PERANCANGAN MEDIA PERANGKAT KERAS UNTUK SISTEM PENGAMANAN KODE LISENSI SOFTWARE MENGGUNAKAN METODE KRIPTOGRAFI DES MELALUI USB

SKRIPSI

Disusun oleh :
MUHAMMAD BAHESTI S
06.12.918

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

Thesis

LEMBAR PERSETUJUAN

PERANCANGAN MEDIA PERANGKAT KERAS UNTUK SISTEM PENGAMANAN KODE LISENSI SOFTWARE MENGGUNAKAN METODE KRIPTOGRAFI DES MELALUI USB

SKRIPSI

Disusun dan Diajukan Untuk Melengkapi dan Memenuhi Syarat
Guna Mencapai Gelar Sarjana Teknik

Disusun oleh :

MUHAMMAD BAHESTI S

06.12.918

Malang,Maret 2008

Disetujui Dosen Pembimbing I	Disetujui Dosen Pembimbing II
M.Ashar, ST, MT	**Ir. Yusuf Ismail Nakhoda, MT**
NIP.Y. 103 0500 408	**NIP. Y. 101 8800 0189**

Mengetahui
Ketua Jurusan Teknik Elektro S-1

Ir. F. Yudi Limpraptono, MT
NIP.Y. 103 9500 274

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

ABSTRAK

PERANCANGAN MEDIA PERANGKAT KERAS UNTUK SISTEM PENGAMANAN KODE LISENSI SOFTWARE MENGGUNAKAN METODE KRIPTOGRAFI DES MELALUI USB

Muhammad Bahesti S - 06.12.918
Dosen Pembimbing I : M. Ashar, ST, MT
Dosen Pembimbing II : Ir. Yusuf Ismail Nakhoda, MT

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

Kata Kunci : DES, FT232,Kode Lisensi, Kriptografi, *Donggle*, Usb *to* Serial.

Pada saat ini aturan mengenai perlindungan karya cipta perangkat lunak komputer dilindungi oleh Undang-Undang Hak Cipta, sehingga penggandaan perangkat lunak secara bebas hanya dapat dilakukan oleh pengembang dari perangkat lunak itu sendiri atau orang yang menerima hak untuk hal tersebut. Apabila dilakukan oleh orang lain maka dapat dikatakan orang itu telah membajak suatu karya cipta. Namun walaupun pembatasan terhadap perangkat lunak komputer telah diatur dalam suatu undang-undang tidak menutup kemungkinan untuk terjadinya penggandaan ilegal pada *software*, hal ini terbukti karena di Indonesia sampai saat ini masih saja terjadi pelanggaran hak cipta khususnya penggandaan terhadap perangkat lunak dan lisensinya.

Berdasarkan permasalahan diatas, maka diterapkan metode kriptografi DES (*Data Encryption Standard*) pada penyamaran data untuk pengamanan kode lisensi software. Sistem pengamanan yang dibuat menggunakan sebuah perangkat keras yang berfungsi sebagai mediator kunci yang akan mengirimkan kunci atau kode lisensi program pada PC melalui port USB. Pada teknisnya, dari sisi pihak pengembang sistem perangkat lunak perangkat keras ini dapat digunakan untuk operasi penguncian *software*, dan sebaliknya, pada sisi pengguna perangkat ini dapat digunakan untuk operasi pengaktifan *software*.

Dari analisa data pada perancangan sistem ini didapatkan hasil tingkat kesamaran dari hasil penyembunyian pesan tersamar sebesar 42,875% untuk hasil penyamaran dengan *plaintext*(pesan tersamar) yang sama dengan perbedaan satu karakter besar/kecil karakter pada kode lisensi(kunci), dan 26,56% untuk *plaintext*(pesan tersamar) yang sama dengan karakter kode lisensi(kunci) berurut dan perbedaan pada besar/kecil karakter.

LEMBAR PERSEMBAHAN

Segala puji bagi Allah SWT, Tuhan semesta alam. Terima kasih atas rahmat yang diberikan dalam menyelesaikan skripsi ini sebagai syarat untuk memperoleh Gelar Sarjana Teknik Elektro.

Skripsi ini saya persembahkan untuk kedua orang tua yang selalu berkorban dengan tulus, ikhlas dan kebesaran hatinya untuk terus memberi motivasi kepada saya agar terus berusaha menjadi manusia yang lebih baik.

Dengan terselesaikannya skripsi ini, penulis mengucapkan terima kasih yang tulus kepada :

Bapak M. Ashar, ST, MT dan bapak Ir. Yusuf Ismail Nakhoda, MT selaku dosen pembimbing atas waktu dalam memberikan bimbingan, pengarahan, bantuan dan masukan positif dalam penyusunan skripsi ini kepada saya.

Bapak Ir. F. Yudi Limpraptono selaku ketua jurusan Teknik Elektro S-1 yang telah banyak membantu penulis dalam menyelesaikan studi pada Institut Teknologi Nasional Malang.

Seluruh keluarga besar Lamahala di Malang atas masukan baik secara langsung maupun tidak langsung, dan semua pihak yang tidak dapat saya sebutkan satu persatu, saya mengucapkan terima kasih.

MALANG, Maret 2008

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1. Latar Belakang

Lisensi suatu produk merupakan suatu hal yang sangat penting, dikarenakan dengan lisensi tersebut, produk dapat dilindungi oleh hukum dari pembajakan. Ini berlaku juga bagi produk yang berbentuk perangkat lunak (*software*), namun walaupun terlindung oleh hukum, *software* masih banyak juga yang digandakan secara ilegal.

Untuk mencegah pembajakan ini tim pengembang *software* menambah sistem keamanan dari *software* yang akan dijual dengan menambahkan nomor seri pada tiap-tiap *compact disk*, sehingga *user* dapat menggunakan perangkat lunak tersebut jika *user* telah memiliki kode lisensi tersebut. Sehingga apabila ada yang menggandakan *software* tersebut, dan pengganda tersebut tidak mengetahui nomor seri yang disertakan produsen dalam penjualan *software*, maka *software* tidak dapat diaktifkan..

Namun, walaupun telah digunakan nomor seri, pembajakan tetap terjadi, yaitu pembajak membeli *software* asli dengan nomor seri kemudian menggandakan *software* tersebut berikut dengan nomor serinya, sehingga didapat hasil ganda dari *software* berikut nomor seri dan kemudian dikomersilkan.

Oleh karena permasalahan yang disebutkan diatas maka penulis mencoba untuk merancang suatu sistem pengamanan pada lisensi *software*

menggunakan perangkat keras yang berfungsi sebagai media penyimpan dan pengirim kode lisensi dari *software* tersebut, dengan menggunakan hardware ini diharapkan sistem yang dirancang dapat meningkatkan tingkat keamanan lisensi *software* tersebut dari penggandaan yang ilegal atau praktek pembajakan, dikarenakan pada sistem ini pihak pembajak tidak dapat memasukan kode lisensi tersebut secara manual.

1.2. Tujuan Penulisan

Tujuan dari skripsi ini adalah untuk merancang Sistem Pengamanan Kode Lisensi *Software* Menggunakan Media Perangkat Keras mengggunakan metode Kriptografi DES melalui USB untuk melindungi *software* dari penggandaan yang ilegal.

1.3. Rumusan Masalah

Adapun perumusan masalah pada perancangan sistem ini adalah :

Bagaimana merancang suatu sistem pengamanan lisensi *software* yang memanfaatkan perangkat keras USB yang berfungsi sebagai media penyimpan dan pengirim kode lisensi tersamar yang akan didekripsikan pada sisi *software* untuk menghindari penyalah gunaan pada lisensi *software* tersebut.

1.4. Batasan Masalah

Dalam pembahasan ini dibatasi dengan beberapa batasan masalah, yaitu :

1. Mikrokontroler yang digunakan adalah mikrokontroler AT89S8252
2. Perangkat lunak pada PC menggunakan compiler *Borland Delphi 6.0.*
3. Metode kriptografi yang digunakan adalah DES (*Data Encryption Standard*)

   64 Bit plaintext

1.5. Metodologi

Metodologi yang dipakai dalam pembuatan skripsi ini adalah:

1. Studi Literatur

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

2. *Field Research*

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

3. *Design hardware software*

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

4. Pengujian Alat

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

5. Penyusunan Laporan

Membuat laporan yang terdiri dari: Pendahuluan, Landasan Teori, Perencanaan dan Pembuatan Alat, Pengujian Alat dan Penutup.

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 TINJAUAN PUSTAKA

Membahas teori dasar penunjang perancangan dan pembuatan alat.

BAB III PERANCANGAN SISTEM

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

BAB IV PENGUJIAN DAN SIMULASI SISTEM

Membahas tentang pengujian peralatan secara keseluruhan dan analisa hasil pengujian.

BAB V KESIMPULAN

Berisikan kesimpulan dan saran.

BAB II

TINJAUAN PUSTAKA

2.1. Pendahuluan

Pada bab ini akan membahas tentang dasar teori penunjang yang menyangkut komponen-komponen dan teori penyamaran data yang akan digunakan pada perancangan sistem ini.

2.2. Mikrokontroler AT89S8252

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.

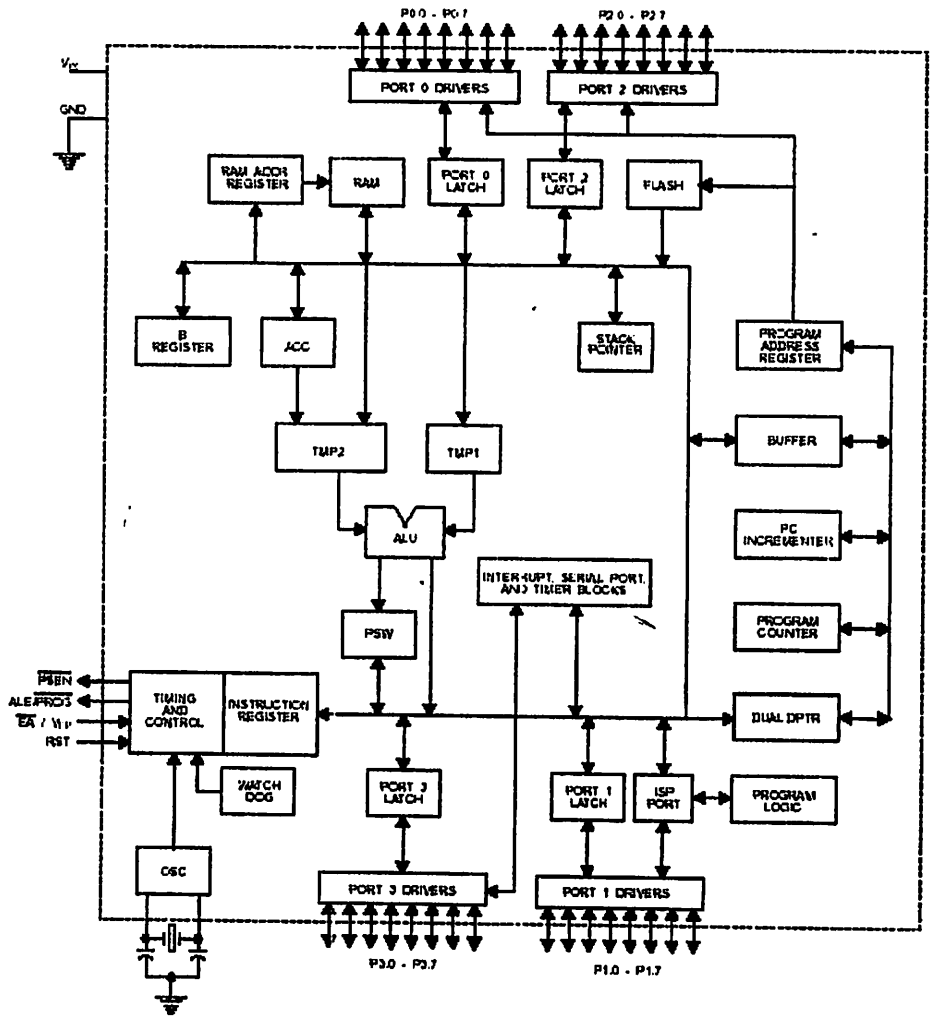
Mikrokontroler AT89S8252 merupakan mikrokontroller 8 bit kompatibel dengan standart industri MCS-51 baik atas segi pemrograman maupun atas kaki tiap pin.

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)

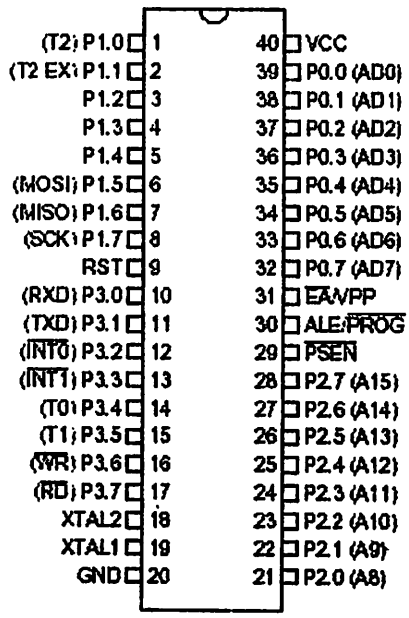
- 2 Kbyte EEPROM
- 256 byte RAM internal
- 32 *Programmable I/O lines*
- 3 buah timer / counter 16 bit
- Sebuah *port serial* dengan *control full duplex* UART (*Universal Asynchronous Receiver Transmiter*)
- SPI *Serial Interface*
- *Programmable Watchdog Timer*
- *Dual Data Pointer*
- Frekuensi kerja 0 sampai 24 MHz
- Tegangan operasi 2,7 sampai 6 Volt
- Kemampuan melaksanakan operasi *Boolean* (bit)

Sedangkan untuk blok diagram AT89S8252 diperlihatkan dalam Gambar 2.1.



Gambar 2.1. Blok Diagram AT89S8252 [1]

Berikut adalah struktur fisik dari AT89S8252.



Gambar 2.2. Konfigurasi Pin-Pin AT89S8252 [1]

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 direrima melalui kaki P3.0 (RXD), dan kaki P3.1 (TXD) dipakai untuk menyalurkan *clock* pendorong data seri yang di bangkitkan oleh 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 ½ 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 bt 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 berfungsi 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 frekwensi 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 89C51. 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).

MSB							LSB
SMO	SM1	SM2	REN	TB8	RB8	SM4	R1

Gambar 2.3. *Special Function Register* AT89S8252[1]

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*).

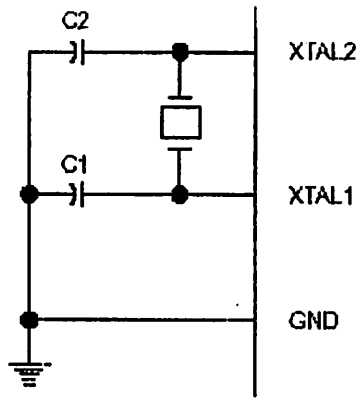
$$\text{Baud Rate} = \frac{(K \times \text{Frekuerisi Osilator})}{(32 \times 12 \times [256 - TH1])} \quad \ldots\ldots (2.1)$$

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]

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

Nilai THl 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.

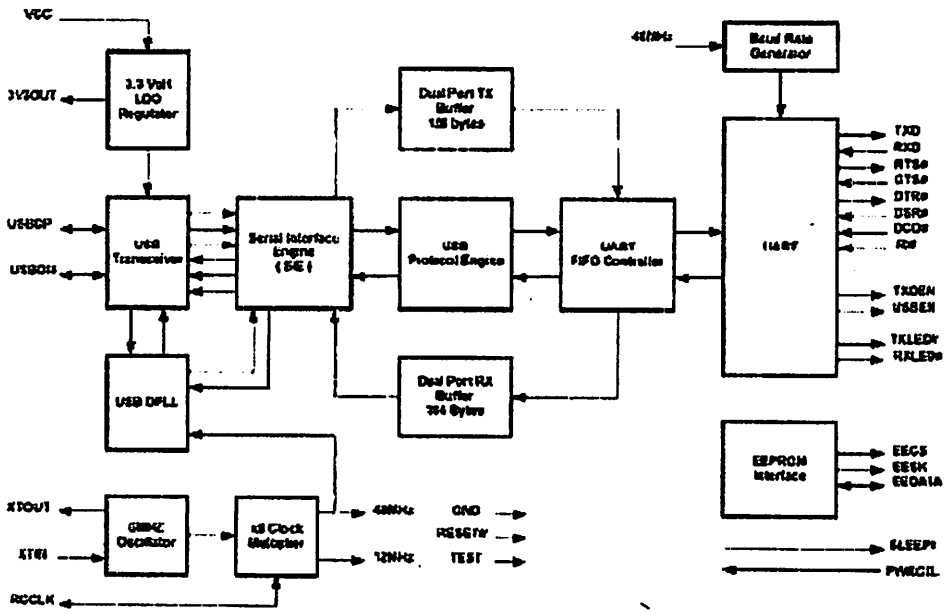


Gambar 2.4. Osilator Eksternal AT89S8252[1]

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, degan 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. Berikut adalah blok diagram dari FT232BM.



Gambar 2.5. 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 kabel USB.

3) USB DPLL

Sinyal ini adalah sinyal pengunci data USB.

4) 6 MHz Oscilator

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 dan *Baud Rate Generator.*

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

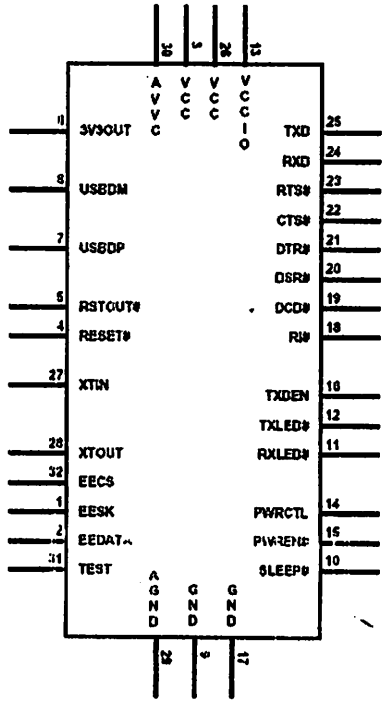
UART mendukung *asynchronous* 7 / 8 bit pengkonversian *data Parallel-Serial* dan *Serial-Parallel* pada antarmuka RS232 (RS422 and RS485). Sinyal kontrol didukung oleh UART RTS, CTS,DSR , DTR, DCD and RI. UART mengatur sinyal *transmiter enable control signal* (TXDEN) untuk berhubungan dengan penerima 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 2.6. 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.*
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 1.1 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 2.7. Konektor USB (Tipe A dan Tipe B)[2]

Tabel 2.2
Konfigurasi/pin Konektor USB[2]

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.

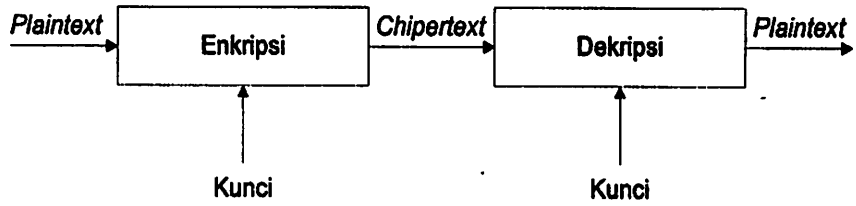
2. *Message type*

   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 DES *(Data Encryption Standard)*

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 2.8 Fungsi enkripsi dan dekripsi sederhana[5]

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 *chipertext* (informasi atau pesan yang tidak jelas atau rahasia)

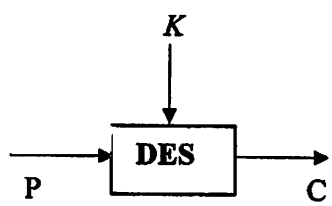
KE adalah kunci enkripsi

KD adalah kunci dekripsi

Pada Januari 1977, pemerintah AS menetapkan *chiper* produk yang dibuat oleh IBM sebagai *standard* resmi informasi yang tidak rahasia. DES *(Data Encryption Standard)*, merupakan blok *chiper* kunci simetrik, yang telah secara luas digunakan dalam dunia industri. DES merupakan nama dari sebuah

algoritma untuk mengenkripsi data yang dikeluarkan oleh *Federal Information Processing Standard* (FIPS) 46-1 Amerika Serikat.

