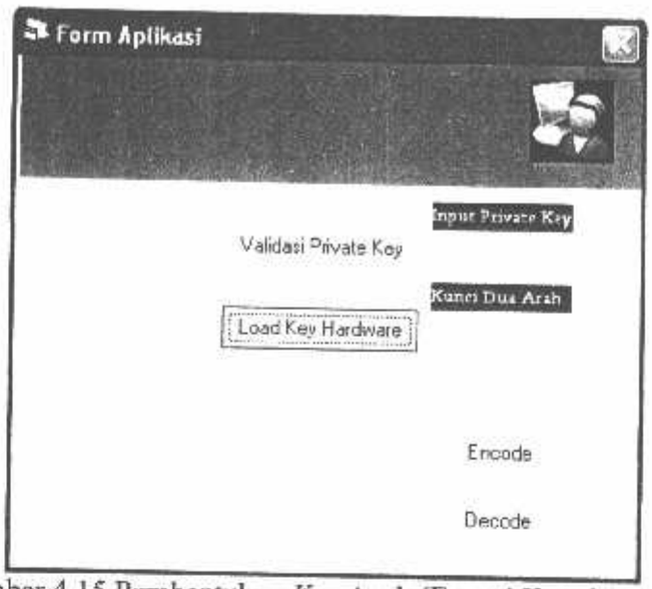
Gambar 4.13 Tampilan *Validasi Private key*

6. Setelah *private key* tersimpan (dalam karakter asterik "*") maka proses *download primary key* dari MCU AT89S8252 dapat dilakukan. Karakter "SyahrirR" yang di-*download* dari *hardware* dengan tampilan sebagai berikut :



Gambar 4.14 *Visualisasi Key* dari *Hardware*

7. Maka proses pembentukan kunci dua arah pun dapat dilakukan oleh program aplikasi dengan pembentukan karakter yang dapat dilihat pada *teks box key* (dengan *Label* "Kunci Dua Arah")acak

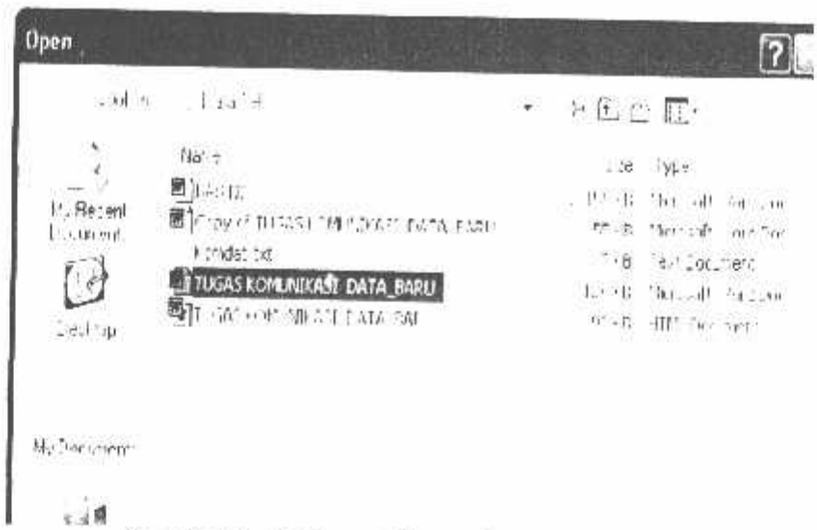


Gambar 4.15 Pembentukan *Key* Acak (Fungsi Kunci 2 Arah)

Pada dasarnya proses yang dilakukan oleh *software* maupun *hardware* sampai pada proses ini adalah sama. Perbedaan yang *significant* yang terjadi dari sisi *software* maupun *hardware* adalah pada proses yang dilakukan antara proses *encode* maupun *decode*.

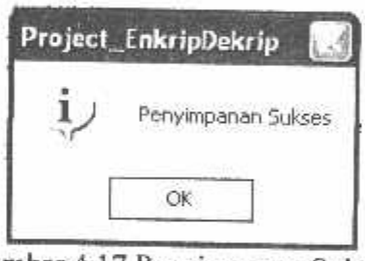
4.2.1. Proses *encode* data.

Dalam proses ini, *software* akan mengacak susunan tiap bit dari *byte* karakter yang disamarkan. Dengan tetap memperhatikan aturan main dari metode kriptografi yaitu kapasitas data sebelum maupun sesudah disamarkan tetap sama.



Gambar 4.16 *Open File* untuk proses *encode*

Proses *enkripsi* pun dapat dilkukan. Setelah proses selesai dilakukan, maka *software* akan memberikan *feedback* kepada *user* berupa suatu pesan penyimpanan data hasil penyamaran sukses dilakukan.



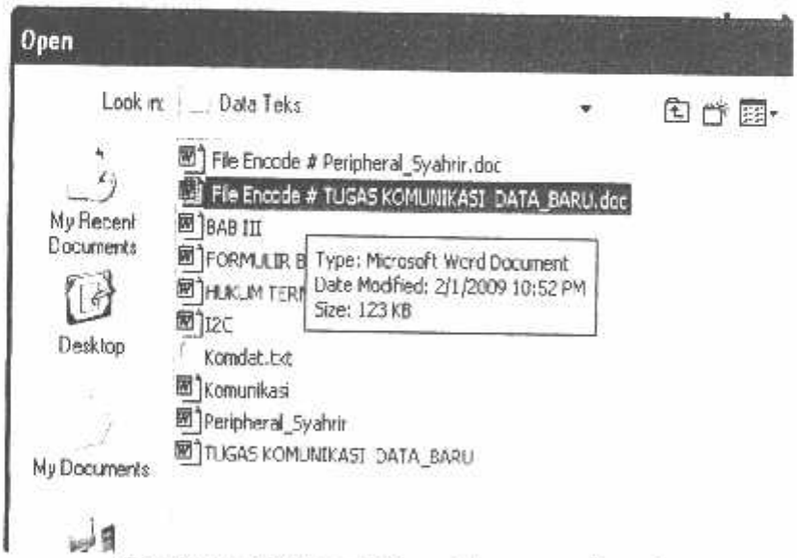
Gambar 4.17 Penyimpanan Sukses

Sedikit tampilan dari hasil proses penyamaran data yang telah dilakukan adalah:

Gambar 4.18 Data Hasil *Encode*

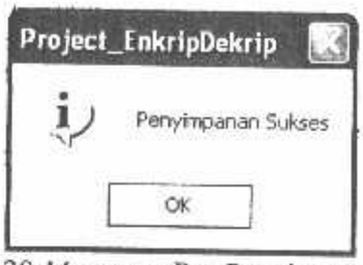
4.2.2. Proses de*code* data.

Dalam proses ini, *software* akan mengembalikan susunan tiap bit dari byte karakter yang telah tersamarkan. Dengan tetap memperhatikan aturan main dari metode kriptografi yaitu kapasitas data sebelum maupun sesudah disamarkan tetap sama.