Secara umum algoritma DES dibagi menjadi 3 kelompok yang mana kelompok yang satu dengan yang lain saling berinteraksi antara satu dengan yang lain, yaitu: Penjadualan kunci, Enkripsi data dan Dekripsi data, DES merupakan algoritma simetris yang beroperasi pada sebuah blok *plaintext* 64 bit dan menggunakan kunci yang sama untuk proses enkripsi dan dekripsinya yaitu kunci 64 bit, akan tetapi panjang kunci efektif yang digunakan dalam proses eksekusi adalah 56 bit. Atau lebih tepatnya kunci input K, k=64 bit dan 8 *bit parity* tidak dilibatkan sehingga mengurangi jumlah kunci menjadi 56 bit. Oleh karena itu, output tergantung pada 64 bit input dan 56 bit kunci. Pesan dekripsi menggunakan blok penyandi yang sama dengan blok proses enkripsi dimana kunci dekripsinya didapatkan dari kunci enkripsi.



Gambar 2.9. Input-Output DES[4]

Keterangan :

P = *Plaintext*

C = *Chipertext*

K = Kunci *(key)*

Pada teknisnya algoritma DES memiliki struktur sebagai berikut:

a. Ukuran blok masukan *(plaintext)*: 64 bit.
b. Ukuran kunci *(key)*: 64 bit.
c. Masukan blok *input* (64 bit) ke inisial permutasi.
d. Penjadualan kunci yang terdiri dari 16 putaran.
e. 16 putaran untuk mendapatkan 48 bit sub kunci dari 56 bit kunci.
f. Setiap iterasi dari 16 iterasi terdiri dari 16 pengacakan blok 64 bit menjadi kunci 48-bit.
g. Blok output dimasukan ke *invers inisial permutasi* (IP^{-1})

Berikut adalah konsep dasar dari algoritma DES:

1) Operasi DES pada 64 bit blok *plaintext*. Kemudian 64 bit blok dibagi menjadi 2 bagian yaitu separuhnya sebelah kanan dan separuhnya lagi sebelah kiri. Masing-masing bagian memiliki panjang 32 bit
2) Terdiri dari 16 putaran, setiap putaran memiliki operasi yang identik disebut fungsi *f*, yang mana data dikombinasikan dengan kunci
3) Setelah 16 putaran selesai, data sebelah kanan dan kiri digabung dan dimasukan ke permutasi terakhir (kebalikan dari inisial permutasi).

4) Inisial permutasi

Tabel 2.3. Tabel *Initial Permutation*[3]

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

- ✓ Dilakukan sebelum putaran pertama
- ✓ Bit plaintext dipindah ke lokasi baru. Misalnya: bit 58 ke bit 1, bit 50 ke bit 2, dan seterusnya.
- ✓ Inisial permutasi (IP) tidak mempengaruhi tingkat keamanan DES.

5) Tranformasi kunci

Kunci 48 bit diekstrak dari kunci 56 bit.

Setiap putaran menghasilkan kunci 48 bit yang berbeda-beda.

6) Ekspansi permutasi

Tabel 2.4. Tabel *Expansi*[3]

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

- ✓ Operasi ini untuk mengekspan atau memperbesar ukuran bit data sebelah kanan Ri dari 32 bit ke 48 bit.
- ✓ Merubah beberapa bit dengan menduplikasikan/mengulang bit-bit tertentu
- ✓ Merubah ukuran bit agar sama dengan ukuran subkunci sehingga dapat melakukan operasi XOR

7) Subtitusi S-box

Setelah *subkey* di XORkan dengan blok sebelah kanan yang telah diekspansi, blok 48 bit itu kemudian dimasukan ke operasi subtitusi melalui *box* subtitusi (S-*box*)

8) Subtitusi P-*box*

Tabel 2.5. Tabel *Permutation*[3]

16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

- ✓ Hasil 32 bit dipermutasi sesuai P-*box*
- ✓ Permutasi ini memetakan setiap bit *input* ke posisi *output*
- ✓ Tidak ada bit yang digunakan dua kali, tidak ada bit yang diabaikan.

9) *Final* permutasi (FP) atau yang sering disebut invers inisial permutasi (IP^{-1})

- ✓ Merupakan kebalikan dari permutasi awal (*initial permutation*)
- ✓ Blok sebelah kanan dan blok sebelah kiri tidak di *cross over* setelah melewati putaran terakhir yaitu putaran ke 16, sebagai gantinya blok R16 L16 digunakn sebagi masukan ke *final* permutasi atau *invers* permutasi.

Pada proses enkripsi dan dekripsi digunakan algoritma yang sama. Hanya saja penjadualan kuncinya yang berbeda, penjadualan kunci enkripsi terbalik dengan penjaduaIn kunci pada dekripsi misalnya, operasi enkripsi penjadulan kunci K1, K2,..., K16 sedangkan pada dekripsi K16, K15,...,K1

BAB III

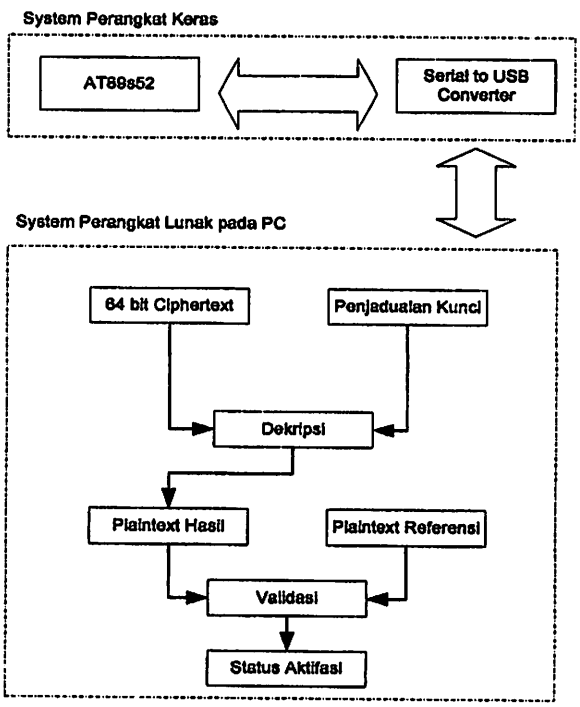
PERANCANGAN SISTEM

3.1 Pendahuluan

Bab ini akan membahas tentang perancangan sistem yang meliputi perancangan perangkat keras (*Hardware*) dan perangkat lunak (*Software*) dari sistem pengamanan kode lisensi *software* aplikasi ini.

3.2. Desain sistem

Pada tahap desain sistem ini memiliki perwakilan blok diagram sistem sebagai berikut:



Gambar 3.1. Digram Blok Sistem

Dari gambar blok diagram 3.1 dapat dijelaskan fungsi dari setiap blok diatas sebagai berikut:

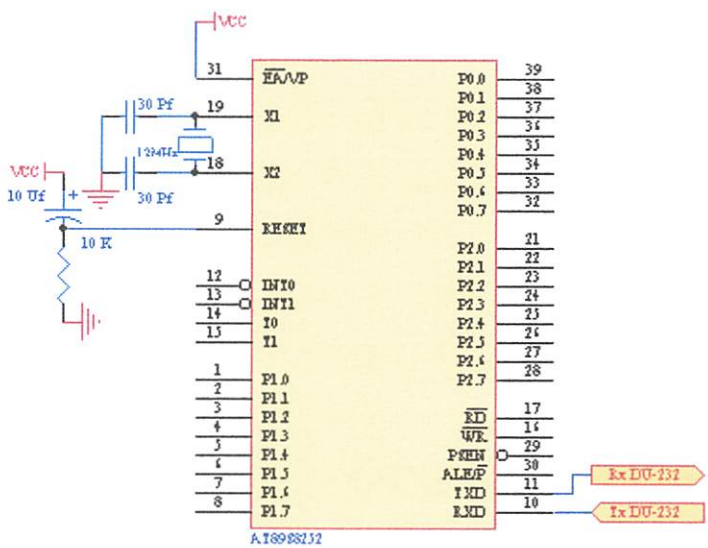
- Mikrokontroler berfungsi sebagai pengirim data kunci tersamar.
- *Serial to USB Converter* adalah modul rangkaian yang berfungsi sebagai pengubah Serial ke USB.
- 64 bit *ciphertext* adalah data yang berukuran 64bit yang akan di dekripsikan bersama-sama dengan kunci
- Penjadualan kunci adalah proses pembagkitan 15 kunci untuk mendapatkan kunci ke-15 yang tersamarkan.
- Dekripsi adalah proses pengembalian data *cipher* (tersamar) menjadi data *plaintext* (asli)
- *Plaintext* hasil adalah data hasil proses dekripsi
- *Plaintext* referensi adalah data *plaintext* asli yang berfungsi sebagai pembanding.
- Validasi adalah proses perbandingan data *plaintext* hasil dengan data *plaintext* referensi.
- Status aktivasi adalah keputusan pengaktifan atau penonaktifan *software* tersebut yang didapat dari hasil proses validasi.

3.3 Perancangan perangkat keras

Pada perancangan perangkat keras sistem ini terdapat beberapa modul yang akan dirangkaikan yaitu perencanaan mikrokontroler dan perencanaan rangkaian konverter *serial to USB*. Berikut adalah pembahasan dari perencanaan yang diterapkan pada implementasi sistem ini.

3.3.1 Perencanaan Mikrokontroller AT89S8252

Pada perencanaan ini mikrokontroler digunakan sebagai media yang akan mengirimkan data berupa kode lisensi 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 AT89S8252

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.

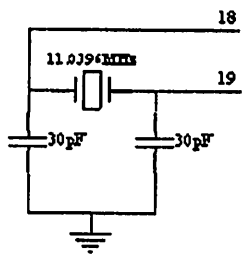
3.3.1.1 Rangkaian Clock

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.

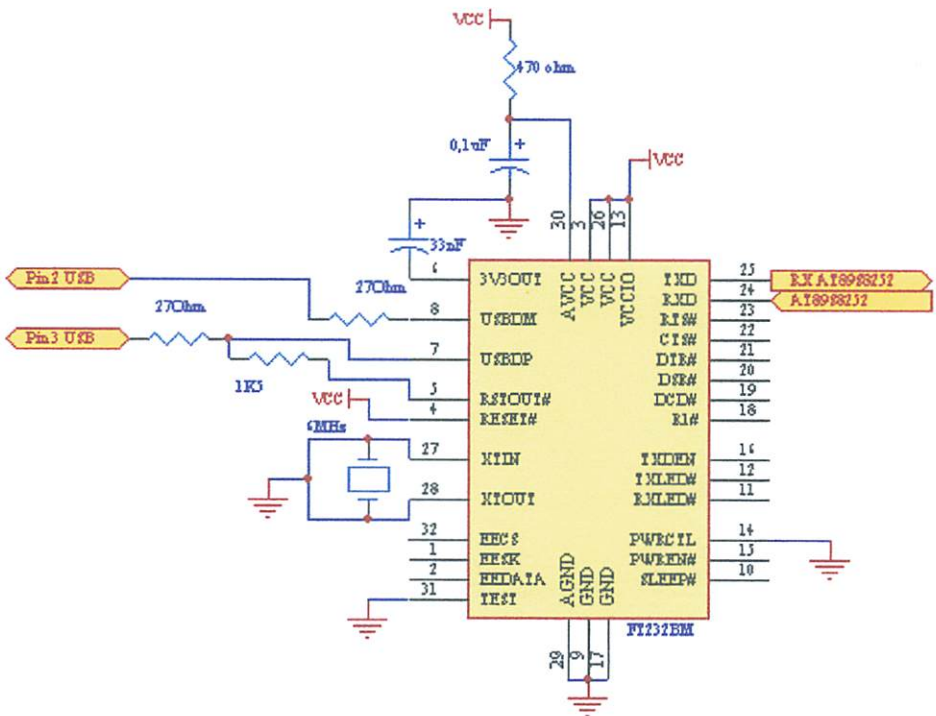
Gambar dari rangkaian clock adalah sebagai berikut:



Gambar 3.3. Rangkaian Clock AT89S8252

3.3.2 Konverter Serial USB

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



Gambar 3.4. Rangkaian Konverter Serial USB FT232BM

Konfigurasi pin yang digunakan pada rangkaian ini adalah sebagai berikut:

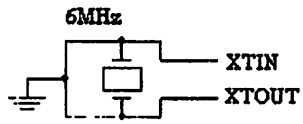
- Pin 30 dihubungkan dengan kapasitor sebesar 33nF dan *ground* merupakan output dari regulator internal.
- Pin 8 dihubungkan dengan pin 2 USB pada PC sebagai sinyal data negatif yang terhubung paralel dengan tahanan sebesar 27 ohm dan pin 5 yang merupakan output dari internal reset generator.

- 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.

Pada sistem ini nominal kristal yang digunakan adalah sebesar 6MHz yang bersesuaian dengan aturan pada datasheet FTDI. Berikut adalah skematik dari rangkaian clock yang digunakan.



Gambar 3.5. Rangkaian Clock FT232BM

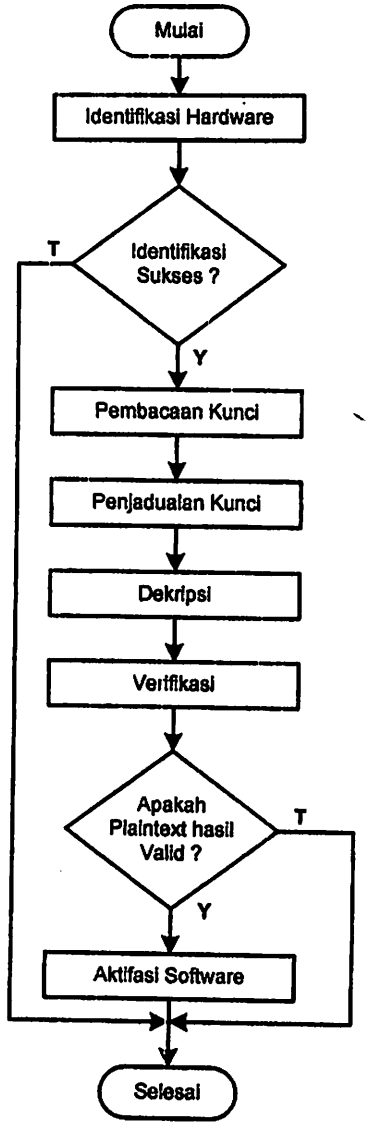
3.4 Perencanaan Perangkat Lunak (*Software*)

Pada teknis perancangan perangkat lunak tahap awal yang perlu dilakukan adalah perencanaan algoritma dari perangkat lunak yang akan dibuat.

Dalam perancangan alat ini perangkat lunak yang dibuat menggunakan bahasa pemrograman *assembler* dan perangkat lunak *Borland Delphi* dengan struktur bahasa *Pascal*. Perencanaan algoritma dari perangkat lunak berfungsi agar *coding* pada perangkat lunak yang dibuat lebih sistematis sehingga akan memudahkan untuk pencarian kesalahannya.

3.4.1 *Flowchart* Sistem

Flowchart sistem adalah tahapan proses yang dilakukan oleh sistem secara keseluruhan dalam satu siklus operasi.



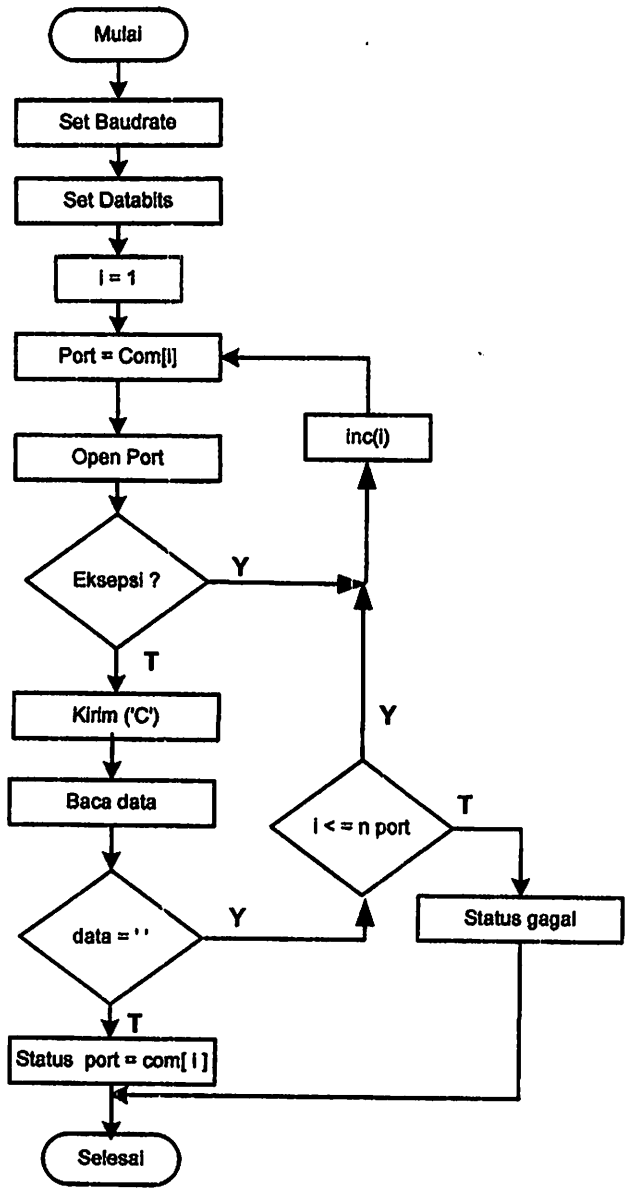
Gambar 3.6. *Flowchart* Sistem

Berikut adalah penjelasan dari diagram alir diatas:

1. Mulai
2. Identifikasi *hardware*
3. Apakah identifikasi sukses
    a. Jika Ya Lakukan langkah selanjutnya.
    b. Jika tidak Lakukan langkah-10
4. Proses pembacaan kunci dari perangkat keras dilakukan
5. Pembangkitan 15 kunci pada proses penjadualan kunci dilakukan
6. Proses dekripsi dengan DES
7. Proses verifikasi *plaintext* hasil dengan *plaintext* acuan
8. Apakah *plaintext* hasil valid:
    a. Jika Ya, Lanjutkan langkah selanjutnya.
    b. Jika Tidak, Lakukan langkah-10
9. Aktivasi *software* dilakukan
10. Selesai

3.4.2 *Flowchart* identifikasi *hardware* pada PC

Flowchart ini menjelaskan tentang sistem jabat tangan perangkat keras dengan PC agar PC dapat mendeteksi secara automatis COM atau *port* koneksi yang digunakan.



Gambar 3.7. *Flowchart* Identifikasi *Hardware*

Berikut ini adalah penjelasan linguistik dari diagram alir diatas:

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. Apakah terjadi eksepsi ?

   Jika Ya lakukan langkah selanjutnya.

   Jika Tidak lakukan langkah-9.
8. Jumlahkan pencacah i dengan 1 dan kembali ke langkah-5.
9. Kirim karakter "C".
10. Baca data.
11. Apakah Data='' ?