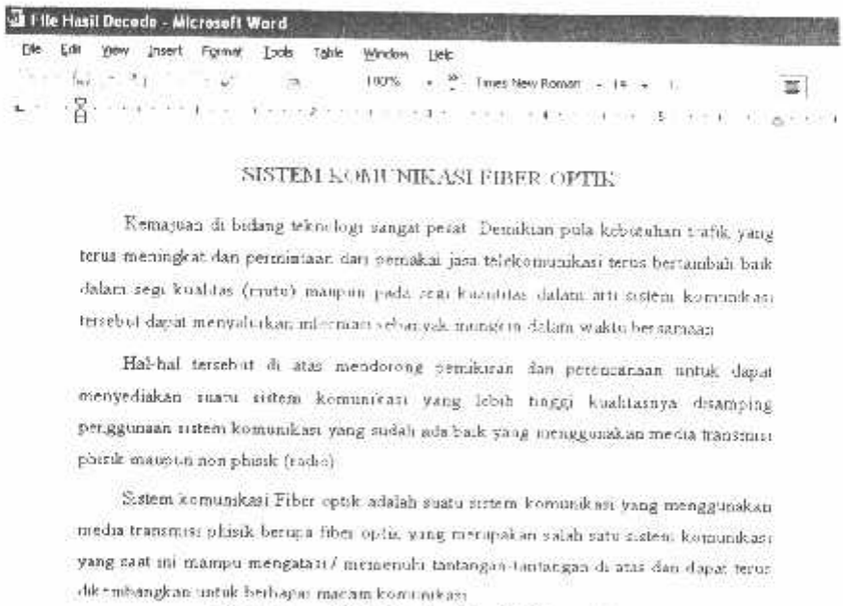
Gambar 4.19 *Open File* untuk proses *decode*

Proses *dekripsi* pun dapat dilkukan. Setelah proses selesai dilakukan, maka *software* akan memberikan *feedback* kepada *user* berupa suatu pesan penyimpanan data hasil penyamaran sukses dilakukan.



Gambar 4.20 *Messages Box* Penyimpanan Sukses

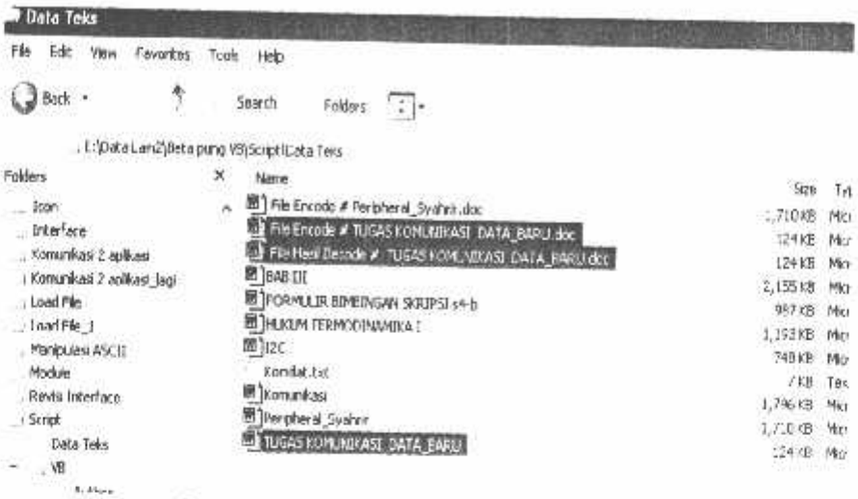
Setelah semua proses dilakukan dan *software* pun telah memberikan isyarat kepada *user* bahwa penyimpanan sukses dilakukan, maka kita dapat melihat sedikit tampilan dari hasil proses pengembalian data samaran yang telah dilakukan adalah:



Gambar 4.21 Data Hasil *Decode*

Maka proses pengembalian data (*decode*) telah selesai dilakukan dengan tidak mengurangi 1 bit pun dari data sebelum *encode*,setelah *encode*,maupun setelah proses *decode*.

Sesuai dengan aturan main dari proses penyamaran data dengan menggunakan metode Kriptografi terkait dengan perihal kapasitas file asli, hasil enkripsi, dan dekripsi yaitu harus sama besar, yang divisualisasikan pada *shoot screen* berikut ini:



Gambar 4.22 Kapasitas data sama besar

Sampai pada tahap ini maka suatu proses secara menyeluruh yang dimulai dari suatu data asli yang kemudian di-*encode*, lalu dikembalikan kebentuk semula melali proses *decode* dimana kedua proses tersebut menggunakan metode *Chriptography Caesar* sebagai metode penyamaran data telah selesai dilakukan.

BAB V
KESIMPULAN DAN SARAN

Pada bab ini akan dijelaskan beberapa poin kesimpulan dan saran yang di dapat dari proses perancangan sistem pengamanan data word menggunakan media perangkat keras menggunakan metode kriptografi *Caesar* ini.

5.1. Kesimpulan

Setelah melewati beberapa proses pada tahap perencanaan sampai dengan tahap perancangan sistem ini, maka dapat ditarik beberapa kesimpulan sebagai berikut:

1. Berdasarkan pengujian sistem secara keseluruhan maka dapat dikatakan sistem dapat berjalan dengan baik.
2. Berdasarkan hasil pengujian modul *konverter serial to USB* maka dapat dikatakan modul dapat berfungsi dengan baik.
3. Pada sisi software untuk hasil metode yang digunakan dalam proses penyamaran data adalah Kriptografi *Caesar* yang telah dimodifikasi dengan penambahan logika XOR.
4. Terbukti bahwa *chiperteks* dengan penambahan logika XOR mempunyai perbedaan karakter yang sangat *significant* dengan *plainteks*, jika dibandingkan dengan penggunaan Kriptografi *Caesar* secara murni. Dikarenakan adanya *over security* yang dilakukan pada proses penyamaran yaitu proses *Caesar* yang dilanjutkan dengan XOR *final key* dengan data yang telah mengalami *Caesar encode*
5. Pada proses *encode* maupun *decode* dengan data berukuran 128kB, dengan penggunaan *memory* komputer sebesar 64MB maka proses dapat berjalan dalam *range* waktu 13 *second*. *Range* waktu pada proses tersebut dapat lebih cepat ataupun sebaliknya, tergantung dari *memory* PC yang digunakan.

6. Semakin besar kapasitas *memory* dari sebuah PC, maka semakin cepat proses yang terjadi baik itu *encode* maupun *decode* dari suatu data.

5.2. Saran

Setelah melihat tingkat hasil dari perancangan sistem ini maka dapat disebutkan beberapa saran untuk pengembangan lebih lanjut pada sistem ini:

1. Untuk pengembangan lebih lanjut diharapkan dapat dirancang suatu algoritma khusus yang lebih kompleks pada tahap *verifikasi* kode lisensi.
2. Adanya pengembangan metode dari metode yang digunakan merupakan suatu hal yang diharapkan untuk meningkatkan *security* dari suatu data.
3. Untuk pengembangan lebih lanjut diharapkan *hardware* yang dirancang dapat berfungsi sebagai *plug-in* agar sistem dapat bersifat *fleksible*.

DAFTAR PUSTAKA

[1] Wasito S, 1990, *Vandemekum Elektronika, Edisi Kedua*, PT. Gramedia Pustaka Utama, Jakarta.

[2] Antony Pranata, 2002, *Aplikasi Sains dan Teknik menggunakan Visual Basic 6.0*, Edisi 4, Penerbit: Andi, Yogyakarta.

[3] Menejes, P. Van Oorscot, dan S. Vanstone. 1996. *Handbook Of Applied Cryptography* : CRC press.

[4] Retna Prasetia & Catur Edi Wibowo, *Interfacing Port Paralel & Port Serial komputer dengan Visual Basic 6.0*, Penerbit : Andi Yogyakarta.

[5] Didin Wahyudin, *Belajar Mudah Mikrokontroler AT89S52*,Penerbit: Andi, Yogyakarta.