    Jika Ya lakukan langkah selanjutnya.

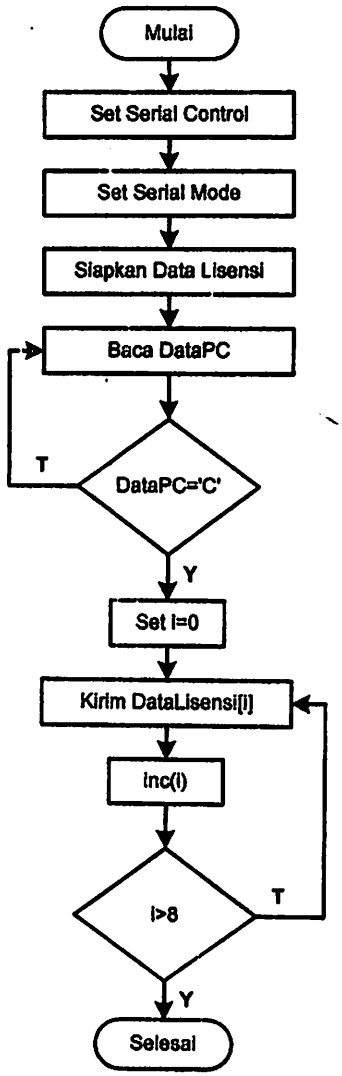
    Jika Tidak lakukan langkah-14.
12. Apakah i <=n_port ?

    Jika Ya, maka kembali ke langkah-8.

    Jika Tidak, maka lakukan langkah-13.
13. Status gagal dan lanjutkan ke langkah-15
14. Status *Port* = COM ke-i.
15. Selesai .

3.4.3 *Flowchart* Pengiriman kode lisensi

Flowchart ini adalah penjelasan diagram alir dari tahapan proses pada mikrokontroler untuk mengirimkan kode lisensi pada PC.



Gambar 3.8. *Flowchart* Pemgiriman Kode Lisensi

Representasi bahasa alamiah untuk diagram alir diatas adalah sebagai berikut:

1. Mulai.
2. *Set Serial Control.*
3. *Set Serial Mode.*
4. Siapkan data lisensi pada *Serial Buffer.*
5. Baca DataPC.
6. Apakah DataPC ='C' ?

   Jika Ya, maka lakukan langkah selanjutnya.

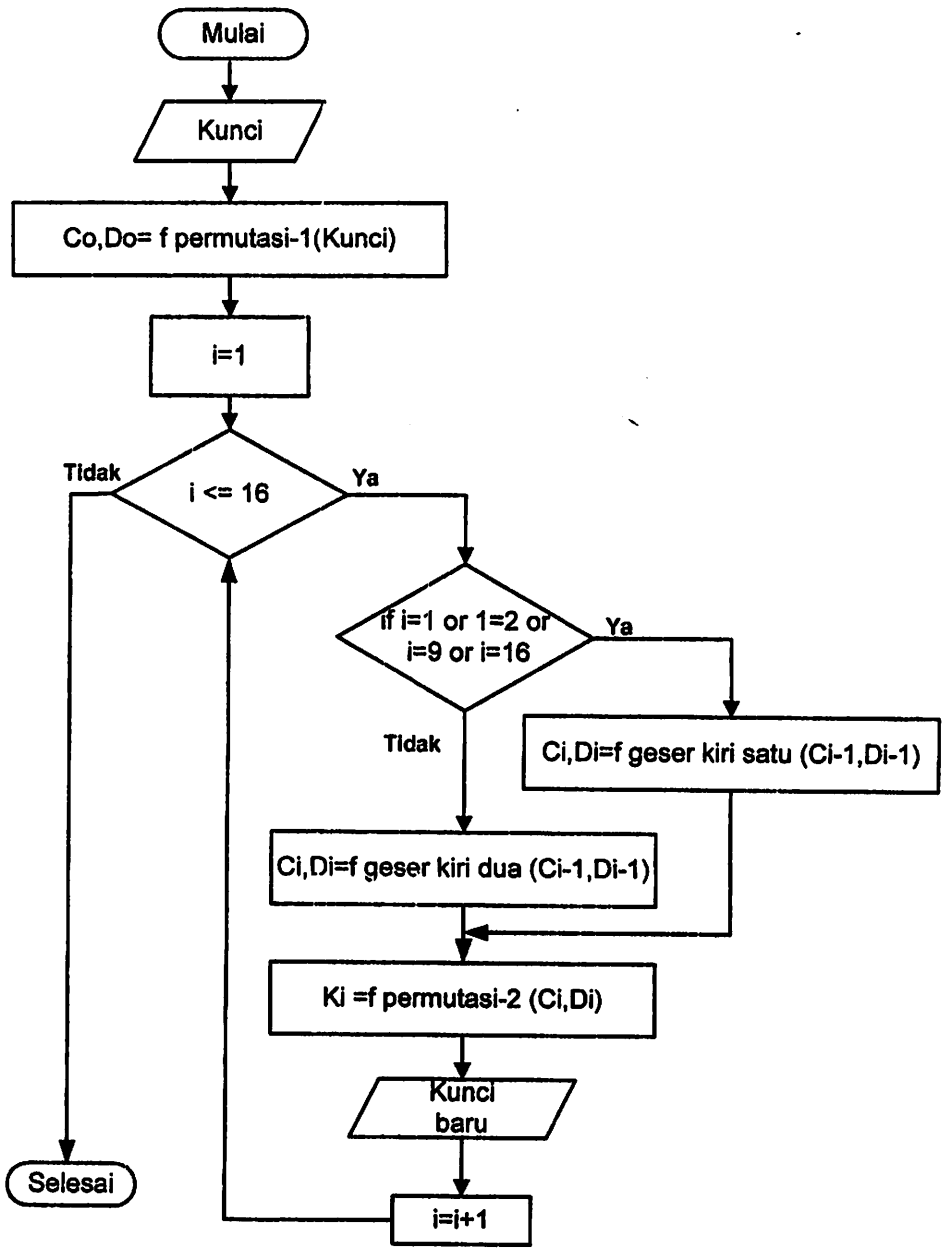
   Jika Tidak, maka kembali ke langkah-5.
7. *Set* pencacah i=0.
8. Kirim karakter data lisensi ke-i
9. Jumlahkan i dengan 1.
10. Apakah i>8 ?

    Jika Ya, maka lakukan langkah selanjutnya.

    Jika Tidak, maka kembali ke langkah-8.
11. Selesai.

3.4.4 *Flowchart* Penjadualan kunci pada PC

Flowchart ini adalah alir proses dari 15 iterasi pembangkitan/penjadualan kunci DES pada PC.



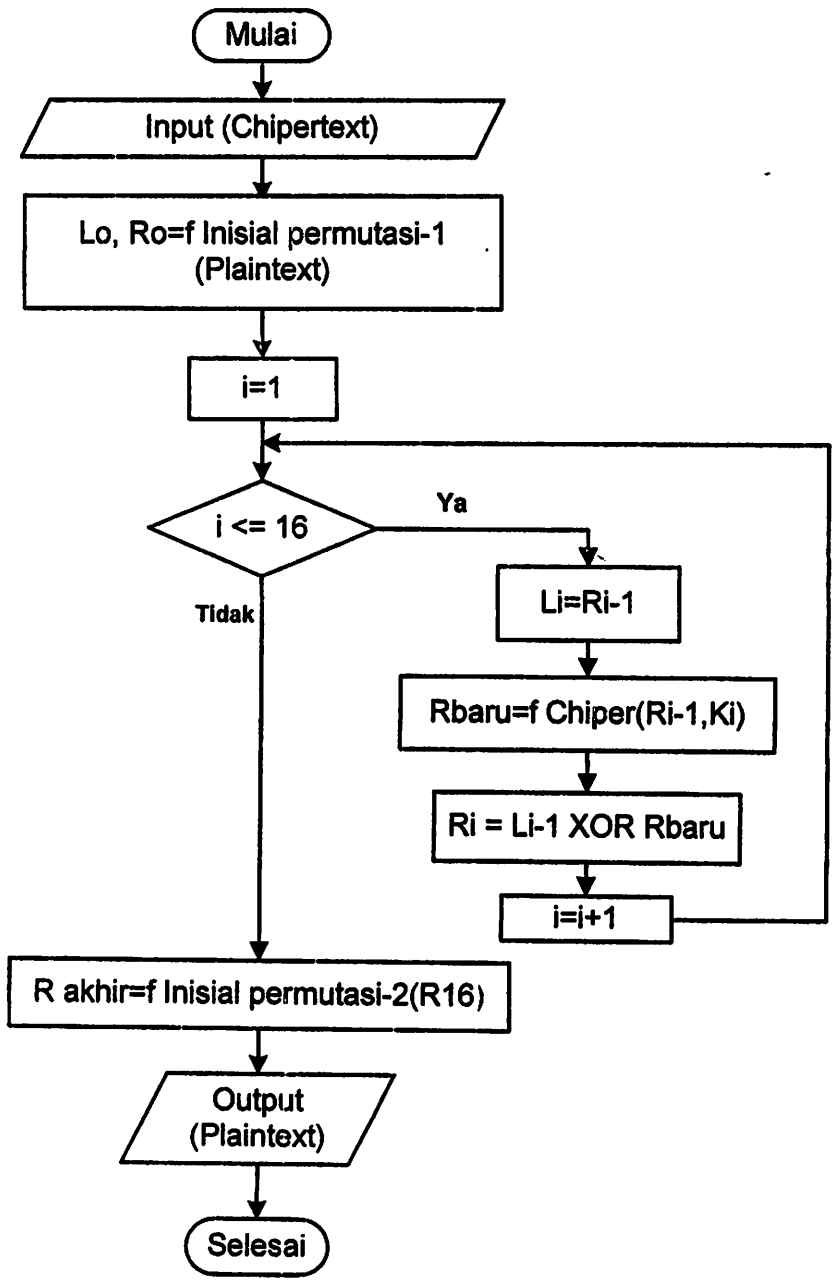
Gambar 3.9. *Flowchart* Penjadualan Kunci DES pada PC

Berikut adalah penjelasan dari penjadualan kunci diatas.

1. Masukan kunci dengan panjang 8 karakter.
2. Kunci dimasukan ke fungsi permutasi pertama dengan memindahkan nomor bit sesuai tabel PC-1.
3. Kunci dipecah menjadi 2 bagian C_0 dan D_0.
4. Inisialisasi *counter* i=1.
5. Selama *counter* iterasi (i) kurang dari 16 maka dilakukan langkah 6 sampai langkah 10.
6. Perhatikan i pada table

   Jika i =1 atau i=2 atau i=9 atau i=16 maka C_{i-1} dan D_{i-1} digeser kekiri satu kali.
7. Jika i selain pada langkah 6 C_{i-1} dan D_{i-1} digeser kekiri dua kali.
8. Jalankan fungsi permutasi-2 dengan memindahkan nomor bit sesuai tabel PC-2 (tabel terlampir).
9. Membentuk kunci baru.
10. Naikan nilai satu pada *counter* i.
11. Ulangi langkah 6 sampai langkah 10 hingga *counter* membentuk 16 kunci baru.
12. Selesai.

3.4.5 *Flowchart* Enkripsi / Dekripsi DES

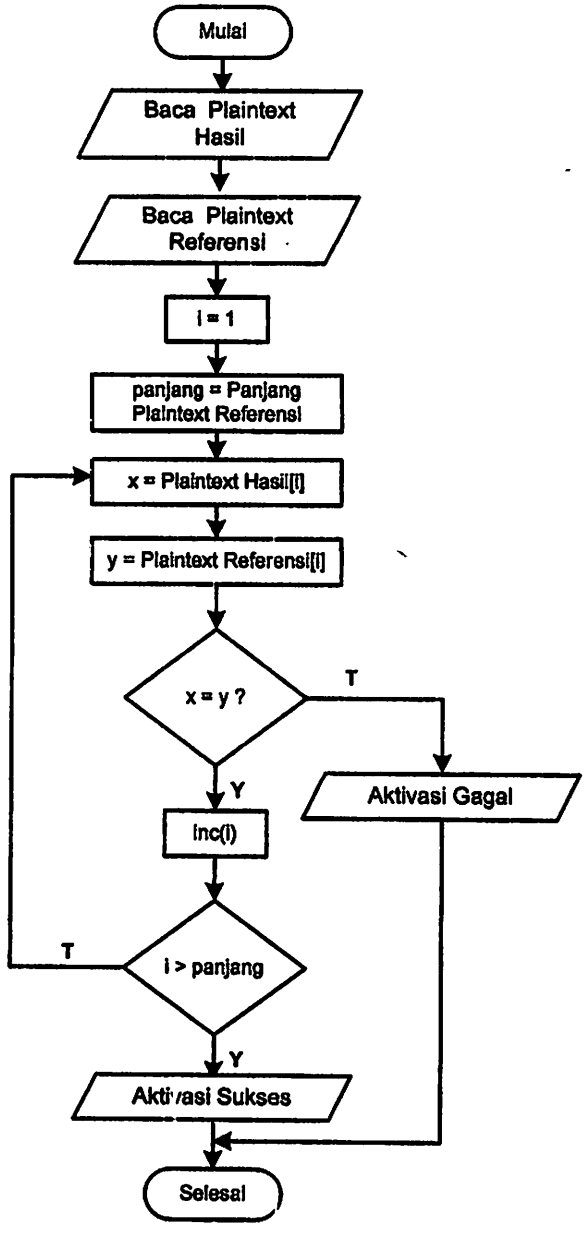


Gambar 3.10. *Flowchart* Enkripsi/Dekripsi DES pada PC

Proses dekripsi dapat dijelaskan dengan rincian algoritma sebagai berikut:

1. Masukan pesan acak *(chipertext)* 8 karakter.
2. *Plaintext* dimasukan ke fungsi permutasi pertama dengan memindahkan nomor bit sesuai tabel IP-1(tabel terlampir).
3. *Plaintext* dipecah menjadi 2 bagian L_0 (kiri) dan R_0 (kanan).
4. Inisialisasi *counter* i=1.
5. Selama *counter* iterasi (i) kurang dari 16 maka dilakukan langkah 6 sampai langkah 9.
6. L_i sama dengan R_{i-1}.
7. Rbaru sama dengan R_{i-1} dan kunci ke i (K_i) dimasukan ke fungsi chiper.
8. R_i sama dengan L_{i-1} di XOR-kan dengan Rbaru.
9. Naikan nilai satu pada *counter* i.
10. Ulangi langkah 6 sampai langkah 9 hingga *counter* iterasi i sama dengan 16.
11. R *akhir* sama dengan R_{16} dimasukan ke fungsi permutasi kedua dengan memindahkan nomor bit sesuai tabel IP-2 (tabel terlampir).
12. Output sama dengan *plaintext.*
13. Selesai.

3.4.6 *Flowchart* validasi pada PC



Gambar 3.11. *Flowchart* Validasi pada PC

Proses validasi dapat dijelaskan dengan rincian algoritma sebagai berikut:

1. Mulai.
2. Baca *Plaintext* Hasil.
3. Baca *Plaintext* Referensi.
4. Set pencancah i =1.
5. Panjang = Panjang dari karakter *Plaintext* Referensi..
6. Karakter x = karakter *Plaintext* Hasil ke-i.
7. Karakter y = karakter *Plaintext* Referensi ke-i.
8. Apakah x = y ?

   Jika Ya, maka lakukan langkah-10.

   Jika Tidak, maka lakukan langkah selanjutnya..
9. Aktivasi gagal dan lakukan langkah 13.
10. Naikan satu nilai pencacah i
11. Apakah i > panjang ?

    Jika Ya, maka lakukan langkah selanjutnya.

    Jika Tidak, maka kembali ke langkah-8.
12. Aktivasi Sukses
13. Selesai.

BAB IV

PENGUJIAN DAN SIMULASI SISTEM

4.1 Analisa dan pengujian

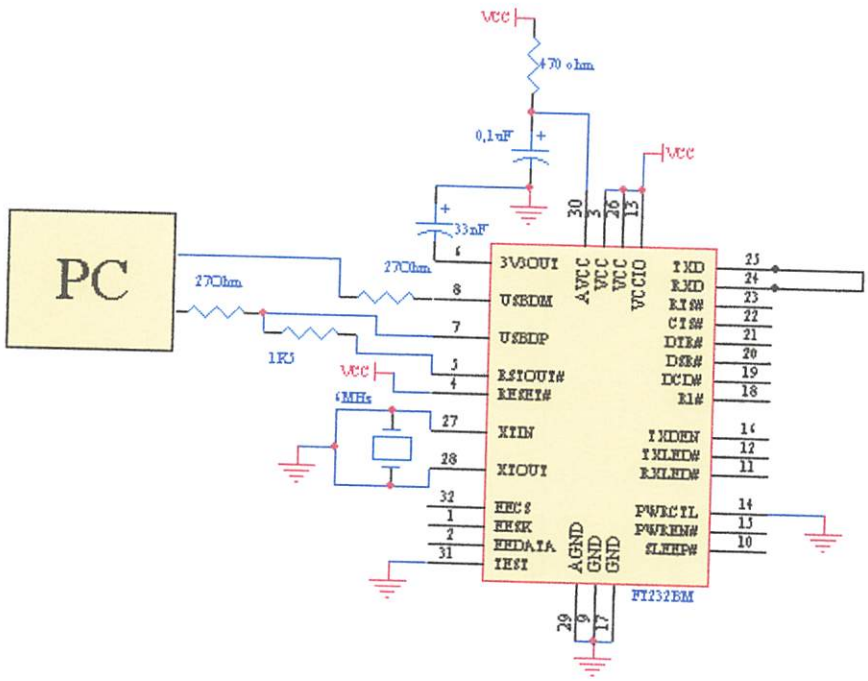
4.1.1 Pengujian Konverter *Serial* - USB

Untuk mengetahui fungsi pin Tx (pengiriman data) dan Rx(penerimaan data) pada rangkaian FT-232BM yang terdapat pada modul rangkaian konverter *serial* to USB. 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 sebagai berikut:

- Rangkaian FT232BM
- Konektor USB
- *Jumper*
- PC

Langkah-langkah dari pengujian ini dapat dijelaskan sebagai berikut:

1.Merangkai rangkaian *driver* seperti pada gambar dibawah ini:



Gambar 4.1. Rangkaian Pengujian *Serial Interface*

2. Pada gambar di atas, output FT232BM (Pin T_{XD}) dihubungkan dengan input (Pin R_{XD}), dengan demikian semua data yang dikirim melalui PC akan diumpan-balikkan ke PC lagi
3. Mengirimkan data 8 *byte* atau lebih melalui *software* tambahan untuk mengetahui apakah data yang diterima sesuai dengan data yang dikirim.

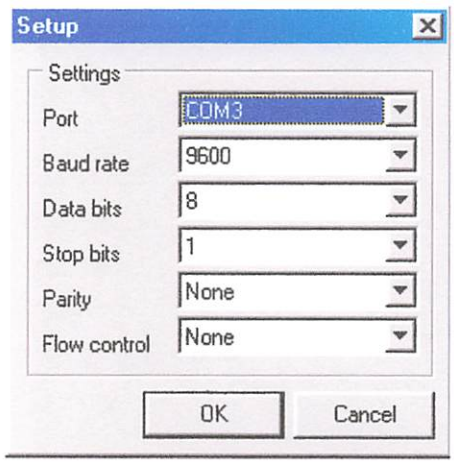
Pada pengujian ini menggunakan beberapa *parameter* pada sistem antarmuka dengan PC, yaitu:

1. *Port*, *port* yang digunakan adalah *virtual com* dengan alamat yang terdapat pada *COM3*.
2. *Baudrate*, *baudrate* yang digunakan 9600 *bit per second.*
3. *Data bits*, d*ata bits* yang digunakan memiliki ukuran panjang *frame* 8 bit.