[6] Dwi Sutadi, *I/O Bus & Mother Board*, Andi Yogyakarta, Yogyakarta, 2002

[7] Yusuf K, *Kriptografi Keamanan Internet dan Jaringan Komunikasi*, Informatika Bandung, Cetakan pertama, Bandung, 2004

[8] FTDI, *FT232BM Data Sheet*, 2002

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

BERITA ACARA UJIAN SKRIPSI
FAKULTAS TEKNOLOGI INDUSTRI

Nama Mahasiswa	:	Syahrir Ramadhan
NIM	:	03.17.075
Jurusan	:	Teknik Elektro S-1
Konsentrasi	:	Teknik Elektronika
Judul Skripsi	:	Perancangan dan Pembuatan Sistem Pengaman Data *Word* Dengan Kriptografi *Caesar* Menggunakan Mikrokontroler AT89S8252

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

Hari	:	Rabu
Tanggal	:	18 Maret 2009
Dengan Nilai	:	83,2 (A)

Panitia Ujian Skripsi

Ketua

Ir.H. Sidik Noertjahjono, MT
NIP.Y. 102 870 0163

Sekretaris

Ir. F. Yudi Limpraptono, MT
NIP.Y. 103 950 0274

Anggota Penguji

Penguji I

I Komang Somawirata, MT
NIP.P. 103 010 0361

Penguji II

Ir.Mimien Mustikawati, MT
NIP. P. 103. 000 0352

INSTITUT TEKNOLOGI NASIONAL
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

LEMBAR PERBAIKAN SKRIPSI

Nama Mahasiswa : Syahrir Ramadhan
NIM : 03.17.075
Jurusan : Teknik Elektro S-1
Konsentrasi : Teknik Elektronika

No.	Materi Perbaikan	Paraf
1.	Pertajam penjelasan di dalam Abstraksi	
2.	Adanya subroutin pada general flowchart sebagai penjelasan adanya proses dalam routin tersebut	
3.	Untuk setiap subroutin flowchart diakhiri dengan perintah return sebagai penjelasan adanya proses dalam routin setelah subroutin tersebut	
4.	Melengkapi kesimpulan dari hasil pengujian	

Telah Diperiksa dan Disetujui :

Dosen Penguji I

I Komang Somawirata, ST, MT
NIP. P. 103 010 0361

Mengetahui,

Pembimbing I

Ir. Yusuf Ismail Nakhoda, MT
NIP. Y. 101 880 0189

Pembimbing II

M. Ashar, ST, MT
NIP. Y. 103 050 0408

INSTITUT TEKNOLOGI NASIONAL
FAKULTAS TEKNOLOGI INDUSTRI
JURUSAN TEKNIK ELEKTRO

LEMBAR PERBAIKAN SKRIPSI

Nama Mahasiswa	:	Syahrir Ramadhan
NIM	:	03.17.075
Jurusan	:	Teknik Elektro S-1
Konsentrasi	:	Teknik Elektronika

No.	Materi Perbaikan	Paraf
1.	Penambahan kata kunci pada bagian Abstraksi	

Telah Diperiksa dan Disetujui :

Dosen Penguji II

Ir. TH. Mimien Mustikawati, MT
NIP. P. 103 000 0352

Mengetahui,

Pembimbing I

Ir. Yusuf Ismail Nakhoda, MT
NIP. Y. 101 880 0089

Pembimbing II

M. Ashar, ST, MT
NIP. Y. 103 050 0408

Features

- Compatible with MCS-51™ Products
- 8K Bytes of In-System Reprogrammable Downloadable Flash Memory
  - SPI Serial Interface for Program Downloading
  - Endurance: 1,000 Write/Erase Cycles
- 2K Bytes EEPROM
  - Endurance: 100,000 Write/Erase Cycles
- 4V to 6V Operating Range
- Fully Static Operation: 0 Hz to 24 MHz
- Three-level Program Memory Lock
- 256 x 8-bit Internal RAM
- 32 Programmable I/O Lines
- Three 16-bit Timer/Counters
- Nine Interrupt Sources
- Programmable UART Serial Channel
- SPI Serial Interface
- Low-power Idle and Power-down Modes
- Interrupt Recovery From Power-down
- Programmable Watchdog Timer
- Dual Data Pointer
- Power-off Flag

Description

The AT89S8252 is a low-power, high-performance CMOS 8-bit microcomputer with 8K bytes of downloadable Flash programmable and erasable read only memory and 2K bytes of EEPROM. The device is manufactured using Atmel's high-density nonvolatile memory technology and is compatible with the industry-standard 80C51 instruction set and pinout. The on-chip downloadable Flash allows the program memory to be reprogrammed in-system through an SPI serial interface or by a conventional nonvolatile memory programmer. By combining a versatile 8-bit CPU with downloadable Flash on a monolithic chip, the Atmel AT89S8252 is a powerful microcomputer which provides a highly-flexible and cost-effective solution to many embedded control applications.

The AT89S8252 provides the following standard features: 8K bytes of downloadable Flash, 2K bytes of EEPROM, 256 bytes of RAM, 32 I/O lines, programmable watchdog timer, two data pointers, three 16-bit timer/counters, a six-vector two-level interrupt architecture, a full duplex serial port, on-chip oscillator, and clock circuitry. In addition, the AT89S8252 is designed with static logic for operation down to zero frequency and supports two software selectable power saving modes. The Idle Mode stops the CPU while allowing the RAM, timer/counters, serial port, and interrupt system to continue functioning. The Power-down mode saves the RAM contents but freezes the oscillator, disabling all other chip functions until the next interrupt or hardware reset.

The downloadable Flash can be changed a single byte at a time and is accessible through the SPI serial interface. Holding RESET active forces the SPI bus into a serial programming interface and allows the program memory to be written to or read from unless Lock Bit 2 has been activated.

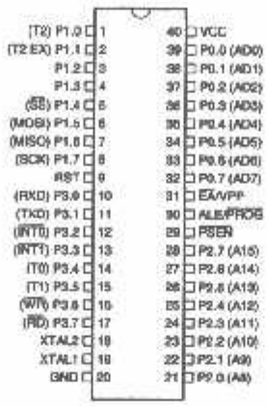
8-bit Microcontroller with 8K Bytes Flash

AT89S8252

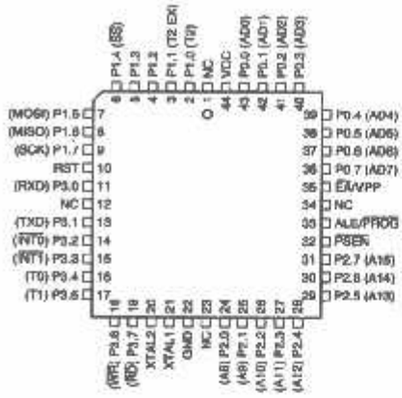
Rev. 0401E–02/00

Pin Configurations

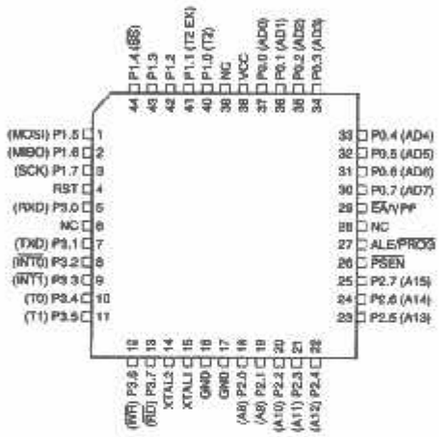
PDIP



PLCC



PQFP/TQFP



Pin Description

VCC

Supply voltage.