4. *Parity,* tidak menggunakan bit *parity.*

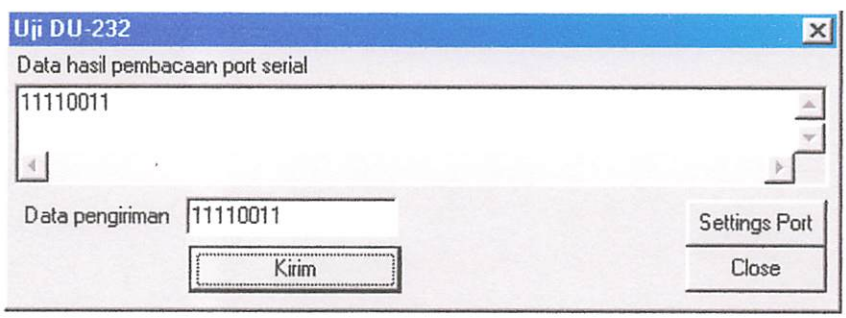
5. *Flow control,* tidak menggunakan *flow control.*

Berikut adalah *Screen Shot* dari *setting port* pada pengujian ini:



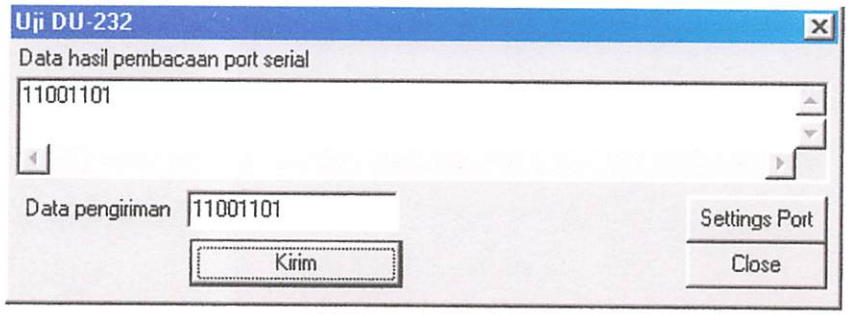
Gambar 4.2. *Setting Port* pada PC

Data yang dikirim PC 11110011 maka data yang diterima oleh PC pun sama 11110011.



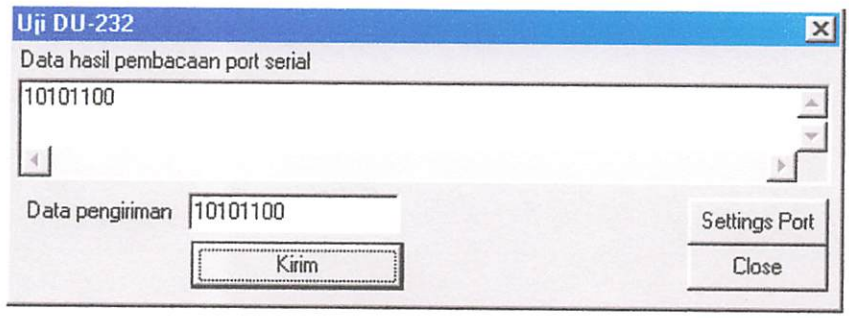
Gambar 4.3. Pengujian *Serial* data biner 11110011

Data yang dikirim PC 11001101 maka data yang diterima oleh PC pun sama 11001101.



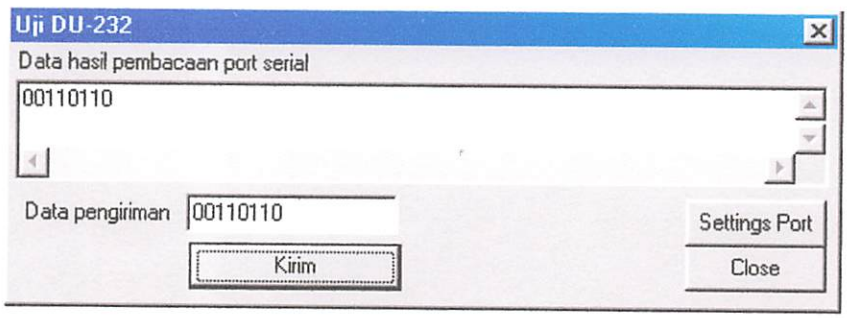
Gambar 4.4. Pengujian *Serial* Data Biner 11001101

Data yang dikirim PC 10101100 maka data yang diterima oleh PC pun sama 10101100.



Gambar 4.5. Pengujian *Serial* Data 10101100

Data yang dikirim PC 00110110 maka data yang diterima oleh PC pun sama 00110110.



Gambar 4.6. Pengujian *Serial* Data 00110110

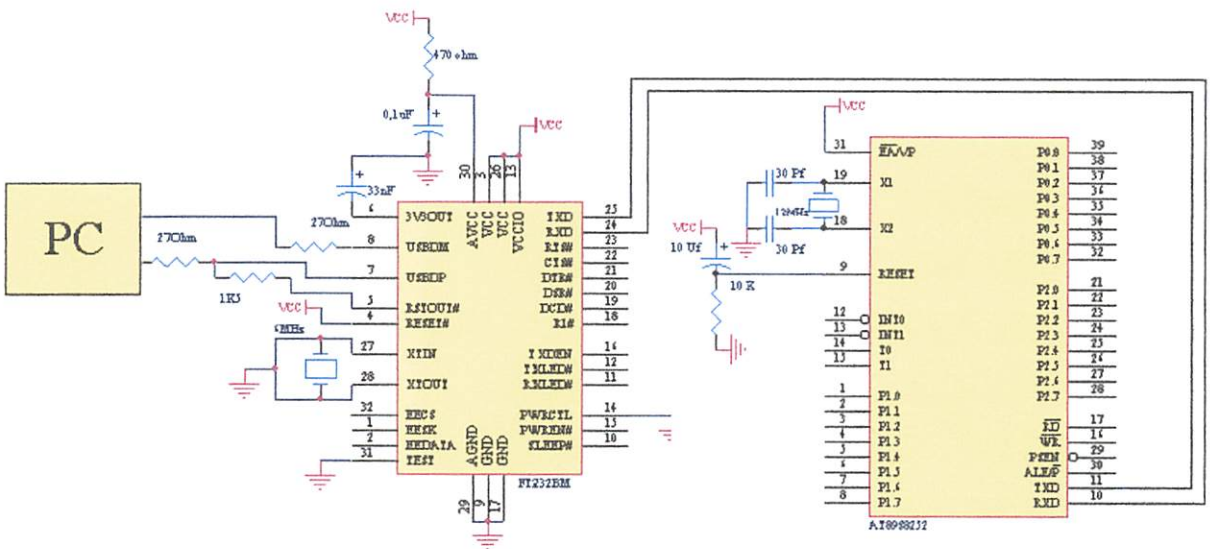
4.1.2 Pengujian Perangkat Kode Lisensi

Untuk mengetahui sistem jabat tangan PC dengan mikrokontroler apakah dapat berjalan dengan baik. Data akan dikirimkan PC kepada mikrokontroler melalui USB konverter. Peralatan yang digunakan pada pengujian ini adalah sebagai berikut:

- Rangkaian FT232BM
- Mikrokontroler
- PC

Langkah-langkah yang perlu dilakukan dalam pengujian ini adalah sebagai berikut:

1. Merangkai rangkaian *driver* seperti pada gambar dibawah ini:



Gambar 4.7. Rangkaian Pengujian Perangkat Kode Lisensi

2. Pada gambar di atas, output FT232BM (Pin T_{XD}) dihubungkan dengan Rx mikrokontroler dan Rx FT232BM di hubungkan dengan pin Tx mikrokontroler,

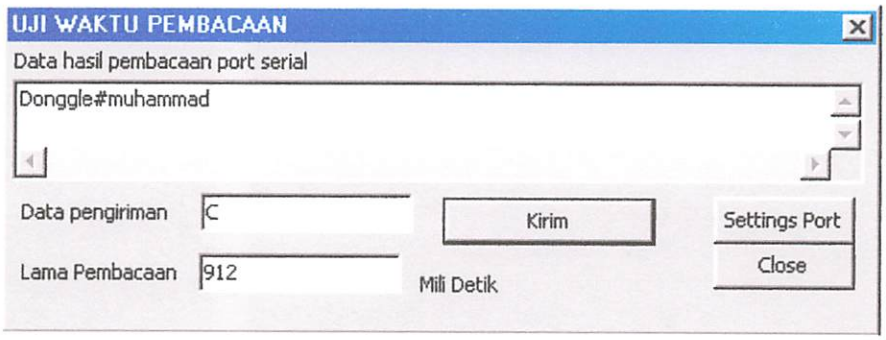
dengan demikian semua data yang dikirim melalui PC akan diterima oleh mikrokontroler atau sebaliknya.

Pada pengujian ini perangkat lunak yang digunakan adalah program aplikasi uji waktu pembacaan dan *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 *COM4*.
2. *Baudrate*, *baudrate* yang digunakan 9600 *bit per second.*
3. *Data bits*, d*ata bits* yang digunakan memiliki ukuran panjang *frame* 8 bit.
4. *Parity,* tidak menggunakan bit *parity.*
5. *Flow control,* tidak menggunakan *flow control.*

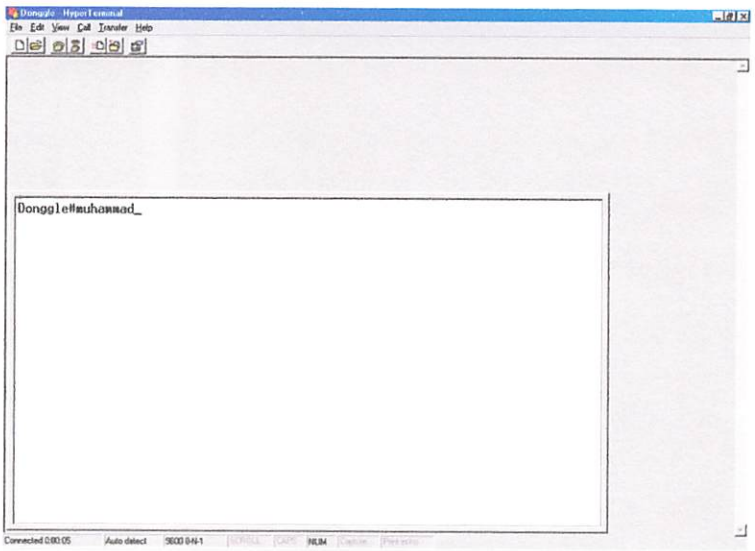
Berikut adalah s*creen shot* dari *setting port* pada pengujian ini:

Gambar 4.8. *Setting Port Serial* pada *Hyper Terminal*



Gambar 4.9. Tampilan Program Uji Waktu Pembacaan

Pada gambar diatas terlihat waktu pembacaan 16(enam belas) karakter adalah 912 milidetik, maka rasio antara waktu pembacaan(milidetik) dan panjang data(bit) adalah sebesar 128 : 912 = 1 : 7,125



Gambar 4.10. Tampilan Data *Hyper Terminal*

Pada gambar diatas mikrokontroler akan memberikan data berupa ID *hardware* dan kode lisensi jika PC akan mengirimkan karakter 'C' melalui USB.

4.1.3 Pengujian Data Tersamar

Pengujian ini dilakukan untuk melihat sejauh mana tingkat kesamaran dan keakuratan data yang telah dienkripsi dengan melakukan beberapa percobaan menggunakan *plaintext* yang sama dengan kunci yang lemah dan memiliki kemiripan.

4.1.3.1 Data *Sample* I

Pada data *sample* pertama ini *parameter* penyamaran yang diberlakukan adalah sebagai berikut:

- String Kunci : "Muhammad", dengan biner dan *ordinal* dari kunci adalah sebagai berikut:

Char	M	u	h	a	m	m	a	d
Heksa	4D	75	68	61	6D	6D	61	64
ASCII	77	117	104	97	109	109	97	100

Maka nilai biner pada ASCII tersebut akan dirangkaiakan menjadi,

0 1 0 0 1 1 0 1 | 0 1 1 1 0 1 0 1 | 0 1 1 0 1 0 0 0 | 0 1 1 0 0 0 0 1 |

0 1 1 0 1 1 0 1 | 0 1 1 0 1 1 0 1 | 0 1 1 0 0 0 0 1 | 0 1 1 0 0 1 0 0

dengan hasil penjadualan kunci adalah sebagai berikut

1 1 1 0 0 0 0 0 | 1 0 1 1 1 1 1 0 | 0 0 1 0 0 1 1 0 | 0 0 1 1 0 0 0 0 |

1 0 0 0 0 1 1 0 | 0 0 1 0 1 1 0 0

- String *Plaintext* : "12345678", dengan biner dan *ordinal* ASCII dari *plaintext*

Char	1	2	3	4	5	6	7	8
Heksa	31	32	33	34	35	36	37	38
ASCII	49	50	51	52	53	54	55	56

Maka nilai biner pada ASCII tersebut akan dirangkaikan menjadi,

0 0 1 1 0 0 0 1 | 0 0 1 1 0 0 1 0 | 0 0 1 1 0 0 1 1 | 0 0 1 1 0 1 0 0 |

0 0 1 1 0 1 0 1 | 0 0 1 1 0 1 1 0 | 0 0 1 1 0 1 1 1 | 0 0 1 1 1 0 0 0

Setelah diterapkannya proses enkripsi pada *parameter* diatas maka didapatkan *ciphertext* sebagai berikut:

Char	<	=	>	?	@	A	B	3
Heksa	3C	3D	3E	3F	40	41	42	33
ASCII	60	61	62	63	64	65	66	51

Dengan hubungan rangkaian biner sebagai berikut:

0 0 1 1 1 1 0 0 | 0 0 1 1 1 1 0 1 | 0 0 1 1 1 1 1 0 | 0 0 1 1 1 1 1 1 |

0 1 0 0 0 0 0 0 | 0 1 0 0 0 0 0 1 | 0 1 0 0 0 0 1 0 | 0 0 1 1 0 0 1 1 |

4.1.3.2 Data *Sample* II

Pada data *sample* pertama ini *parameter* penyamaran yang diberlakukan adalah sebagai berikut:

- String Kunci : "muhammad", dengan biner dan *ordinal* dari kunci adalah sebagai berikut:

Char	m	u	h	a	m	m	a	d
Heksa	6D	75	68	61	6D	6D	61	64
ASCII	109	117	104	97	109	109	97	100

Maka nilai biner pada ASCII tersebut akan dirangkaikan menjadi,

0 1 1 0 1 1 0 1 | 0 0 1 1 0 0 1 0 | 0 0 1 1 0 0 1 1 | 0 0 1 1 0 1 0 0 |

0 0 1 1 0 1 0 1 | 0 0 1 1 0 1 1 0 | 0 0 1 1 0 1 1 1 | 0 0 1 1 1 0 0 0

dengan hasil penjadualan kunci adalah sebagai berikut

1 1 1 1 0 0 0 0 | 1 0 1 1 1 1 1 0 | 0 0 1 0 0 1 1 0 | 0 0 1 1 0 0 0 0 |

1 0 0 0 0 1 1 0 | 0 0 1 0 1 1 0 0

- String *Plaintext* : "12345678", dengan biner dan *ordinal* ASCII dari *plaintext*

Char	1	2	3	4	5	6	7	8
Heksa	31	32	33	34	35	36	37	38
ASCII	49	50	51	52	53	54	55	56

Maka nilai biner pada ASCII tersebut akan dirangkaikan menjadi,

0 0 1 1 0 0 0 1 | 0 0 1 1 0 0 1 0 | 0 0 1 1 0 0 1 1 | 0 0 1 1 0 1 0 0 |

0 0 1 1 0 1 0 1 | 0 0 1 1 0 1 1 0 | 0 0 1 1 0 1 1 1 | 0 0 1 1 1 0 0 0

Setelah diterapkannya proses enkripsi pada *parameter* diatas maka didapatkan *ciphertext* sebagai berikut:

Char	`	a	b	[	\	]	^	W
Heksa	60	61	62	5B	5C	5D	5E	57
ASCII	96	97	98	91	92	93	94	87

Dengan rangkaian biner sebagai berikut:

0 1 1 0 0 0 0 0 | 0 1 1 0 0 0 0 1 | 0 1 1 0 0 0 1 0 | 0 1 0 1 1 0 1 1 |

0 1 0 1 1 1 0 0 | 0 1 0 1 1 1 0 1 | 0 1 0 1 1 1 1 0 | 0 1 0 1 0 1 1 1 |

4.1.3.3 Analisa *Sample* I dan *Sample* II

Pada *sample* I dan II *plaintext* yang digunakan adalah identik atau sama, perbedaan *parameter* pada *sample* I dan II adalah terletak pada string kunci yang digunakan, yaitu:

- Kunci *sample* I

Char	M	U	h	a	m	m	a	d
Heksa	4D	75	68	61	6D	6D	61	64
ASCII	77	117	104	97	109	109	97	100

- Kunci *sample* II

Char	m	U	h	a	m	m	a	d
Heksa	6D	75	68	61	6D	6D	61	64
ASCII	109	117	104	97	109	109	97	100

Dapat dilihat perbedaan pada kedua kunci tersebut hanya pada karakter pertama pada kedua kunci yaitu "M" dan "m" dengan nilai *ordinal* 77 dan 109. Dikarenakan DES adalah algoritma yang bersifat *case sensitife* maka *ciphertext* hasil enkripsi dengan kedua kunci akan memiliki perbedaan, berikut adalah biner *ciphertext* dari kedua operasi tersebut.

Biner *Ciphertext sample* I dan II bit ke 0 - 31

Sample I → 0 0 1 1 1 1 0 0 | 0 0 1 1 1 1 0 1 | 0 0 1 1 1 1 1 0 | 0 0 1 1 1 1 1 1 |

Sample II → 0 1 1 0 0 0 0 0 | 0 1 1 0 0 0 0 1 | 0 1 1 0 0 0 1 0 | 0 1 0 1 1 0 1 1 |

Maka dari data diatas dapat dilihat jumlah bit yang berbeda antara bit 0-31 adalah 15 bit.

Biner *Ciphertext sample* I dan II bit ke 32 - 63

Sample I → 0 1 0 0 0 0 0 0 | 0 1 0 0 0 0 0 1 | 0 1 0 0 0 0 1 0 | 0 0 1 1 0 0 1 1 |

Sample II → 0 1 0 1 1 1 0 0 | 0 1 0 1 1 1 0 1 | 0 1 0 1 1 1 1 0 | 0 1 0 1 0 1 1 1 |

Maka dari data diatas dapat dilihat jumlah bit yang berbeda antara bit 32-63 adalah 12 bit.