GND

Ground.

Port 0

Port 0 is an 8-bit open drain bbi-didirectional 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.

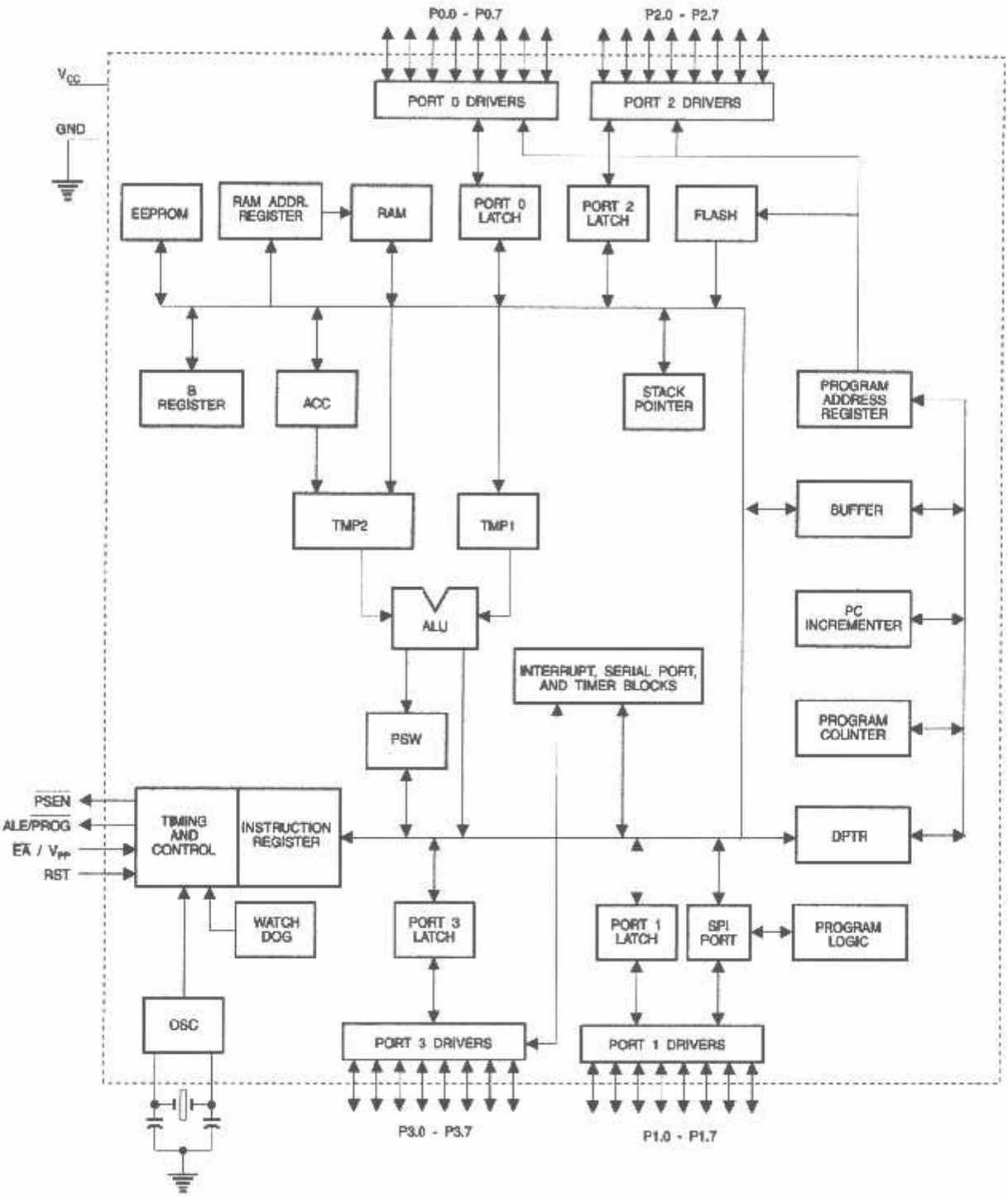
Port 0 can also be configured to be the multiplexed low-order address/data bus during accesses to external program and data memory. In this mode, P0 has internal pullups.

Port 0 also receives the code bytes during Flash programming and outputs the code bytes during program verification. External pullups are required during program verification.

Port 1

Port 1 is an 8-bit bi-directional I/O port with internal pullups. 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 pullups 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 pullups.

Block Diagram



Some Port 1 pins provide additional functions. P1.0 and P1.1 can be configured to be the timer/counter 2 external count input (P1.0/T2) and the timer/counter 2 trigger input (P1.1/T2EX), respectively.

Pin Description

Furthermore, P1.4, P1.5, P1.6, and P1.7 can be configured as the SPI slave port select, data input/output and shift clock input/output pins as shown in the following table.

Port Pin	Alternate Functions
P1.0	T2 (external count input to Timer/Counter 2), clock-out
P1.1	T2EX (Timer/Counter 2 capture/reload trigger and direction control)
P1.4	$\overline{SS}$ (Slave port select input)
P1.5	MOSI (Master data output, slave data input pin for SPI channel)
P1.6	MISO (Master data input, slave data output pin for SPI channel)
P1.7	SCK (Master clock output, slave clock input pin for SPI channel)

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

Port 2

Port 2 is an 8-bit bi-directional I/O port with internal pullups. 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 pullups 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 pullups.

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 pullups when emitting 1s. During accesses to external data memory that use 8-bit addresses (MOVX @ RI), Port 2 emits the contents of the P2 Special Function Register.

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

Port 3

Port 3 is an 8 bit bi-directional I/O port with internal pullups. 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 pullups 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 pullups.

Port 3 also serves the functions of various special features of the AT89S8252, as shown in the following table.

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

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

RST

Reset input. A high on this pin for two machine cycles while the oscillator is running resets the device.

ALE/$\overline{PROG}$

Address Latch Enable 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.

$\overline{PSEN}$

Program Store Enable is the read strobe to external program memory.

When the AT89S8252 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.

$\overline{EA}$/VPP

External Access Enable. $\overline{EA}$ must be strapped to GND in order to enable the device to fetch code from external pro-

gram 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 when 12-volt programming is selected.

XTAL1

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

XTAL2

Output from the inverting oscillator amplifier.

Table 1. AT89S8252 SFR Map and Reset Values

0F8H									0FFH
0F0H	B 00000000								0F7H
0E8H									0EFH
0E0H	ACC 00000000								0E7H
0D8H									0DFH
0D0H	PSW 00000000					SPCR 000001XX			0D7H
0C8H	T2CON 00000000	T2MOD XXXXXX00	RCAP2L 00000000	RCAP2H 00000000	TL2 00000000	TH2 00000000			0CFH
0C0H									0C7H
0B8H	IP XX000000								0BFH
0B0H	P3 11111111								0B7H
0A8H	IE 0X000000		SPSR 00XXXXXX						0AFH
0A0H	P2 11111111								0A7H
98H	SCON 00000000	SBUF XXXXXXXX							9FH
90H	P1 11111111						WMCON 00000010		97H
88H	TCON 00000000	TMOD 00000000	TL0 00000000	TL1 00000000	TH0 00000000	TH1 00000000			8FH
80H	P0 11111111	SP 00000111	DP0L 00000000	DP0H 00000000	DP1L 00000000	DP1H 00000000	SPDR XXXXXXXX	PCON 0XXX0000	87H

Special Function Registers

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.

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

Timer 2 Registers Control and status bits are contained in registers T2CON (shown in Table 2) and T2MOD (shown in Table 9) for Timer 2. The register pair (RCAP2H, RCAP2L) are the Capture/Reload registers for Timer 2 in 16 bit capture mode or 16-bit auto-reload mode.

Table 2. T2CON—Timer/Counter 2 Control Register

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

Bit Addressable

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

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

Watchdog and Memory Control Register The WMCON register contains control bits for the Watchdog Timer (shown in Table 3). The EEMEN and EEMWE bits are used to select the 2K bytes on-chip EEPROM, and to enable byte-write. The DPS bit selects one of two DPTR registers available.

Table 3. WMCON—Watchdog and Memory Control Register

WMCON Address = 96H — Reset Value = 0000 0010B

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

Symbol	Function
PS2 PS1 PS0	Prescaler Bits for the Watchdog Timer. When all three bits are set to "0", the watchdog timer has a nominal period of 16 ms. When all three bits are set to "1", the nominal period is 2048 ms.
EEMWE	EEPROM Data Memory Write Enable Bit. Set this bit to "1" before initiating byte write to on-chip EEPROM with the MOVX instruction. User software should set this bit to "0" after EEPROM write is completed.
EEMEN	Internal EEPROM Access Enable. When EEMEN = 1, the MOVX instruction with DPTR will access on-chip EEPROM instead of external data memory. When EEMEN = 0, MOVX with DPTR accesses external data memory.
DPS	Data Pointer Register Select. DPS = 0 selects the first bank of Data Pointer Register, DP0, and DPS = 1 selects the second bank, DP1
WDTRST RDY/$\overline{BSY}$	Watchdog Timer Reset and EEPROM Ready/$\overline{Busy}$ Flag. Each time this bit is set to "1" by user software, a pulse is generated to reset the watchdog timer. The WDTRST bit is then automatically reset to "0" in the next instruction cycle. The WDTRST bit is Write-Only. This bit also serves as the RDY/$\overline{BSY}$ flag in a Read-Only mode during EEPROM write. RDY/$\overline{BSY}$ = 1 means that the EEPROM is ready to be programmed. While programming operations are being executed, the RDY/$\overline{BSY}$ bit equals "0" and is automatically reset to "1" when programming is completed.
WDTEN	Watchdog Timer Enable Bit. WDTEN = 1 enables the watchdog timer and WDTEN = 0 disables the watchdog timer.

SPI Registers Control and status bits for the Serial Peripheral Interface are contained in registers SPCR (shown in Table 4) and SPSR (shown in Table 5). The SPI data bits are contained in the SPDR register. Writing the SPI data register during serial data transfer sets the Write Collision bit, WCOL, in the SPSR register. The SPDR is double buffered for writing and the values in SPDR are not changed by Reset.

Interrupt Registers The global interrupt enable bit and the individual interrupt enable bits are in the IE register. In addition, the individual interrupt enable bit for the SPI is in the SPCR register. Two priorities can be set for each of the six interrupt sources in the IP register.

Dual Data Pointer Registers To facilitate accessing both internal EEPROM 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 WMCON 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 reset under software control and is not affected by RESET.

Table 4. SPCR—SPI Control Register

SPCR Address = D5H — Reset Value = 0000 01XXB

Bit	7	6	5	4	3	2	1	0
	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0

Symbol	Function
SPIE	SPI Interrupt Enable. This bit, in conjunction with the ES bit in the IE register, enables SPI interrupts: SPIE = 1 and ES = 1 enable SPI interrupts. SPIE = 0 disables SPI interrupts.
SPE	SPI Enable. SPI = 1 enables the SPI channel and connects $\overline{SS}$, MOSI, MISO and SCK to pins P1.4, P1.5, P1.6, and P1.7. SPI = 0 disables the SPI channel.
DORD	Data Order. DORD = 1 selects LSB first data transmission. DORD = 0 selects MSB first data transmission.
MSTR	Master/Slave Select. MSTR = 1 selects Master SPI mode. MSTR = 0 selects Slave SPI mode.
CPOL	Clock Polarity. When CPOL = 1, SCK is high when idle. When CPOL = 0, SCK of the master device is low when not transmitting. Please refer to figure on SPI Clock Phase and Polarity Control.
CPHA	Clock Phase. The CPHA bit together with the CPOL bit controls the clock and data relationship between master and slave. Please refer to figure on SPI Clock Phase and Polarity Control.
SPR0 SPR1	SPI Clock Rate Select. These two bits control the SCK rate of the device configured as master. SPR1 and SPR0 have no effect on the slave. The relationship between SCK and the oscillator frequency, $F_{osc.}$, is as follows: SPR1 SPR0 SCK = $F_{osc.}$ divided by 0 0 4 0 1 16 1 0 64 1 1 128

Table 5. SPSR – SPI Status Register

SPSR Address = AAH — Reset Value = 00XX XXXXB

Bit	7	6	5	4	3	2	1	0
	SPIF	WCOL	–	–	–	–	–	–

Symbol	Function
SPIF	SPI Interrupt Flag. When a serial transfer is complete, the SPIF bit is set and an interrupt is generated if SPIE = 1 and ES = 1. The SPIF bit is cleared by reading the SPI status register with SPIF and WCOL bits set, and then accessing the SPI data register.
WCOL	Write Collision Flag. The WCOL bit is set if the SPI data register is written during a data transfer. During data transfer, the result of reading the SPDR register may be incorrect, and writing to it has no effect. The WCOL bit (and the SPIF bit) are cleared by reading the SPI status register with SPIF and WCOL set, and then accessing the SPI data register.

Table 6. SPDR – SPI Data Register

SPDR Address = 86H — Reset Value = unchanged

Bit	7	6	5	4	3	2	1	0
	SPD7	SPD6	SPD5	SPD4	SPD3	SPD2	SPD1	SPD0

Data Memory – EEPROM and RAM

The AT89S8252 implements 2K bytes of on-chip EEPROM for data storage and 256 bytes of RAM. The upper 128 bytes of RAM occupy a parallel space to the Special Function Registers. That means the upper 128 bytes have the same addresses as the SFR space but are physically separate from SFR space.

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

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

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

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

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

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

The on-chip EEPROM data memory is selected by setting the EEMEN bit in the WMCON register at SFR address location 96H. The EEPROM address range is from 000H to 7FFH. The MOVX instructions are used to access the EEPROM. To access off-chip data memory with the MOVX instructions, the EEMEN bit needs to be set to "0".

The EEMWE bit in the WMCON register needs to be set to "1" before any byte location in the EEPROM can be written. User software should reset EEMWE bit to "0" if no further EEPROM write is required. EEPROM write cycles in the serial programming mode are self-timed and typically take 2.5 ms. The progress of EEPROM write can be monitored by reading the RDY/$\overline{BSY}$ bit (read-only) in SFR WMCON. RDY/$\overline{BSY}$ = 0 means programming is still in progress and RDY/$\overline{BSY}$ = 1 means EEPROM write cycle is completed and another write cycle can be initiated.

In addition, during EEPROM programming, an attempted read from the EEPROM will fetch the byte being written with the MSB complemented. Once the write cycle is completed, true data are valid at all bit locations.

Programmable Watchdog Timer

The programmable Watchdog Timer (WDT) operates from an independent oscillator. The prescaler bits, PS0, PS1 and PS2 in SFR WMCON are used to set the period of the Watchdog Timer from 16 ms to 2048 ms. The available timer periods are shown in the following table and the actual timer periods (at V_{CC} = 5V) are within ±30% of the nominal.