Maka jumlah total perbedaan bit adalah :

15 + 12 = 27 bit,

Sedangkan jumlah total bit yang identik adalah :

64 – 27 = 37 bit,

Maka dapat dikatakan untuk perbedaan satu karakter pada kunci terdapat perbedaan *ciphertext* sebesar

$$\frac{27}{64} \times 100\% = 42{,}875\%$$

4.1.3.4 Data *Sample* III

Pada data *sample* ini *parameter* penyamaran yang diberlakukan adalah sebagai berikut:

- String Kunci : "ABCDEFGH", dengan biner dan *ordinal* dari kunci adalah sebagai berikut:

Char	A	B	C	D	E	F	G	H
Heksa	41	42	43	44	45	46	47	48
ASCII	65	66	67	68	69	70	71	72

Maka nilai biner pada ASCII tersebut akan dirangkaiakan menjadi,

0 1 0 0 0 0 0 1 | 0 1 0 0 0 0 1 0 | 0 1 0 0 0 0 1 1 | 0 1 0 0 0 1 0 0 |

0 1 0 0 0 1 0 1 | 0 1 0 0 0 1 1 0 | 0 1 0 0 0 1 1 1 | 0 1 0 0 1 0 0 0

dengan hasil penjadualan kunci adalah sebagai berikut

1 0 1 0 0 0 0 0 | 1 0 0 1 0 0 1 0 | 0 0 1 0 0 0 1 0 | 0 1 1 0 0 0 1 0 |

0 1 0 1 0 0 1 0 | 0 0 0 1 0 0 1 1

- String *Plaintext* : "12345678", dengan biner dan *ordinal* ASCII dari *plaintext*

Char	1	2	3	4	5	6	7	8
Heksa	31	32	33	34	35	36	37	38
ASCII	49	50	51	52	53	54	55	56

Maka nilai biner pada ASCII tersebut akan dirangkaikan menjadi,

0 0 1 1 0 0 0 1 | 0 0 1 1 0 0 1 0 | 0 0 1 1 0 0 1 1 | 0 0 1 1 0 1 0 0 |

0 0 1 1 0 1 0 1 | 0 0 1 1 0 1 1 0 | 0 0 1 1 0 1 1 1 | 0 0 1 1 1 0 0 0

Setelah diterapkannya proses enkripsi pada *parameter* diatas maka didapatkan *ciphertext* sebagai berikut:

Char	4	5	6	7	8	9	:	;
Heksa	34	35	36	37	38	39	3A	3B
ASCII	52	53	54	55	56	57	58	59

Dengan hubungan rangkaian biner sebagai berikut:

0 0 1 1 0 1 0 0 | 0 0 1 1 0 1 0 1 | 0 0 1 1 0 1 1 0 | 0 0 1 1 0 1 1 1 |

0 0 1 1 1 0 0 0 | 0 0 1 1 1 0 0 1 | 0 0 1 1 1 0 1 0 | 0 0 1 1 1 0 1 1 |

4.1.3.5 Data *Sample* IV

Pada data *sample* ini *parameter* penyamaran yang diberlakukan adalah sebagai berikut:

- String Kunci : "abcdefgh", dengan biner dan *ordinal* dari kunci adalah sebagai berikut:

Char	a	b	c	d	e	f	g	H
Heksa	41	42	63	44	45	46	47	48
ASCII	65	66	99	68	69	70	71	72

Maka nilai biner pada ASCII tersebut akan dirangkaikan menjadi,

0 1 0 0 0 0 0 1 | 0 1 0 0 0 0 1 0 | 0 1 1 0 0 0 1 1 | 0 1 0 0 0 1 0 0 |

0 1 0 0 0 1 0 1 | 0 1 0 0 0 1 1 0 | 0 1 0 0 0 1 1 1 | 0 1 0 0 1 0 0 0

dengan hasil penjadualan kunci adalah sebagai berikut

1 0 1 0 0 0 0 0 | 1 0 0 1 0 0 1 0 | 0 0 1 0 0 0 1 0 | 0 1 1 0 0 0 1 0 |

0 1 0 1 0 0 1 0 | 0 0 0 1 0 0 1 1

- String *Plaintext* : "12345678", dengan biner dan *ordinal* ASCII dari *plaintext*

Char	1	2	3	4	5	6	7	8
Heksa	31	32	33	34	35	36	37	38
ASCII	49	50	51	52	53	54	55	56

Maka nilai biner pada ASCII tersebut akan dirangkaikan menjadi,

0 0 1 1 0 0 0 1 | 0 0 1 1 0 0 1 0 | 0 0 1 1 0 0 1 1 | 0 0 1 1 0 1 0 0 |

0 0 1 1 0 1 0 1 | 0 0 1 1 0 1 1 0 | 0 0 1 1 0 1 1 1 | 0 0 1 1 1 0 0 0

Setelah diterapkannya proses enkripsi pada *parameter* diatas maka didapatkan *ciphertext* sebagai berikut:

Setelah diterapkannya proses enkripsi pada *parameter* diatas maka didapatkan *ciphertext* sebagai berikut:

Char	<	=	>	?	@	A	B	3
Heksa	3C	3D	3E	3F	40	41	42	33
ASCII	60	61	62	63	64	65	66	51

Dengan hubungan rangkaian biner sebagai berikut:

0 0 1 1 1 1 0 0 | 0 0 1 1 1 1 0 1 | 0 0 1 1 1 1 1 0 | 0 0 1 1 1 1 1 1 |

0 1 0 0 0 0 0 0 | 0 1 0 0 0 0 0 1 | 0 1 0 0 0 0 1 0 | 0 0 1 1 0 0 1 1 |

4.1.3.6 Analisa Data *Sample* III dan IV

Pada *sample* III dan IV *plaintext* yang digunakan adalah identik atau sama, perbedaan *parameter* pada *sample* III dan IV adalah terletak pada string kunci yang digunakan, yaitu:

- Kunci *sample* III

Char	A	B	C	D	E	F	G	H
Heksa	41	42	43	44	45	46	47	48
ASCII	65	66	67	68	69	70	71	72

- Kunci *sample* IV

Char	a	b	c	d	e	f	g	H
Heksa	41	42	63	44	45	46	47	48
ASCII	65	66	99	68	69	70	71	72

String kunci yang digunakan pada kedua *sample* termasuk dalam kategori kunci lemah, dikarenakan karakter kunci yang digunakan memiliki nilai *ordinal* yang relatif berurutan.

Dapat dilihat perbedaan pada kedua kunci tersebut hanya pada besar atau kecil karakter dengan nilai *ordinal* 67 dan 99. Dikarenakan DES adalah bersifat *case sensitife* maka *ciphertext* hasil enkripsi dengan kedua kunci akan memiliki perbedaan, berikut adalah biner *ciphertext* dari kedua operasi tersebut.

Biner *Ciphertext sample* III dan IV bit ke 0 - 31

Sample III → 0 0 1 1 0 1 0 0 | 0 0 1 1 0 1 0 1 | 0 0 1 1 0 1 1 0 | 0 0 1 1 0 1 1 1 |

Sample IV → 0 0 1 1 1 1 0 0 | 0 0 1 1 1 1 0 1 | 0 0 1 1 1 1 1 0 | 0 0 1 1 1 1 1 1 |

Maka dari data diatas dapat dilihat jumlah bit yang berbeda antara bit 0-31 adalah 4 bit.

Biner *Ciphertext sample* III dan IV bit ke 32 - 63

Sample III → 0 0 1 1 1 0 0 0 | 0 0 1 1 1 0 0 1 | 0 0 1 1 1 0 1 0 | 0 0 1 1 1 0 1 1 |

Sample IV → 0 1 0 0 0 0 0 0 | 0 1 0 0 0 0 0 1 | 0 1 0 0 0 0 1 0 | 0 0 1 1 0 0 11 |

Maka dari data diatas dapat dilihat jumlah bit yang berbeda antara bit 32-64 adalah 13 bit.

Maka jumlah total perbedaan bit adalah :

13 + 4 = 17 bit,

Sedangkan jumlah total bit yang identik adalah :

64 – 17 = 47 bit.

Maka dapat dikatakan untuk perbedaan karakter berurut pada kunci yang tergolong lemah terdapat perbedaan bit *ciphertext* sebesar :

$$\frac{17}{64} \times 100\% = 26{,}56\%$$

4.2 Simulasi Program Aplikasi

Tahap ini adalah penjelasan secara detail tentang rincian proses yang akan dijalankan pada aplikasi pengamanan kode lisensi *software* ini.

4.2.1 Tahapan Proses

Pada aplikasi ini tahapan-tahapan proses yang akan dijalankan pada program PC adalah sebagai berikut:

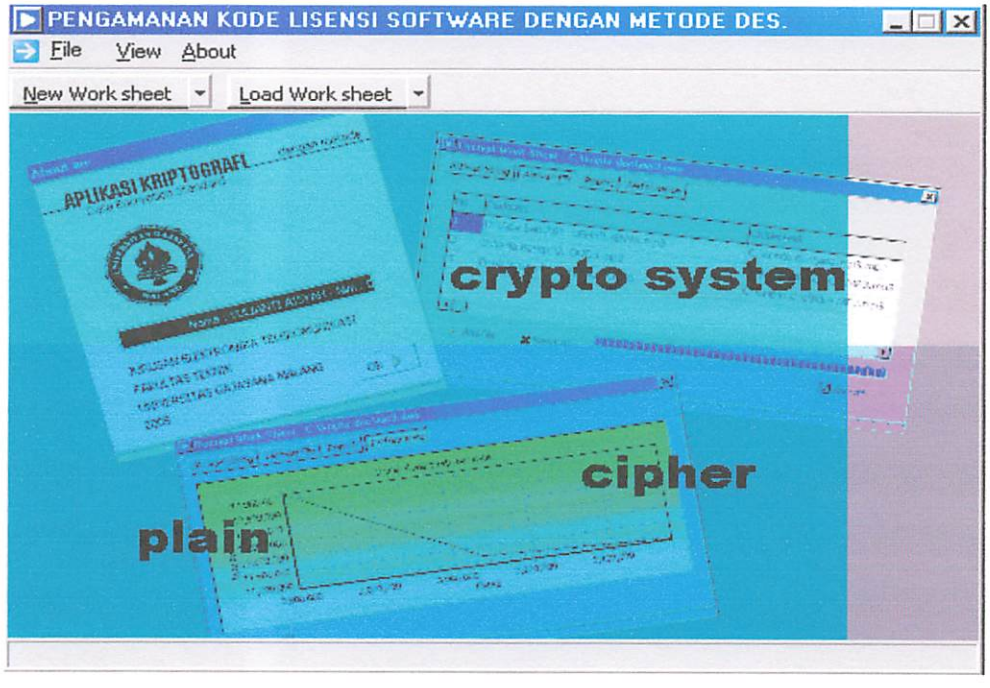
1. Jika proses aktivasi belum dilakukan maka, program pada PC akan memberikan kotak dialog peringatan pada *user*, berikut adalah visualisasi dari proses pada aplikasi ini.

Gambar 4.11. Kotak Dialog Peringatan Aktivasi *Software*

2. *Form* berikutnya yang akan ditampilkan adalah "*Form About author*". Berikut adalah *Screen Shot* dari *Form* ini.

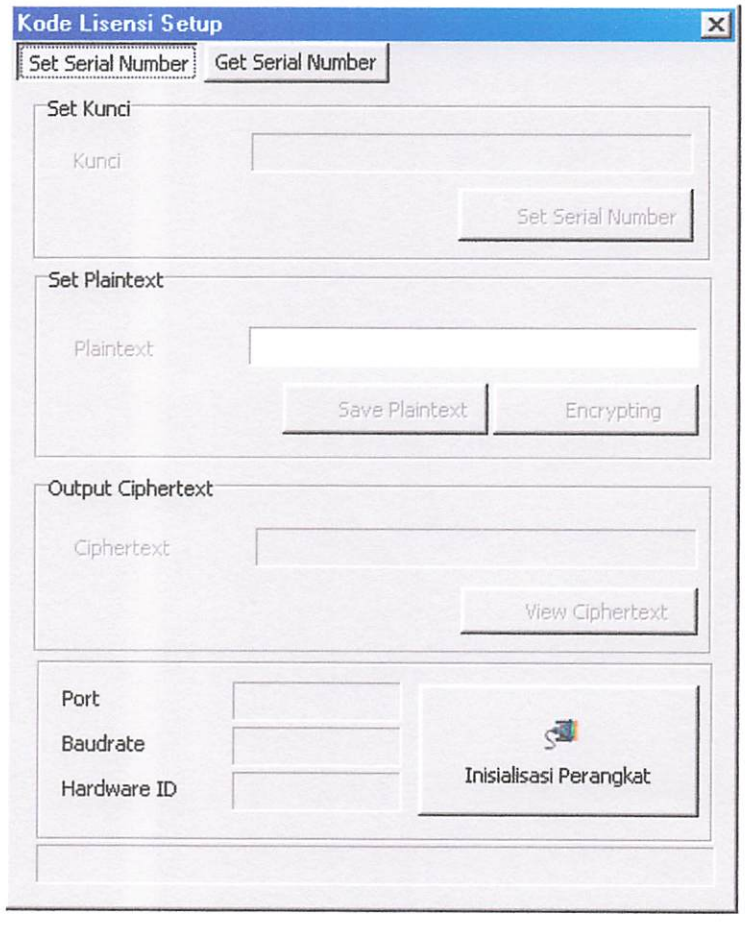
Gambar 4.12. Tampilan *About Box*

3. "*Form* utama" akan dimunculkan dengan meng-*klik* tombol "*OK*" pada *Form* ini. Berikut adalah tampilan dari "*form utama*".



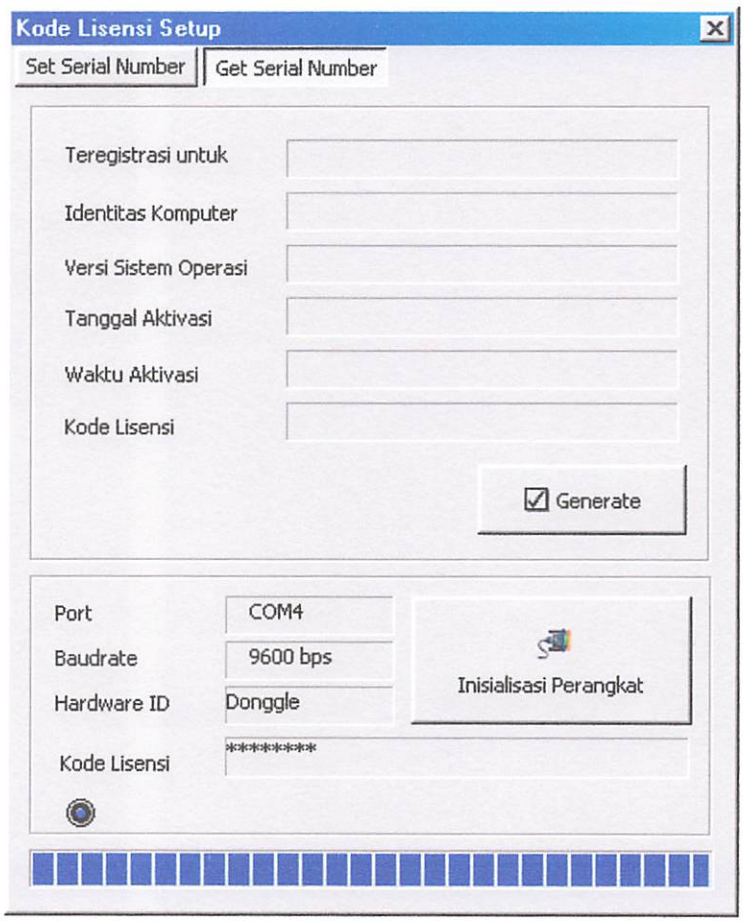
Gambar 4.13. Tampilan *Form* Utama

4. Setelah *Form* utama telah dimunculkan maka proses Aktivasi *Setup* dapat diterapkan. Penerapan proses ini di implementasikan pada menu "Kode Lisensi *Setup*" dengan memilih menu "*About*"→"*Kode Lisensi Setup*". Berikut adalah tampilan dari "*form kode lisensi setup*".



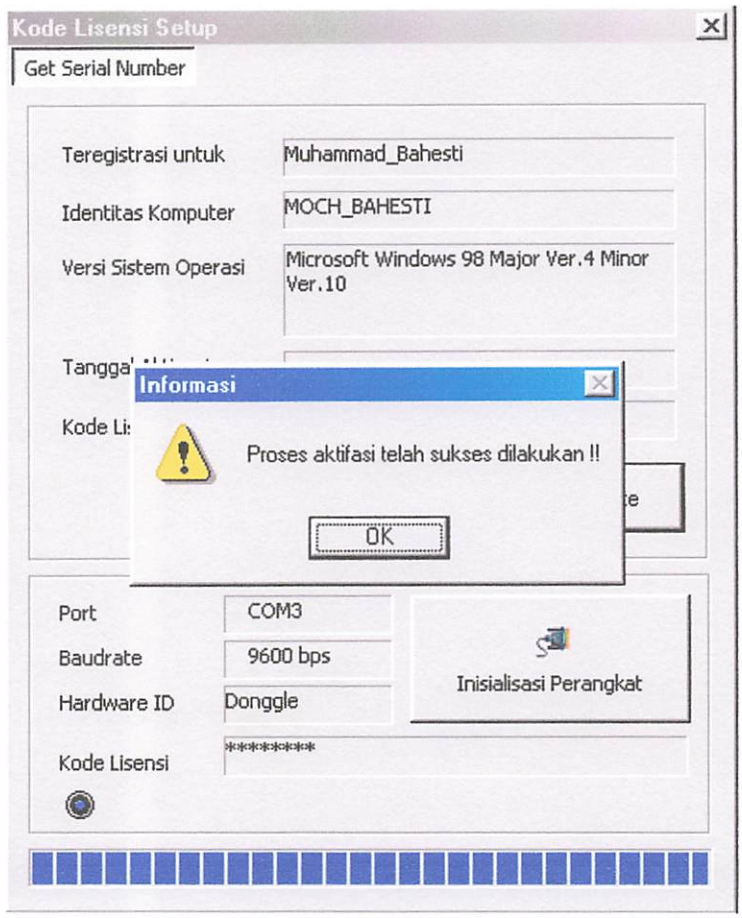
Gambar 4.14. Tampilan *Form* Kode Lisensi *Setup*

5. *User* dapat melakukan pembacaan kode lisensi pada perangkat keras lisensi dengan mengakses fungsi pada tombol "*Inisialisasi Perangkat*".
6. Setelah pembacaan kode dilakukan maka proses aktivasi akan dilakukan,berikut adalah tampilannya.



Gambar 4.15. *Screen Shot* Proses Inisialisasi Perangkat

7. Kode Lisensi yang tertera pada *textbox* "*Kode Lisensi*" akan dijalankan sebagai *parameter* pada fungsi aktivasi *software*, user dapat menjalankan proses tersebut dengan menjalankan tombol "*Generate*".
8. Jika proses aktivasi sukses maka program akan mengaktifkan *fitur-fitur* utama pada program aplikasi tersebut. Berikut adalah tampilannya



Gambar 4.16. Tampilan Proses Aktivasi *Software*

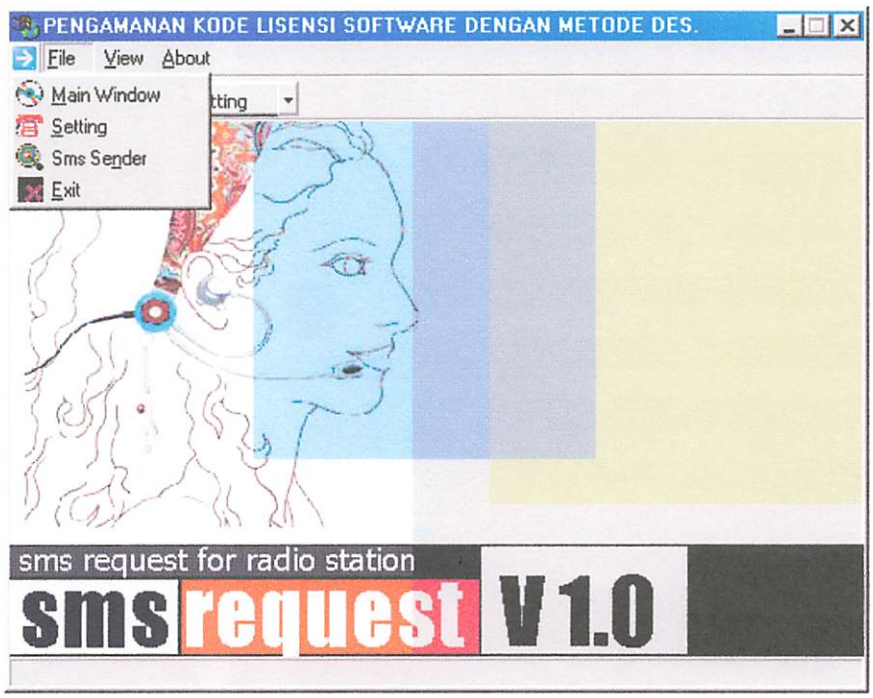
9. Jika proses aktivasi gagal maka program akan mengnonaktifkan fitur-fitur dan mengakhiri proses.

4.2.2 Proses Enkripsi

Proses enkripsi ini akan melakukan penguncian pada kode lisensi *software* "*SMS REQUEST V 1.0*" yang akan dijadikan sebagai data percobaan.

Berikut adalah penjelasan langkah dari proses penguncian kode lisensi:

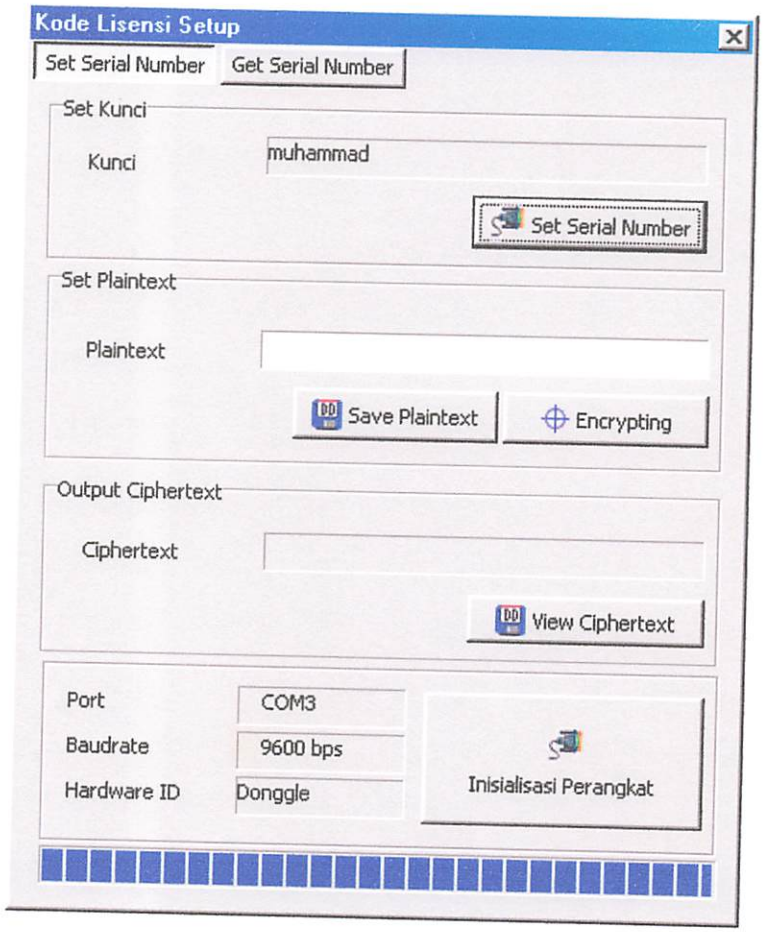
1. Jalankan program yang akan di protek.



Gambar 4.17. *Screen Shot* Program SMS REQUEST

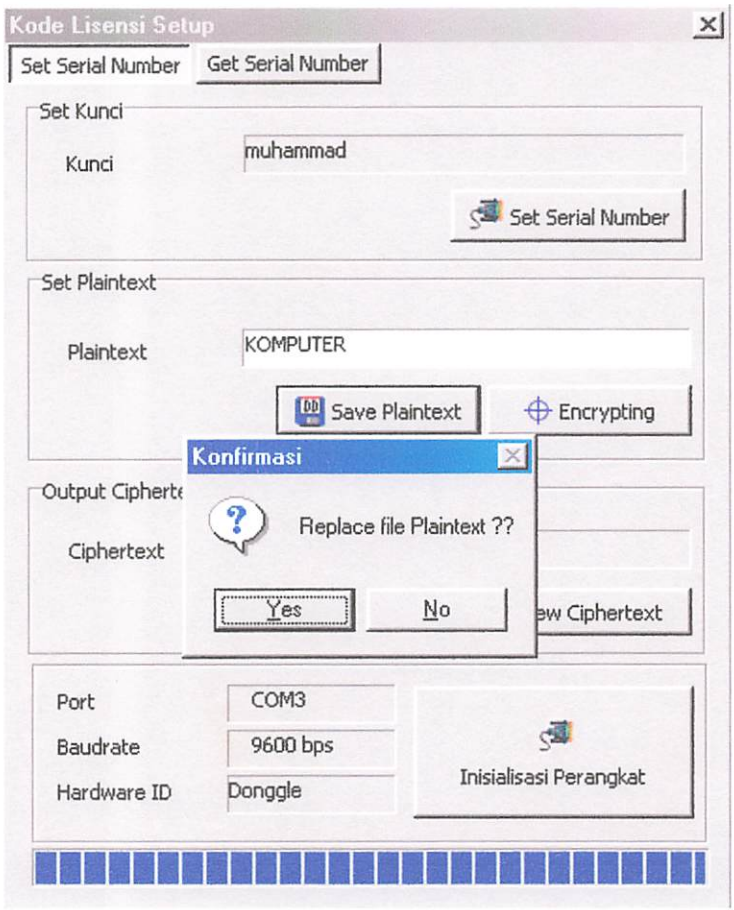
Pada tampilan diatas terlihat menu utama "*file*" masih dapat dijalankan, hal tesebut dikarenakan proses penguncian belum dilakukan.

2. Koneksikan perangkat kode lisensi pada port USB PC.
3. Pilih menu "*About*" "→"*Kode Lisensi Setup*" untuk menampilkan "*form kode lisensi setup*".
4. Pilih *tabsheet* "*Set Serial Number*".
5. Lakukan inisialisasi perangkat kode lisensi dengan menerapkan fungsi pada tombol "*Inisialisasi Perangkat*".



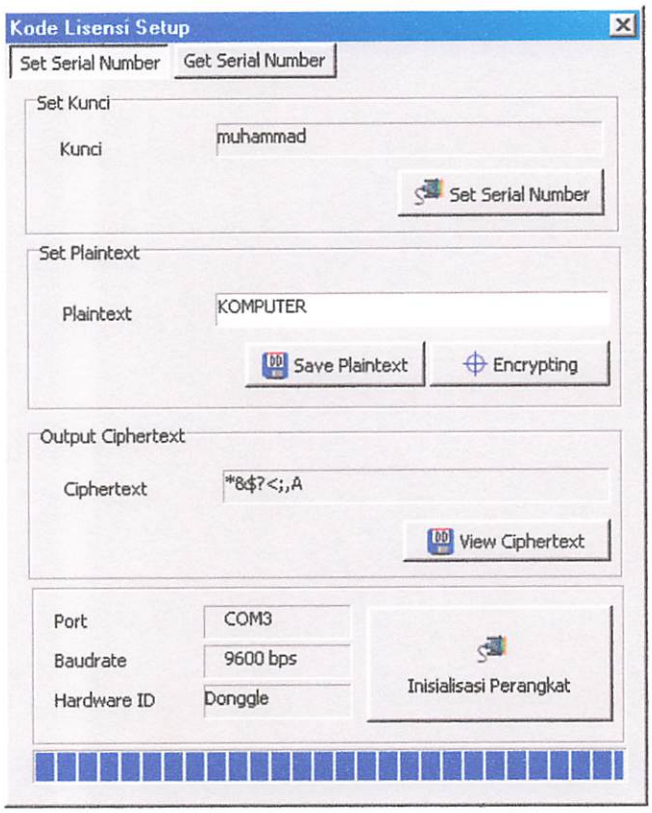
Gambar 4.18. *Screen Shot* Proses Inisialisasi Perangkat Set Kode Lisensi

6. Tekan tombol "*Set Serial Number*" untuk mendapatkan kode lisensi yang terdapat pada perangkat kode lisensi.
7. Isikan string *plaintext* pada *text box plaintext* yang tersedia.
8. Simpan *plaintext* pada yang telah diisikan dengan menekan tombol "*Save Plaintext*".

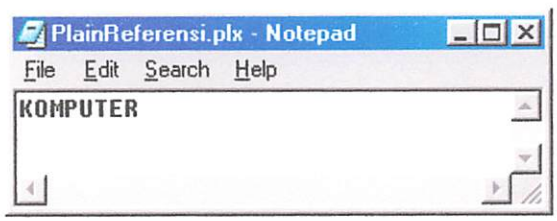


Gambar 4.19. *Screen Shot* Proses Simpan *Plaintext*

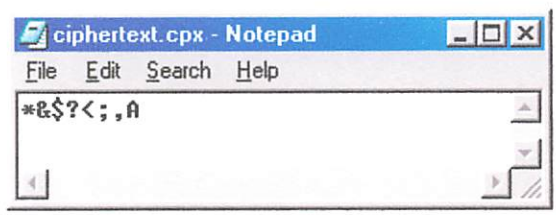
9. Lakukan penguncian *software* dengan mengaplikasikan fungsi pada tombol "*Encrypting*".
10. Tekan tombol "*View ciphertext*" untuk melihat *ciphertext* yang terbentuk dari hasil enkripsi.



Gambar 4.20. *Screen Shot* Proses *View Ciphertext*

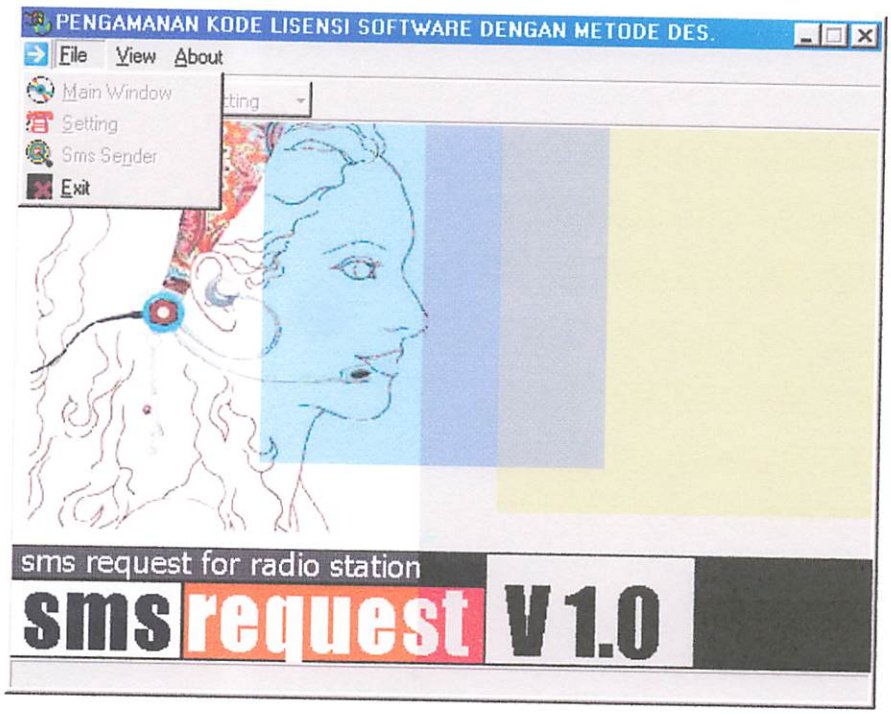


Gambar 4.21. *Plaintext*Referensi.plx yang terbentuk



Gambar 4.22. *Ciphertext*.cpx yang terbentuk

11. Tutup program aplikasi untuk melihat apakah penguncian sukses dilakukan.



Gambar 4.23. Tampilan Program Setelah dilakukan Enkripsi

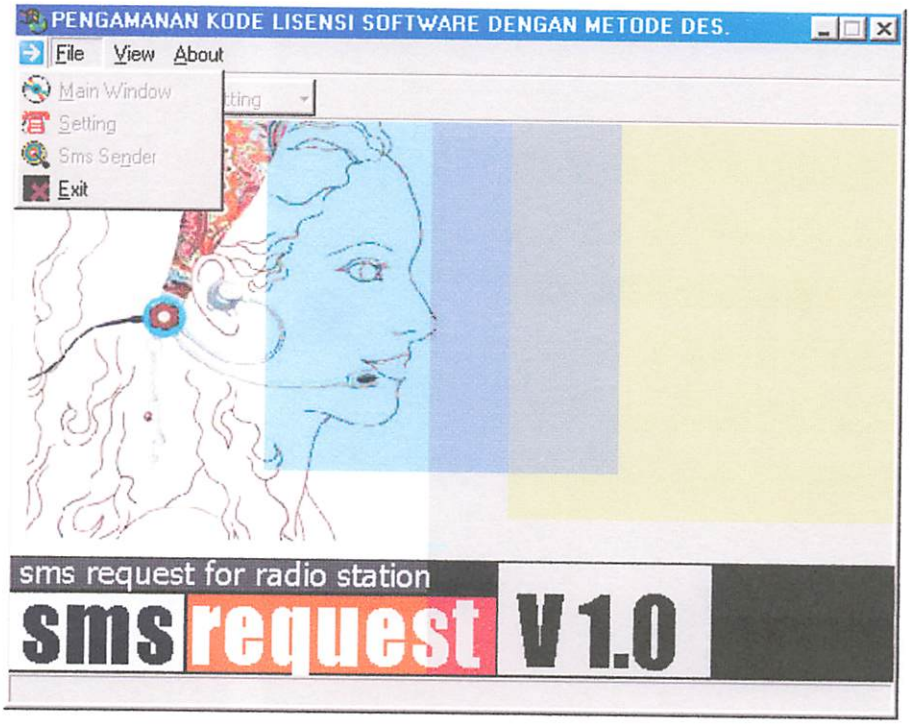
Berdasarkan *preview* diatas maka dapat dilihat setelah proses enkripsi dilakukan, menu atau fitur-fitur pada aplikasi tersebut tidak dapat dijalankan. Dengan kata lain proses enkripsi telah berhasi dilakukan.

4.2.3 Proses Dekripsi

Proses dekripsi ini dilakukan untuk menjalankan proses aktivasi *software* "*SMS REQUEST V 1.0*" yang telah di *protect*.

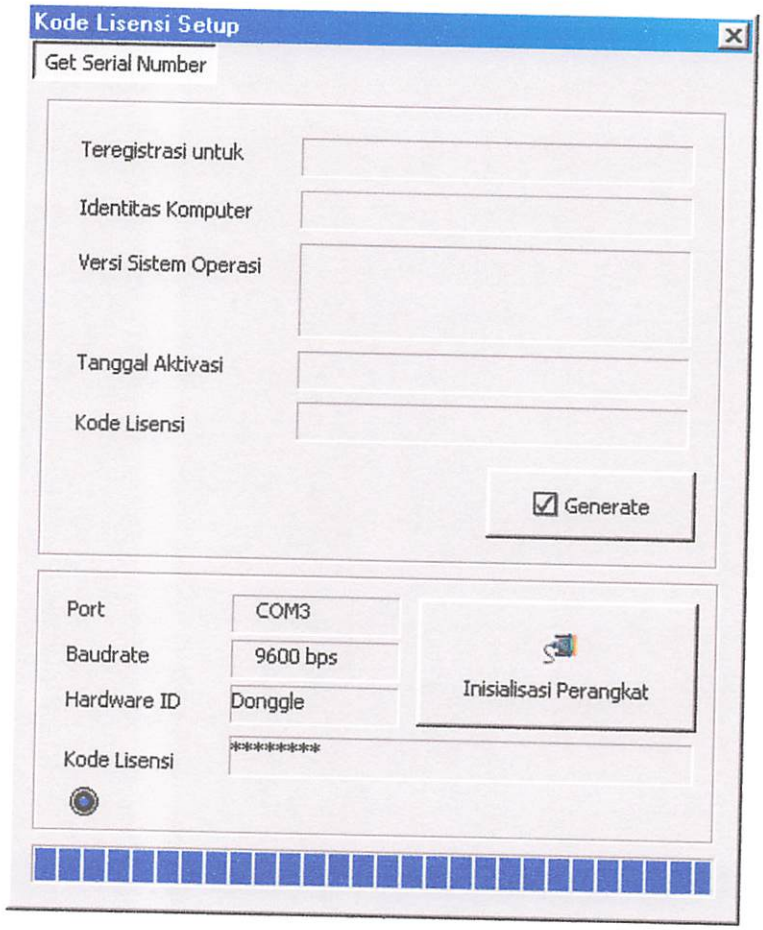
Berikut adalah penjelasan langkah dari proses penguncian kode lisensi:

1. Jalankan program yang akan di aktifkan.



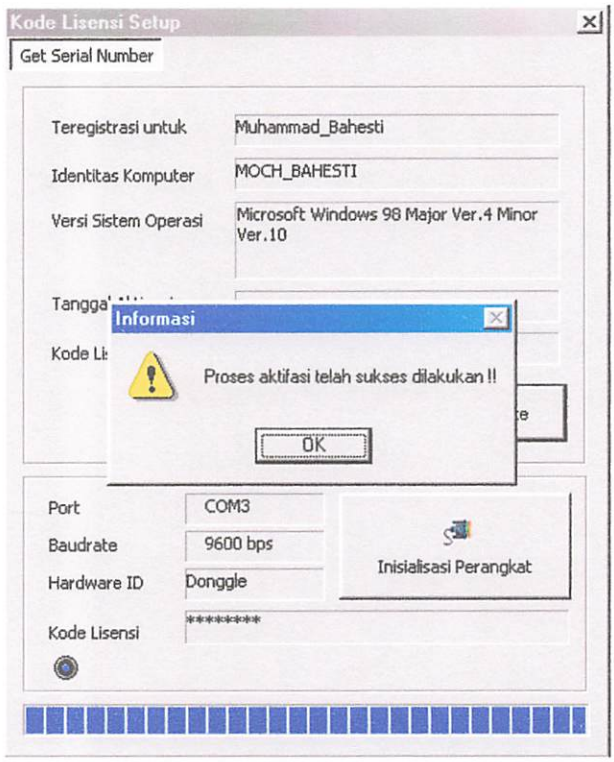
Gambar 4.24. Tampilan Program Sebelum dilakukan Aktivasi *Software*

2. Koneksikan perangkat kode lisensi pada port USB PC.
3. Pilih menu "*About*" "→"*Kode Lisensi Setup*" untuk menampilkan "*form kode lisensi setup*".
4. Pilih *tabsheet* "*Get Serial Number*".
5. Lakukan inisialisasi perangkat kode lisensi dengan menerapkan fungsi pada tombol "*Inisialisasi Perangkat*".