The WDT is disabled by Power-on Reset and during Power-down. It is enabled by setting the WDTEN bit in SFR WMCON (address = 96H). The WDT is reset by setting the WDTRST bit in WMCON. When the WDT times out without being reset or disabled, an internal RST pulse is generated to reset the CPU.

Table 7. Watchdog Timer Period Selection

WDT Prescaler Bits			
PS2	PS1	PS0	Period (nominal)
0	0	0	16 ms
0	0	1	32 ms
0	1	0	64 ms
0	1	1	128 ms
1	0	0	256 ms
1	0	1	512 ms
1	1	0	1024 ms
1	1	1	2048 ms

Timer 0 and 1

Timer 0 and Timer 1 in the AT89S8252 operate the same way as Timer 0 and Timer 1 in the AT89C51, AT89C52 and AT89C55. For further information, see the October 1995 Microcontroller Data Book, page 2-45, section titled, "Timer/Counters."

Timer 2

Timer 2 is a 16 bit Timer/Counter that can operate as either a timer or an event counter. The type of operation is selected by bit C/$\overline{T2}$ in the SFR T2CON (shown in Table 2). Timer 2 has three operating modes: capture, auto-reload (up or down counting), and baud rate generator. The modes are selected by bits in T2CON, as shown in Table 8.

Timer 2 consists of two 8-bit registers, TH2 and TL2. In the Timer function, the TL2 register is incremented every machine cycle. Since a machine cycle consists of 12 oscillator periods, the count rate is 1/12 of the oscillator frequency.

In the Counter function, the register is incremented in response to a 1-to-0 transition at its corresponding external input pin, T2. In this function, the external input is sampled during S5P2 of every machine cycle. When the samples show a high in one cycle and a low in the next cycle, the count is incremented. The new count value appears in the register during S3P1 of the cycle following the one in which

the transition was detected. Since two machine cycles (24 oscillator periods) are required to recognize a 1-to-0 transition, the maximum count rate is 1/24 of the oscillator frequency. To ensure that a given level is sampled at least once before it changes, the level should be held for at least one full machine cycle.

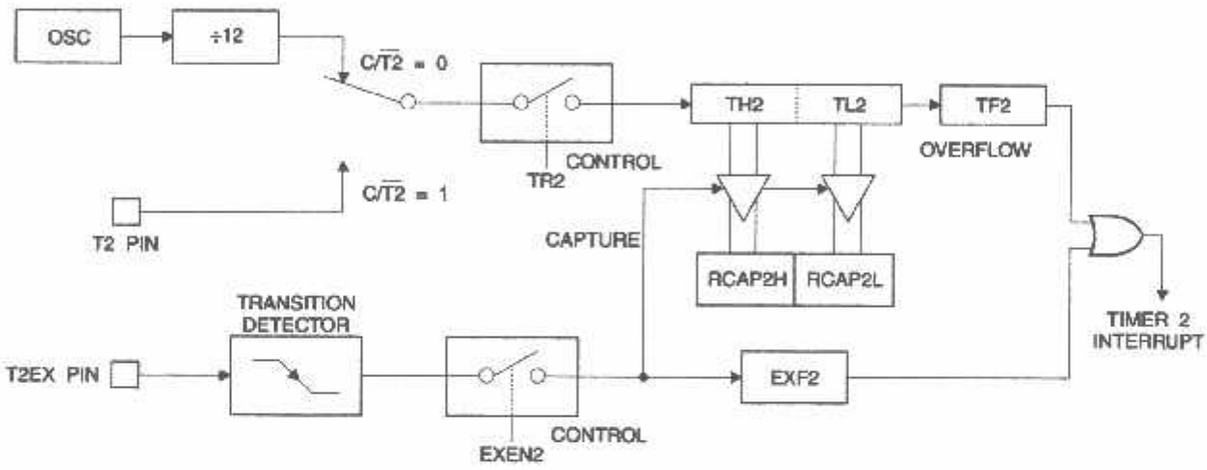
Table 8. Timer 2 Operating Modes

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

Capture Mode

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

Figure 1. Timer 2 in Capture Mode



Auto-reload (Up or Down Counter)

Timer 2 can be programmed to count up or down when configured in its 16 bit auto-reload mode. This feature is invoked by the DCEN (Down Counter Enable) bit located in the SFR T2MOD (see Table 9). Upon reset, the DCEN bit is set to 0 so that timer 2 will default to count up. When DCEN is set, Timer 2 can count up or down, depending on the value of the T2EX pin.

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

Setting the DCEN bit enables Timer 2 to count up or down, as shown in Figure 3. In this mode, the T2EX pin controls the direction of the count. A logic 1 at T2EX makes Timer 2 count up. The timer will overflow at 0FFFFH and set the TF2 bit. This overflow also causes the 16 bit value in RCAP2H and RCAP2L to be reloaded into the timer registers, TH2 and TL2, respectively.

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

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

Figure 2. Timer 2 in Auto Reload Mode (DCEN = 0)

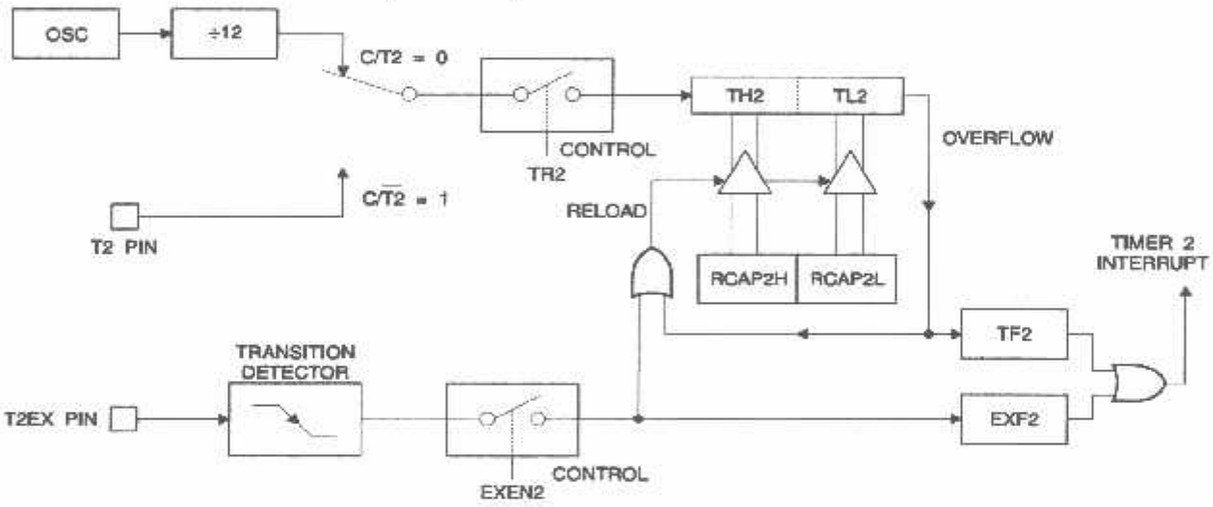


Table 9. T2MOD – Timer 2 Mode Control Register

T2MOD Address = 0C9H — Reset Value = XXXX XX00B

Not Bit Addressable

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

Symbol	Function
–	Not implemented, reserved for future use.
T2OE	Timer 2 Output Enable bit.
DCEN	When set, this bit allows Timer 2 to be configured as an up/down counter.