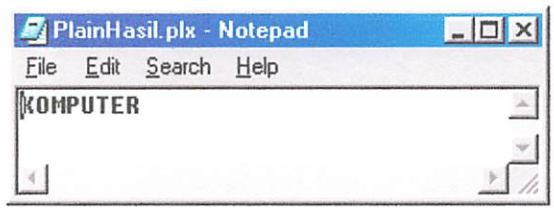


Gambar 4.25. *Screen Shot* Proses Inisialisasi Perangkat *Get* Kode Lisensi

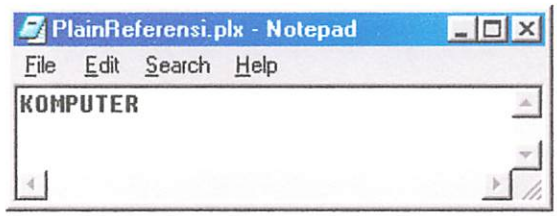
6. Terapkan proses aktivasi dengan menjalankan fungsi pada tombol "*Generate*"



Gambar 4.26. *Screen Shot* Proses Aktivasi *Software*

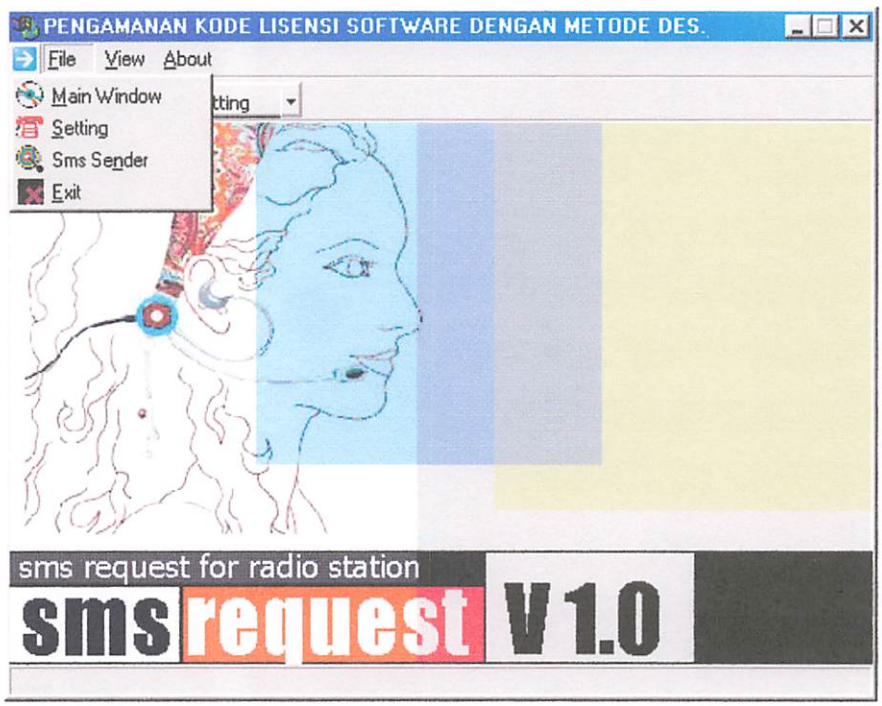


Gambar 27. *Plaintext*Hasil.plx yang terbentuk



Gambar 28. *Plaintext*Referensi.plx sebagai *plaintext* acuan

7. Tutup program aplikasi untuk melihat apakah aktivasi sukses dilakukan.



Gambar 4.29. Tampilan Program Setelah dilakukan Aktivasi *Software*

Berdasarkan *form preview* diatas dapat dilihat setelah proses dekripsi atau aktivasi *software* dilakukan, maka menu atau fitur-fitur pada aplikasi tersebut telah dapat dijalankan. Dengan kata lain proses dekripsi telah berhasil dilakukan.

BAB V

KESIMPULAN

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. Pada pengujian hardware maka dapat dikatakan waktu komunikasi antara perangkat keras dan PC cukup cepat dengan perbandingan antara panjang data dan waktu pembacaan(milidetik) sebesar 1 : 7,125.
3. Pada sisi *software* untuk hasil penyamaran dengan *plaintext* yang sama dengan perbedaan satu karakter besar/kecil huruf pada kunci terdapat perbedaan bit *ciphertext* sebesar 42,875%.
4. Pada sisi *software* untuk hasil penyamaran dengan *plaintext* yang sama dengan karakter kunci berurut dan perbedaan pada besar/kecil huruf terdapat perbedaan bit *ciphertext* sebesar 26,56%.

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. Untuk pengembangan sistem selanjutnya kunci atau kode lisensi yang tersimpan dalam perangkat keras sebaiknya telah disamarkan terlebih dahulu.
3. Untuk pengembangan lebih lanjut diharapkan *software* yang dirancang dapat berfungsi sebagai *plug-in* untuk program aplikasi lain agar sistem dapat bersifat *fleksible*.

DAFTAR PUSTAKA

[1] Atmel, *AT89S8252 Data Sheet*, Desember 1997.

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

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

[4] Rhee, Man Young.1994. *Cryptography And Secure Communication.* Singapore: McGraw-Hill Companies

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

[6] Antony Pranata, 2002, *Pemrograman Borland Delphi 7*, Edisi 4, Penerbit: Andi, Yogyakarta.

[7] Moh. Ibnu Malik, ST, 2003, *Belajar Mikrokontroller ATMEL AT890S8252, Edisi* Pertama, Penerbit: Gava Media, Yogyakarta.

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

LAMPIRAN

```
;=================================================
; Program assembler Donggle Serial  Kunci "Muhammad"
; dengan baud rate 9600 bps  X-TAL = 11.0592 MHz
; Oleh : Muhammad Bahesti Syuaiban – 06.12.918
:INSTITUT TEKNOLOGI NASIONAL MALANG
;=================================================
        org     0h

mulai: setb     EA
        mov     TMOD,#20h
        mov     TH1,#0FDH
        setb    TR1
        mov     SCON,#50h
cek:    SETB    REN
        Mov     R1,#00h
        Call    Cekbaca
        Mov     R1,A
        Mov     A,#00h
        CJNE    R1,#'C',cek
        Clr     REN
        Call    tulis
        Jmp     cek

CekBaca:
        JNB RI,$        ; tunggu SBUF berisi data baru
        MOV A,SBUF ; ambil data
        CLR RI                  ; pertanda data sudah diambil
        RET
;=================================================
; mengirimkan data lewat serial port
;=================================================
tulis:  mov     DPTR,#Pesan ; alamat text pesan
next_1:call     Ldelay
        clr     A
        movc    A,@A+DPTR ; ambil data
        clr     ES
        mov     sbuf,A          ; kirim lewat serial comm.
        jnb     ti,$
        clr     ti
        setb    ES
        Inc     DPTR
        cjne    A,#0,next_1
        Ret
;-- Routine delay --
Ldelay: mov     R2,#50
Ld1:    djnz    R6,$
```

```
{=================================================
  Source code Delphi USB_Donggle.
  File name : USB_Donggle.pas
  Oleh : Muhammad Bahesti Syuaiban – 06.12.918
  INSTITUT TEKNOLOGI NASIONAL MALANG
=================================================}

{==============Begin Of Usetuplisensi.pas=====================}
unit UsetupLisensi;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, ComCtrls, StdCtrls, Buttons, CPort, CPortCtl;
type
  TForm1 = class(TForm)
    PageControl1: TPageControl;
    TabSheet1: TTabSheet;
    TabSheet2: TTabSheet;
    GroupBox1: TGroupBox;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    ComPort1: TComPort;
    GroupBox2: TGroupBox;
    Label11: TLabel;
    Label12: TLabel;
    Label13: TLabel;
    BitBtn7: TBitBtn;
    ProgressBar1: TProgressBar;
    Editbr: TEdit;
    editId: TEdit;
    Edit2: TEdit;
    Edit5: TEdit;
    Edit6: TEdit;
    Label9: TLabel;
    GroupBox3: TGroupBox;
    BitBtn1: TBitBtn;
    Label10: TLabel;
    Edit3: TEdit;
    GroupBox4: TGroupBox;
    Label1: TLabel;
    Memoplainset: TMemo;
    BitBtn4: TBitBtn;
    BitBtn2: TBitBtn;
    GroupBox5: TGroupBox;
```

```
    Memocipherset: TMemo;
    Label2: TLabel;
    BitBtn3: TBitBtn;
    GroupBox6: TGroupBox;
    Label14: TLabel;
    Label15: TLabel;
    Label16: TLabel;
    BitBtn6: TBitBtn;
    Edit7: TEdit;
    Edit8: TEdit;
    Edit9: TEdit;
    ProgressBar2: TProgressBar;
    Editport: TEdit;
    BitBtn5: TBitBtn;
    Label8: TLabel;
    Edit1: TEdit;
    Edit11: TEdit;
    ComLed1: TComLed;
    Memo1: TMemo;
    procedure BitBtn2Click(Sender: TObject);
    procedure BitBtn3Click(Sender: TObject);
    procedure BitBtn4Click(Sender: TObject);
    procedure FormDblClick(Sender: TObject);
    procedure BitBtn7Click(Sender: TObject);
    procedure BitBtn5Click(Sender: TObject);
    procedure BitBtn6Click(Sender: TObject);
    procedure BitBtn1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
  Form1: TForm1;
implementation
uses Unitvar, UProsesView, DES_Function, UDes_Fuction, UFrmMama;
{$R *.dfm}
procedure TForm1.BitBtn2Click(Sender: TObject);
begin
If FileExists(GetCurrentDir+'\resources\PlainReferensi.plx') then
   Begin
encode_file(GetCurrentDir+'\resources\PlainReferensi.plx',GetCurrentDir+'\resources\c
iphertext.cpx',Edit3.Text);
   PDelay(1000,ProgressBar2);
   end
else
```

```
Application.MessageBox('Parameter belum lengkap !!','Error',MB_OK or
MB_ICONHAND);
end;
procedure TForm1.BitBtn3Click(Sender: TObject);
var kotaksimpan:TSaveDialog;
begin
If FileExists(GetCurrentDir+'\resources\ciphertext.cpx') then
 Begin
  Memocipherset.Lines.LoadFromFile(GetCurrentDir+'\resources\ciphertext.cpx');
 end
 else
  Application.MessageBox('File cipher tidak ditemukan !!','Konfirmasi',MB_OK or
MB_ICONHAND);
end;
procedure TForm1.BitBtn4Click(Sender: TObject);
 begin
If FileExists(GetCurrentDir+'\resources\PlainReferensi.plx') then
 Begin
  If Application.MessageBox('Replace file Plaintext ??','Konfirmasi',MB_YESNO or
MB_ICONQUESTION)=mryes then
  Memoplainset.Lines.SaveToFile(GetCurrentDir+'\resources\PlainReferensi.plx');
 end
 else
   Memoplainset.Lines.SaveToFile(GetCurrentDir+'\resources\PlainReferensi.plx');
end;
procedure TForm1.FormDblClick(Sender: TObject);
begin
 TabSheet1.TabVisible:=Not(TabSheet1.Visible);
end;
procedure TForm1.BitBtn7Click(Sender: TObject);
var pesan:string;
    data:TDongle;
    init:boolean;
begin
BitBtn7.Enabled:=false;
 init:=FSetParameterPort(Form1.ComPort1);
 if not init then
  begin
   Editport.Text:='Tidak terdeteksi';
   Editbr.Text:='Tidak terdeteksi';
   editId.Text:='Tidak dikenali';
   Label3.Enabled:=False;
   Label4.Enabled:=False;
   Label5.Enabled:=False;
   Label6.Enabled:=False;
   Label9.Enabled:=False;
```

```
  Edit6.Enabled:=false;
  Edit5.Enabled:=false;
  Memo1.Enabled:=false;
  Edit2.Enabled:=false;
  Edit11.Enabled:=false;
  BitBtn5.Enabled:=false;
  Application.MessageBox('Koneksikan perangkat aktivasi !!','Informasi',Mb_OK or
MB_ICONINFORMATION);
 end
 else
 Begin
  Form1.Editport.Text:='   '+Form1.ComPort1.Port+'   ';
  Form1.Editbr.Text:='   9600 bps   ';
  pesan:=FGetSerial(ComPort1,ProgressBar1);
  If pesan='' then
  ShowMessage('Proses gagal !!!')
  else
  Begin
  data:=FPisahString(pesan);
  editId.Text:=data.ID;
  Edit1.Text:=data.Kunci;
  Edit6.Enabled:=True;
  Edit5.Enabled:=True;
  Memo1.Enabled:=True;
  Edit2.Enabled:=True;
  Edit11.Enabled:=True;
  Label3.Enabled:=True;
  Label4.Enabled:=True;
  Label5.Enabled:=True;
  Label6.Enabled:=True;
  Label9.Enabled:=True;
  BitBtn5.Enabled:=True;
 end;
end;
Global_key:=data.Kunci;
BitBtn7.Enabled:=True;
end;
procedure TForm1.BitBtn5Click(Sender: TObject);
Var status:TStatusAktifasi;
   username,compname:string;
   versiOS:string;
begin
If (FileExists(GetCurrentDir+'\resources\PlainReferensi.plx')) and
(FileExists(GetCurrentDir+'\resources\ciphertext.cpx'))and (Edit1.Text<>'') then
 Begin
```

```
decode_file(GetCurrentDir+'\resources\ciphertext.cpx',GetCurrentDir+'\resources\Plain
Hasil.plx',Edit1.Text);
PDelay(1000,ProgressBar1);
status:=Aktifasi(GetCurrentDir+'\resources\PlainReferensi.plx',GetCurrentDir+'\resour
ces\PlainHasil.plx');
  If status=Aktif then
  Begin
    With Frmmama do
    Begin
     NewWorksheet1.Enabled:=true;
     LoadWorksheet1.Enabled:=true;
     Save1.Enabled:=True;
     GraphViewe1.Enabled:=true;
     TBNew.Enabled:=True;
     TBLoad.Enabled:=True;
     OperationViewer1.Enabled:=true;
     OperationBitViewer1.Enabled:=True;
     GetCurrentUserName(username);
     GetCompName(compname);
     Edit6.Text:=username;
     Edit5.Text:=compname;
     versiOS:=GetOS;
     Memo1.Clear;
     Memo1.Text:=versiOS;
     Edit1.Text:=Global_key;
     Application.MessageBox('Proses aktifasi telah sukses dilakukan
!!','Informasi',MB_OK or MB_ICONEXCLAMATION);
      end;
  end
  else
  Begin
    With Frmmama do
     Begin
     NewWorksheet1.Enabled:=false;
     LoadWorksheet1.Enabled:=false;
     Save1.Enabled:=false;
     GraphViewe1.Enabled:=false;
     OperationViewer1.Enabled:=false;
     OperationBitViewer1.Enabled:=false;
     TBNew.Enabled:=False;
     TBLoad.Enabled:=False;
     end;
     Application.MessageBox('Maaf, Proses aktifasi gagal dilakukan
!!','Informasi',MB_OK or MB_ICONHAND);
     close;
  end;
```

```
 end
else
Application.MessageBox('Parameter belum lengkap !!','Error',MB_OK or
MB_ICONHAND);
end;
procedure TForm1.BitBtn6Click(Sender: TObject);
var pesan:string;
   data:TDongle;
   init:boolean;
begin
BitBtn6.Enabled:=false;
 init:=FSetParameterPort(Form1.ComPort1);
 if not init then
  begin
   Edit7.Text:='Tidak terdeteksi';
   Edit8.Text:='Tidak terdeteksi';
   edit9.Text:='Tidak dikenali';
   Edit3.Enabled:=false;
   BitBtn1.Enabled:=False;
   Memoplainset.Enabled:=False;
   BitBtn4.Enabled:=False;
   BitBtn2.Enabled:=False;
   Memocipherset.Enabled:=False;
   BitBtn3.Enabled:=False;
   Label10.Enabled:=false;
   Label1.Enabled:=false;
   Label2.Enabled:=false;
   Application.MessageBox('Koneksikan perangkat kode lisensi  !!','Informasi',Mb_OK or
MB_ICONINFORMATION);
  end
  else
  Begin
   Edit3.Enabled:=True;
   BitBtn1.Enabled:=True;
   Memoplainset.Enabled:=True;
   BitBtn4.Enabled:=True;
   BitBtn2.Enabled:=True;
   Memocipherset.Enabled:=True;
   BitBtn3.Enabled:=True;
   Label10.Enabled:=True;
   Label1.Enabled:=True;
   Label2.Enabled:=True;
   Edit7.Text:='   '+Form1.ComPort1.Port+'   ';
   Edit8.Text:='   9600 bps   ';
  pesan:=FGetSerial(ComPort1,ProgressBar2);
   If pesan='' then
```

```
  ShowMessage('Proses gagal !!!')
  else
  Begin
  data:=FPisahString(pesan);
  edit9.Text:=data.ID;
 end;
end;
Global_key:=data.Kunci;
BitBtn6.Enabled:=True;
end;
procedure TForm1.BitBtn1Click(Sender: TObject);
begin
Edit3.Text:=Global_key;
end;
end
{==============End Of Setup lisensi.pas ======================}


{==============Begin Of UnitVar.pas=========================}
.unit Unitvar;
interface
uses sysutils,windows,Messages,classes,ComCtrls,CPort,Dialogs;
Type   TDongle=record
       ID:string;
       Kunci:String;
       end;
Type   TDataUser=record
       Komp,NamaUser,OS,Tgl,Jam:string;
       end;
type   Tdata=array [1..8] of char;
       TStatusAktifasi='(Aktif,Non_Aktif);
       Pdata=^Tdata;
       TPBox=array[byte] of byte;
       TJadwal_kunci=array[1..16] of integer;
       Tkunci=packed record
       R,L:Dword;
       end;
Var    Global_key:string;
Function FSetParameterPort(komponen:TComPort):boolean;
Procedure PDelay(lama:integer;Progress:TProgressBar);
Function FPisahString(kata:string):TDongle;
Function FGetSerial(komponen:TComPort;Progress:TProgressBar):string;
procedure encode_file(fileinput,fileoutput,key:string);
procedure decode_file(fileinput,fileoutput,key:string);
function encodebyte(huruf:char;key:string):char;
function decodebyte(huruf:char;key:string):char;
```

```
function encodeblock(data:Tdata;key:string):Tdata;
function decodeblock(data:Tdata;key:string):Tdata;
Function bentuk_kunci(kunci:string):Tkunci;
function keyschedule(kunci:Tkunci):Tkunci;
Function Aktifasi(Ref,Hasil:string):TStatusAktifasi;
function GetCurrentUserName(var UserName: string) : boolean;
function GetCompName(var CompName: string) : boolean;
Function GetOS:string;
function IsComPortAvailable(Port: PChar): Boolean;
var s:string;
implementation
const
 Jadwal_kunci:TJadwal_kunci=(2,2,1,1,1,1,1,1,2,1,1,1,1,1,1,2);
 PBox1:TPBox=(
        0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,
        21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,
        41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,
        61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,
        81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99,100,
        1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,
        21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,
        41,42,43,44,45,46,47,48,49,50,51,52,53,54,55,56,57,58,59,60,
        61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,80,
        81,82,83,84,85,86,87,88,89,90,91,92,93,94,95,96,97,98,99,100
        ,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,
        21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,37,38,39,40,
        41,42,43,44,45,46,47,48,49,50,51,52,53,54,55
        );
function IsComPortAvailable(Port: PChar): Boolean;
var
  DeviceName: array[0..80] of Char;
  ComFile: THandle;
begin
  StrPCopy(DeviceName, Port);

  ComFile := CreateFile(DeviceName,
        GENERIC_READ or GENERIC_WRITE, 0, nil,
        OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, 0);
  Result := ComFile <> INVALID_HANDLE_VALUE;
  CloseHandle(ComFile);
end;
Function GetOS:string;
var WinVersion : TOSVersionInfo;
   SWinVer : string;
   MyNum, MnNum, BldNum : Integer;
begin
```

```
  WinVersion.dwOSVersionInfoSize := SizeOf(TOSVersionInfo);
  GetVersionEx(WinVersion);  // cek versi windows
  case WinVersion.dwMajorVersion of
   3 : SWinVer := 'Windows 3.1';
   4 :
    begin
     case WinVersion.dwPlatformId of
      VER_PLATFORM_WIN32_NT :
        SWinVer := 'Microsoft Windows NT 4.0';
      VER_PLATFORM_WIN32_WINDOWS :
       case WinVersion.dwMinorVersion of
        0  : SWinVer := 'Microsoft Windows 95';
        10 : SWinVer := 'Microsoft Windows 98';
        90 : SWinVer := 'Microsoft Windows Millenium';
       end;
     end; // end case platformID
    end;
   5 :
    begin
     case WinVersion.dwMinorVersion of
      0 : SWinVer := 'Microsoft Windows 2000';
      1 : SWinVer := 'Microsoft Windows Xp';
     end;
    end;
   end;
  MyNum := WinVersion.dwMajorVersion;
  MnNum := WinVersion.dwMinorVersion;
  BldNum := WinVersion.dwBuildNumber;
  Result:=SWinVer+' Major Ver.'+inttostr(MyNum)+' Minor Ver.'+inttostr(MnNum);
end;
function GetCompName(var CompName: string) : boolean;
const
  MaxCompNameLen = 255;
var
  dwCompNameLen : DWORD;
begin
  Result := false;
  dwCompNameLen := MaxCompNameLen - 1;
  SetLength(CompName, MaxCompNameLen);
  if GetComputerName(PChar(CompName), dwCompNameLen) then
   Result := true;
  SetLength(CompName, dwCompNameLen);
end;
function GetCurrentUserName(var UserName: string) : boolean;
const
  MaxUserNameLen = 255;
```

```
var
  dwUserNameLen : DWORD;
begin
  Result := false;
  dwUserNameLen := MaxUserNameLen-1;
  SetLength(UserName, MaxUserNameLen);
  if GetUserName(PChar(UserName), dwUserNameLen) then
    Result := true;
  SetLength(UserName, dwUserNameLen);
end;
Function Aktifasi(Ref,Hasil:string):TStatusAktifasi;
  var
   filemasukan,filekeluaran:Tfilestream;
   penyangga1,penyangga2:Tdata;
   ukuranfile:double;
   N,i:integer;
   iterasi:boolean;
  begin
  N:=0;
  filemasukan:=TFileStream.Create(Ref,fmopenread);
  filemasukan.Seek(0,soFromBeginning);
  filekeluaran:=TFileStream.Create(Hasil,fmopenread);
  filekeluaran.Seek(0,soFromBeginning);
  ukuranfile:=1;
  Result:=Aktif;
  while N<=ukuranfile do
  begin
  filemasukan.Position:=N;
  filemasukan.read(penyangga1,sizeof(penyangga1));
  filekeluaran.Read(penyangga2,sizeof(penyangga2));
  N:=N+8;
  iterasi:=true;
  i:=0;
   While iterasi do
    Begin
      inc(i);
      If i<=8 then Begin
       if penyangga1[i]<>penyangga2[i] then
       Begin
        iterasi:=False;
        Result:=Non_Aktif;
        end;
              end else
        iterasi:=false;
    end;
 end;
```

```
 filekeluaran.Free;
 filemasukan.Free;
end;
Function keyschedule(kunci:Tkunci):Tkunci;
 var
  I:integer;
  R,L:Dword;
 begin
 R:=kunci.R;
 L:=kunci.L;
 for I:=1 to length(Jadwal_kunci)do
 begin
   R:=R shl jadwal_kunci[I];
   L:=L shl jadwal_kunci[I];
 end;
 Result.R:=R;
 Result.L:=L;
end;
Function bentuk_kunci(kunci:string):Tkunci;
 var
  nilaiHexa:array [1..8] of string;
  nilaiordinal:byte;
  N:integer;
 begin
 for N:=1 to 4 do
  begin
   nilaiordinal:= ord(kunci[N]);
   nilaihexa[N]:=IntToHex(nilaiordinal,2);
  end;
  result.R:=StrToInt('$'+nilaihexa[1]+nilaihexa[2]+nilaihexa[3]+nilaihexa[4]);
 for N:=5 to 8 do
 begin
  nilaiordinal:=ord(kunci[N]);
  nilaihexa[N]:=IntToHex(nilaiordinal,2);
 end;
 result.L:=StrToInt('$'+nilaihexa[5]+nilaihexa[6]+nilaihexa[7]+nilaihexa[8]);
end;
Procedure decode_file(fileinput,fileoutput,key:string);
 var
 filemasukan,filekeluaran:TFilestream;
 buffer1,buffer2:Tdata;
 N:integer;
 ukuranfile:double;
begin
N:=0;
filemasukan:=TFileStream.Create(fileinput,FMopenread);
```

```
  filemasukan.seek(0,soFromBeginning);
  filekeluaran:=TFileStream.Create(fileoutput,FMcreate);
  filekeluaran.seek(0,soFromBeginning);
  ukuranfile:=1;
  while N<=ukuranfile do
  begin
   filemasukan.Position:=N;
   filemasukan.Read(buffer1,sizeof(buffer1));
   buffer2:=decodeblock(buffer1,key);
   filekeluaran.Write(buffer2,sizeof(buffer2));
   N:=N+8;
  end;
  filemasukan.Free;
  filekeluaran.Free;
end;
Procedure encode_file(fileinput,fileoutput,key:string);
  var
   filemasukan,filekeluaran:Tfilestream;
   penyangga1,penyangga2:Tdata;
   ukuranfile:double;
   N:integer;
  begin
  N:=0;
  filemasukan:=TFileStream.Create(fileinput,fmopenread);
  filemasukan.Seek(0,soFromBeginning);
  filekeluaran:=TFileStream.Create(fileoutput,fmcreate);
  filekeluaran.Seek(0,soFromBeginning);
  ukuranfile:=1;
  while N<=ukuranfile do
  begin
  filemasukan.Position:=N;
  filemasukan.read(penyangga1,sizeof(penyangga1));
  penyangga2:=encodeblock(penyangga1,key);
  filekeluaran.Write(penyangga2,sizeof(penyangga2));
  N:=N+8;
 end;
 filekeluaran.Free;
 filemasukan.Free;
 end;
Function decodeblock(data:Tdata;key:string):Tdata;
 var
  panjangdata,N:integer;
  pointerdata:Pdata;
  Pkey:PString;
 begin
 panjangdata:=length(data);
```

```
  Pkey:=@key;
  pointerdata:=@data;
  for N:=1 to panjangdata do
   result[N]:=decodebyte(pointerdata^[N],Pkey^);
  end;
Function encodeblock(data:Tdata;key:string):Tdata;
  var
   panjangdata,N:integer;
   pointerdata:Pdata;
   Pkey:PString;
  begin
  panjangdata:=length(data);
  Pkey:=@key;
  pointerdata:=@data;
  for N:=1 to panjangdata do
    result[N]:=encodebyte(pointerdata^[N],Pkey^);
  end;

  function decodebyte(huruf:char;key:string):char;
  var
   panjangkunci,N:Integer;
   ordinalkey,ordinal:byte;
   Pordinal,Pordinalkey:Pbyte;
  begin
  ordinal:=ord(huruf)-3;
  Pordinal:=@ordinal;
  panjangkunci:=length(key);
  for N:=1 to panjangkunci do
  begin
   ordinalkey:=ord(key[N]);
   Pordinal^:=Pordinal^ xor PBox1(.ordinalkey.);
   Pordinal^:=Pordinal^ xor ordinalkey;
  end;
  result:=chr(Pordinal^);
end;
Function encodebyte(huruf:char;key:string):char;
 var
  panjangkunci,N:Integer;
  ordinalkey,ordinal:byte;
  Pordinal:Pbyte;
 begin
 ordinal:=ord(huruf);
 Pordinal:=@ordinal;
 panjangkunci:=length(key);
 for N:=1 to panjangkunci do
 begin
```

```
  ordinalkey:=ord(key[N]);
  Pordinal^:=Pordinal^ xor ordinalkey;
  Pordinal^:=Pordinal^ xor PBox1(.ordinalkey.);
 end;
 result:=chr(Pordinal^+3);
 end;
Function FGetSerial(komponen:TComPort;Progress:TProgressBar):string;
var kata:string;
Begin
kata:='';
try
komponen.WriteStr('C');
if Progress<>nil then
PDelay(1000,Progress);
komponen.ReadStr(kata,17);
If Length(kata)=17 then
Begin
Result:=kata;
end;
except
ShowMessage('Kesalahan pembacaan !!!');
end;
end;
Function FPisahString(kata:string):TDongle;
var i:integer;
   p:integer;
   kata1,kata2:string;
   loop:Boolean;
Begin
P:=Length(kata);
kata1:='';
Kata2:='';
i:=0;
Loop:=True;
While loop do
 Begin
  inc(i);
  If kata[i]<>'#' then
  kata1:=kata1+kata[i]
  else
  loop:=false;
end;
Loop:=True;
While loop do
Begin
 inc(i);
```

```
  kata2:=kata2+kata[i];
  if i= p then  Loop:=False;
 end;
 Result.ID:=kata1;
 Result.Kunci:=kata2;
end;
Procedure PDelay(lama:integer;Progress:TProgressBar);
Var i,j,k,x:integer;
Begin
Progress.Min:=0;
Progress.Max:=lama;
 For i:=0 to lama do
 Begin
 x:=0;
 If Progress<> nil then
 Progress.Position:=i;
 For j:=0 to lama do
  For k:=0 to lama do
   inc(x);
 end;
end;
Function FSetParameterPort(komponen:TComPort):boolean;
var ketemu:boolean;
   i,eksepsi,mulai,j:integer;
   x:string;
   Port:TStringList;
Begin
 komponen.Close;
 komponen.BaudRate:=br9600;
 komponen.StopBits:=sbOneStopBit;
 komponen.Parity.Bits:=prNone;
 komponen.DataBits:=dbEight;
 Port:=TStringList.Create;
 EnumComPorts(Port);
 eksepsi:=0;
 i:=0;
 j:=0;
 Result:=false;
 ketemu:=false;
 For i:=0 to Port.Count-1 do
  begin
  komponen.Port:=Port[i];
  try
   komponen.Connected:=true;
  except
   komponen.Connected:=false;
```

```
  end;
  if komponen.Connected then
  begin
   j:=i;
   komponen.Close;
  end;
 end;
If j<>0 then
 Begin
 komponen.Port:=Port[j];
 try
 komponen.Open;
 except
 komponen.Close;
 end;
 Result:=true;
 end
 else
 Result:=false;
end;
end.
{==================End Of UnitVar.pas=====================}

{=====================End Program=====================}
```