Figure 3. Timer 2 Auto Reload Mode (DCEN = 1)

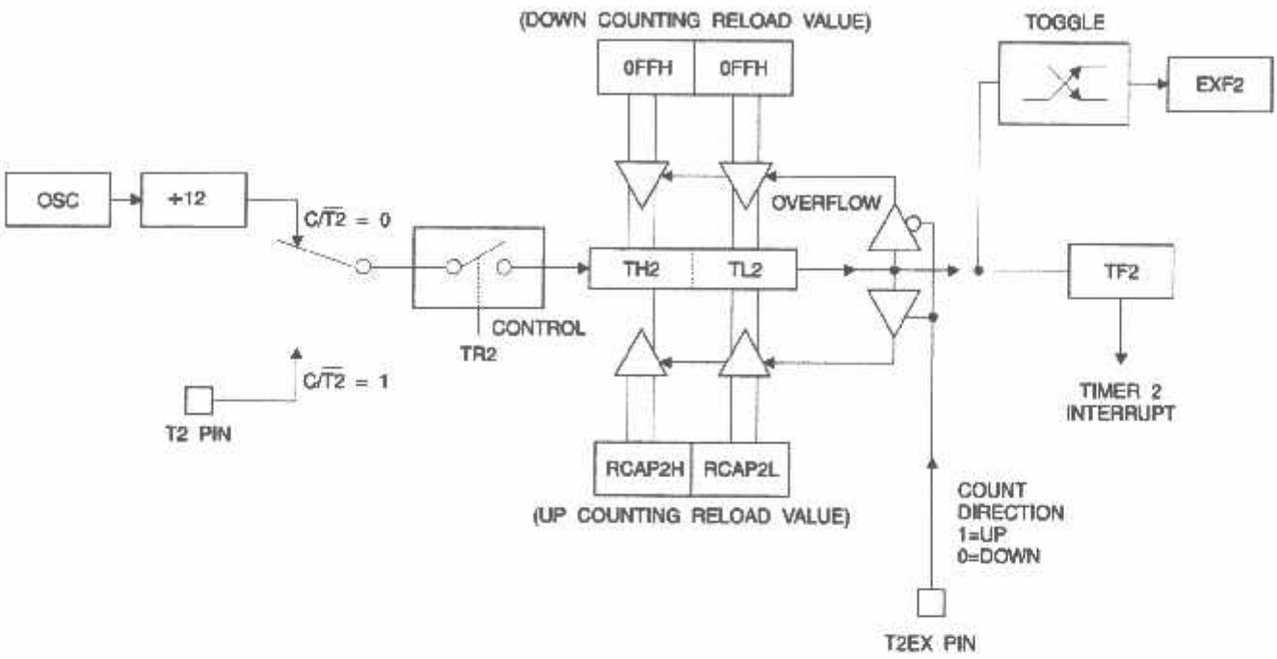
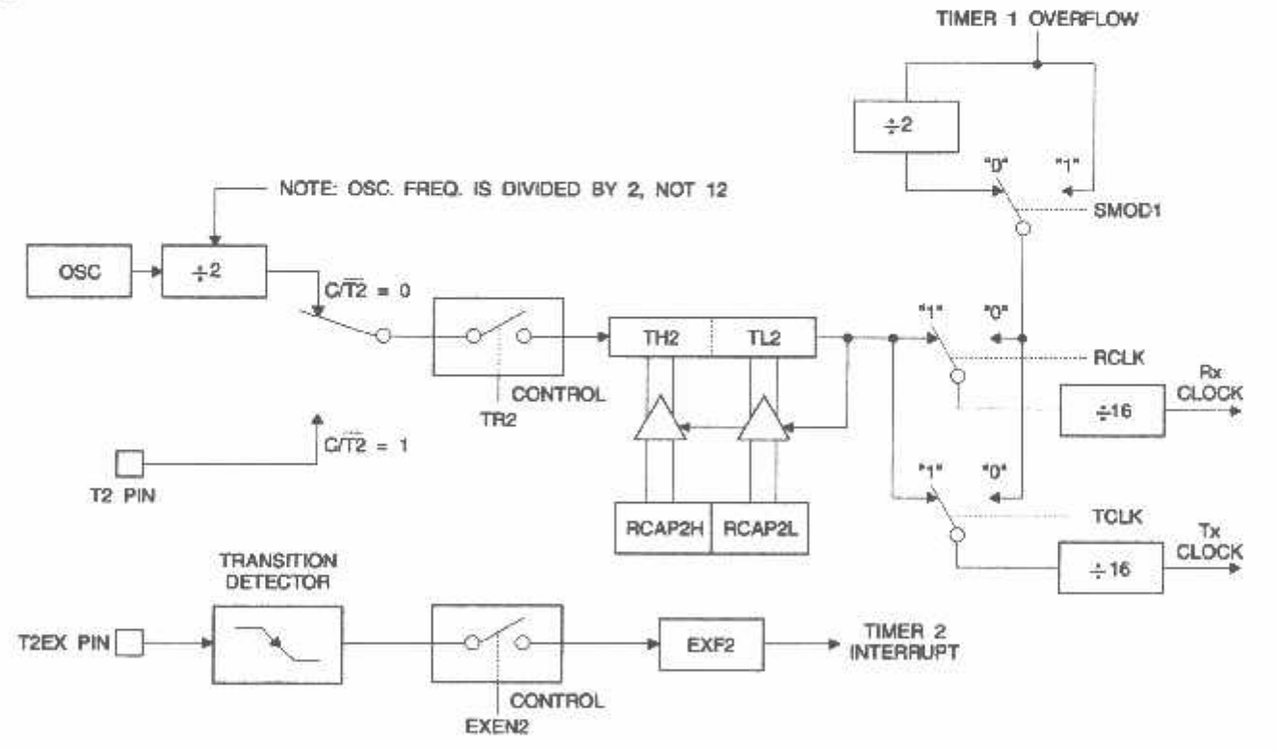
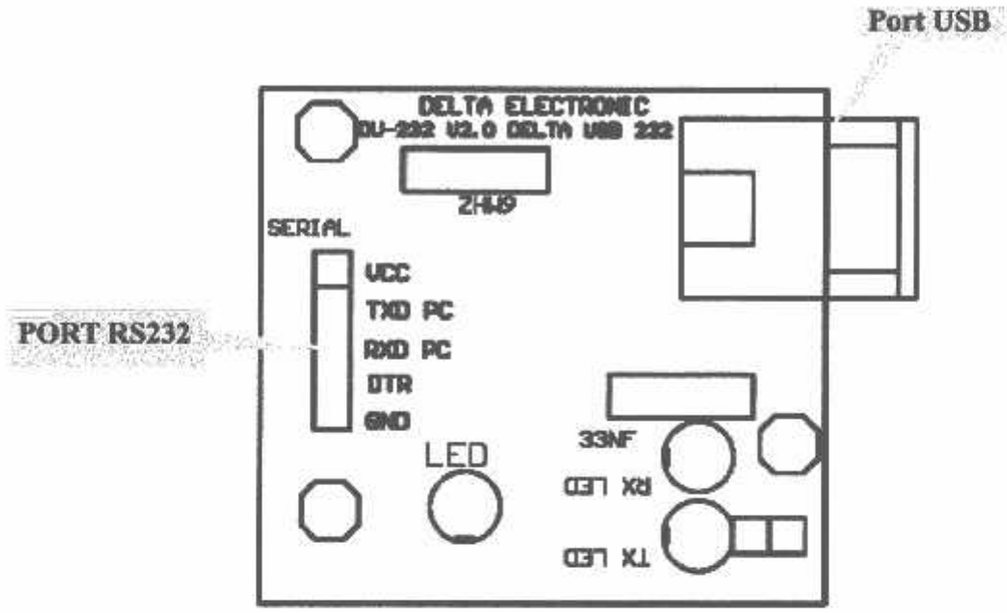


Figure 4. Timer 2 In Baud Rate Generator Mode



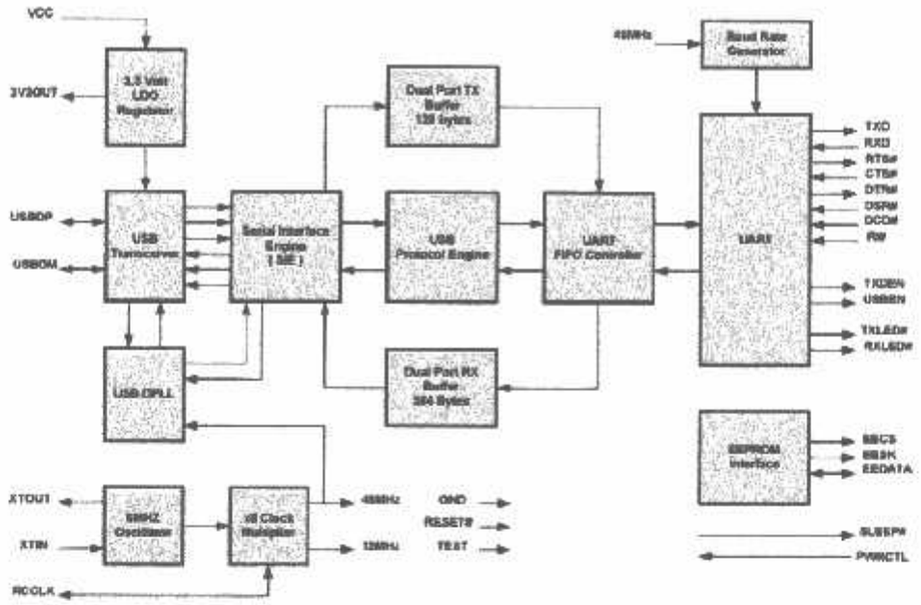
DU-232 DELTA USB TO RS232 CONVERTER



Spesifikasi

- RS232 level TTL
- TX/RX indicator LED
- Power output connector
- USB Cable include

Deskripsi



Sebagian besar PC maupun notebook dewasa ini mulai meninggalkan port serial dan beralih ke USB. Penggunaan USB memang lebih praktis karena selain kecepatannya yang lebih tinggi, port ini memiliki sumber tegangan 5 Volt yang dapat digunakan untuk memberi sumber daya pada sistem elektronik yang terhubung ke dalamnya.

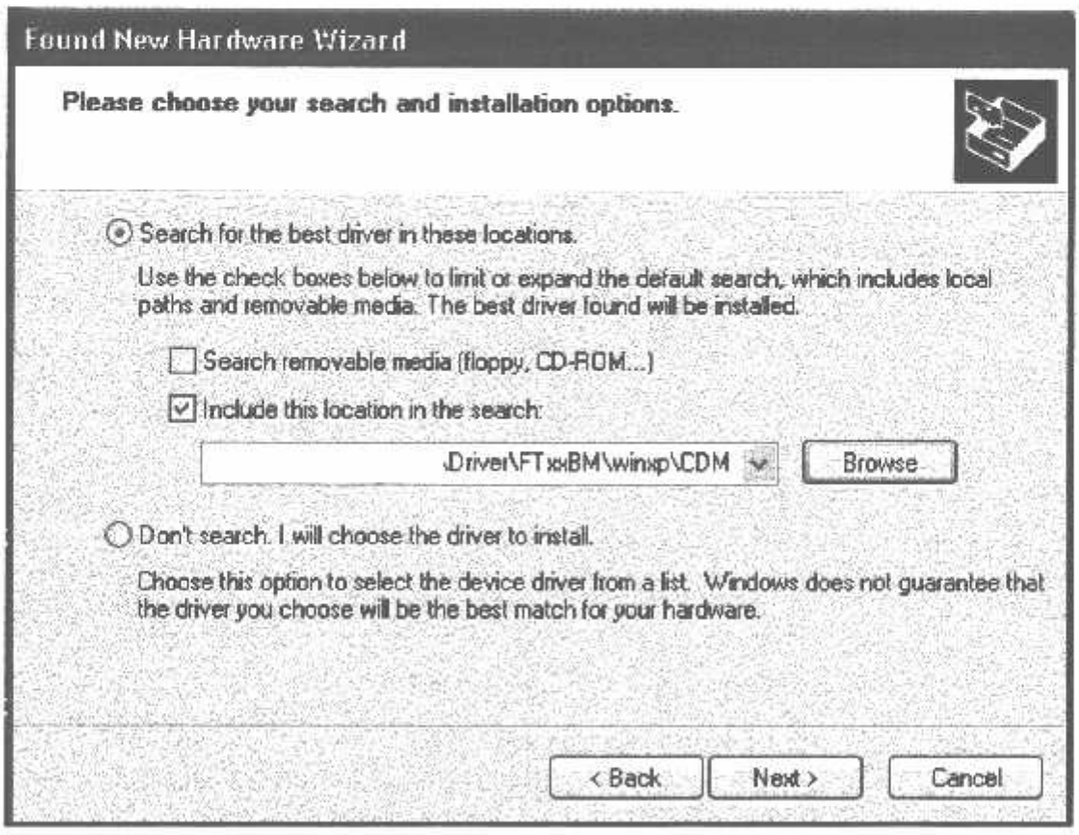
Sementara saat ini sebagian besar perangkat elektronik masih menggunakan port RS232 media komunikasinya dengan PC. Untuk menjembatani permasalahan tersebut maka banyak diluncurkan produk USB to RS232 yang membuat perangkat elektronik tersebut tetap terdeteksi sebagai COM (Port RS232) pada PC ataupun notebook. Software lama yang sebelumnya masih menggunakan COM pun tidak perlu diubah lagi karena perangkat tersebut masih dianggap berkomunikasi dengan COM (Port RS232)

Namun sebagian besar produk USB to RS232 yang ada di pasaran masih menggunakan level +/-12 Volt pada bagian RS232nya sedangkan mikrokontroler hanya dapat menggunakan level TTL 0/+5 Volt saja sehingga dibutuhkan IC MAX232 lagi untuk berkomunikasi dengan modul ini. Modul DU-232, USB to RS232 Converter produksi Delta Electronic memiliki level TTL 0/+5 volt pada bagian RS232nya sehingga dapat dihubungkan langsung pada sistem mikrokontroler tanpa menggunakan MAX232.

Instalasi DU-232

- Hubungkan Kabel USB ke Port USB PC dan Port USB DU-232
- LED Power indicator akan aktif yang menunjukkan adanya sumber tegangan memasuki Modul DU-232
- Windows akan meminta driver dari modul yang akan diinstall

- Pilih instalasi dari lokasi yang spesifik



- Arahkan ke folder driver yang ada di CD

- Setelah instalasi selesai, coba komunikasi dengan hyperterminal atau software-software lain yang mengakses ke Port RS232 (Com)
- Atur setting port serial sesuai pada gambar berikut