FT232BM is the 2nd generation of FTDI's popular USB UART i.c. This device not only adds extra functionality s FT8U232AM predecessor and reduces external component count, but also maintains a high degree of pin patibility with the original, making it easy to upgrade or cost reduce existing designs as well as increasing the ntial for using the device in new application areas.

Features

RDWARE FEATURES

- Single Chip USB ⇔Asynchronous Serial Data Transfer
- Full Handshaking & Modem Interface Signals
- UART I/F Supports 7 / 8 Bit Data, 1 / 2 Stop Bits and Odd/Even/Mark/Space/No Parity
- Data rate 300 => 3M Baud (TTL)
- Data rate 300 => 1M Baud (RS232)
- Data rate 300 => 3M Baud (RS422/RS485)
- 384 Byte Receive Buffer / 128 Byte Transmit Buffer for high data throughput
- Adjustable RX buffer timeout
- Full hardware assisted hardware or X-On / X-Off handshaking
- In-built support for event characters and line break condition
- Auto Transmit Buffer control for RS485
- Support for USB Suspend / Resume through SLEEP# and RI# pins
- Support for high power USB Bus powered devices through PWREN# pin
- Integrated level converter on UART and control signals for interfacing to 5V and 3.3V logic
- Integrated 3.3V regulator for USB IO
- Integrated Power-On-Reset circuit
- Integrated 6MHz – 48Mhz clock multiplier PLL
- USB Bulk or Isochronous data transfer modes
- 4.35V to 5.25V single supply operation
- UHCI / OHCI / EHCI host controller compatible
- USB 1.1 and USB 2.0 compatible
- USB VID, PID, Serial Number and Product Description strings in external EEPROM
- EEPROM programmable on-board via USB
- Compact 32-LD LQFP package

VIRTUAL COM PORT (VCP) DRIVERS for

- Windows 98 and Windows 98 SE
- Windows 2000 / ME / XP
- Windows CE **
- MAC OS-8 and OS-9
- MAC OS-X
- Linux 2.40 and greater

D2XX (USB Direct Drivers + DLL S/W Interface)

- Windows 98 and Windows 98 SE
- Windows 2000 / ME / XP

APPLICATION AREAS

- USB ⇔RS232 Converters
- USB ⇔ RS422 / RS485 Converters
- Upgrading RS232 Legacy Peripherals to USB
- Cellular and Cordless Phone USB data transfer cables and interfaces
- Interfacing MCU based designs to USB
- USB Audio and Low Bandwidth Video data transfer
- PDA ⇔ USB data transfer
- USB Smart Card Readers
- Set Top Box (S.T.B.) PC - USB interface
- USB Hardware Modems
- USB Wireless Modems
- USB Instrumentation
- USB Bar Code Readers

[** = In planning or under development]

Enhancements

s section summarises the enhancements of the 2nd generation device compared to it's FT8U232AM predecessor. further details, consult the device pin-out description and functional descriptions.

Integrated Power-On-Reset (POR) Circuit
The device now incorporates an internal POR function. The existing RESET# pin is maintained in order to allow external logic to reset the device where required, however for many applications this pin can now simply be hard wired to VCC. In addition, a new reset output pin (RSTOUT#) is provided in order to allow the new POR circuit to provide a stable reset to external MCU and other devices. RSTOUT# was the TEST pin on the previous generation of devices.

Integrated RCCLK Circuit
In the previous devices, an external RC circuit was required to ensure that the oscillator and clock multiplier PLL frequency was stable prior to enabling the clock internal to the device. This circuit is now embedded on-chip – the pin assigned to this function is now designated as the TEST pin and should be tied to GND for normal operation.

Integrated Level Converter on UART interface and control signals
The previous devices would drive the UART and control signals at 5V CMOS logic levels. The new device has a separate VCC-IO pin allowing the device to directly interface to 3.3V and other logic families without the need for external level converter i.c.'s

Improved Power Management control for USB Bus Powered, high current devices
The previous devices had a USBEN pin, which became active when the device was enumerated by USB. To provide power control, this signal had to be externally gated with SLEEP# and RESET#. This gating is now done on-chip - USBEN has now been replaced with the new PWREN# signal which can be used to directly drive a transistor or P-Channel MOSFET in applications where power switching of external circuitry is required. A new EEPROM based option makes the device pull gently down it's UART interface lines when the power is shut off (PWREN# is High). In this mode, any residual voltage on external circuitry is bled to GND when power is removed thus ensuring that external circuitry controlled by PWREN# resets reliably when power is restored.

- **Lower Suspend Current**
  Integration of RCCLK within the device and internal design improvements reduce the suspend current of the FT232BM to under 200uA (excluding the 1.5k pull-up on USB DP) in USB suspend mode. This allows greater margin for peripherals to meet the USB Suspend current limit of 500uA.

- **Support for USB Isochronous Transfers**
  Whilst USB Bulk transfer is usually the best choice for data transfer, the scheduling time of the data is not guaranteed. For applications where scheduling latency takes priority over data integrity such as transferring audio and low bandwidth video data, the new device now offers an option of USB Isochronous transfer via an option bit in the EEPROM.

- **Programmable Receive Buffer Timeout**
  In the previous device, the receive buffer timeout used to flush remaining data from the receive buffer was fixed at 16ms timeout. This timeout is now programmable over USB in 1ms increments

from 1ms to 255ms, thus allowing the device to be better optimised for protocols requiring faster response times from short data packets.

TXDEN Timing fix

TXDEN timing has now been fixed to remove the external delay that was previously required for RS485 applications at high baud rates. TXDEN now works correctly during a transmit send-break condition.

Relaxed VCC Decoupling

The 2nd generation devices now incorporate a level of on-chip VCC decoupling. Though this does not eliminate the need for external decoupling capacitors, it significantly improves the ease of PCB design requirements to meet FCC, CE and other EMI related specifications.

Improved PreScaler Granularity

The previous version of the Prescaler supported division by (n + 0), (n + 0.125), (n + 0.25) and (n + 0.5) where n is an integer between 2 and 16,384 (2^{14}). To this we have added (n + 0.375), (n + 0.625), (n + 0.75) and (n+ 0.875) which can be used to improve the accuracy of some baud rates and generate new baud rates which were previously impossible (especially with higher baud rates).

Bit Bang Mode

The 2nd generation device has a new option referred to as "Bit Bang" mode. In Bit Bang mode, the eight UART interface control lines can be switched between UART interface mode and an 8-bit Parallel IO port. Data packets can be sent to the device and they will be sequentially sent to the interface at a rate controlled by the prescaler setting. As well as allowing the device to be used stand-alone as a general purpose IO controller for example controlling lights, relays and switches, some other interesting possibilities exist. For instance, it may be possible to connect the device to an SRAM configurable FPGA as supplied by vendors such as Altera and Xilinx. The FPGA device would normally be un-configured (i.e. have no defined function) at power-up. Application software on the PC could use Bit Bang Mode to download configuration data to the FPGA which would define it's hardware function, then after the FPGA device is configured the FT232BM can switch back into UART interface mode to allow the programmed FPGA device to communicate with the PC over USB. This approach allows a customer to create a "generic" USB peripheral who's hardware function can be defined under control of the application software. The FPGA based hardware can be easily upgraded or totally changed simply by changing the FPGA configuration data file. Application notes, software and development modules for this application area will be available from FTDI and other 3rd parties.

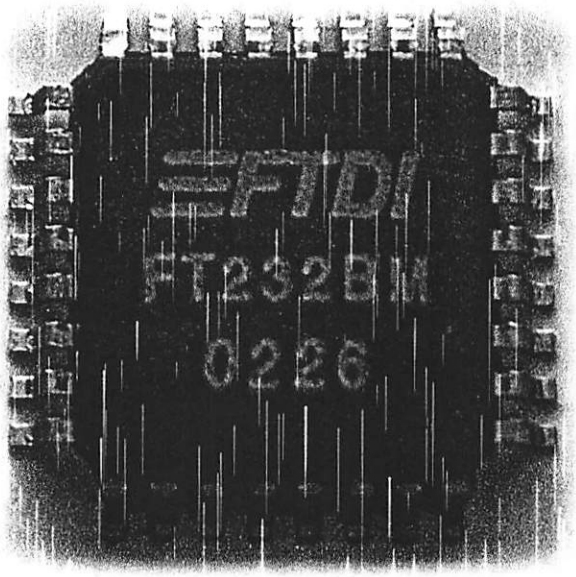
- **PreScaler Divide By 1 Fix**

  The previous device had a problem when the integer part of the divisor was set to 1. In the 2nd generation device setting the prescaler value to 1 gives a baud rate of 2 million baud and setting it to zero gives a baud rate of 3 million baud. Non-integer division is not supported with divisor values of 0 and 1.

- **Less External Support Components**

  As well as eliminating the RCCLK RC network, and for most applications the need for an external reset circuit, we have also eliminated the requirement for a 100k pull-up on EECS to select 6MHz operation. When the FT232BM is being used without the configuration EEPROM, EECS, EESK and EEDATA can now be left n/c. For circuits requiring a long reset time (where the device is reset externally using a reset generator i.c., or

reset is controlled by the IO port of a MCU, FPGA or ASIC device) an external transistor circuit is no longer required as the 1.5k pull-up resistor on USB DP can be wired to the RSTOUT# pin instead of to 3.3V. Note : RSTOUT# drives out at 3.3V level, not at 5V VCC level. This is the preferred configuration for new designs.



Extended EEPROM Support

The previous generation of devices only supported EEPROM of type 93C46 (164 x 16 bit). The new devices will also work with EEPROM type 93C56 (128 x 16 bit) and 93C66 (256 x 16 bit). The extra space is not used by the device, however it is available for use by other external MCU / logic whilst the FT232BM is being held in reset.

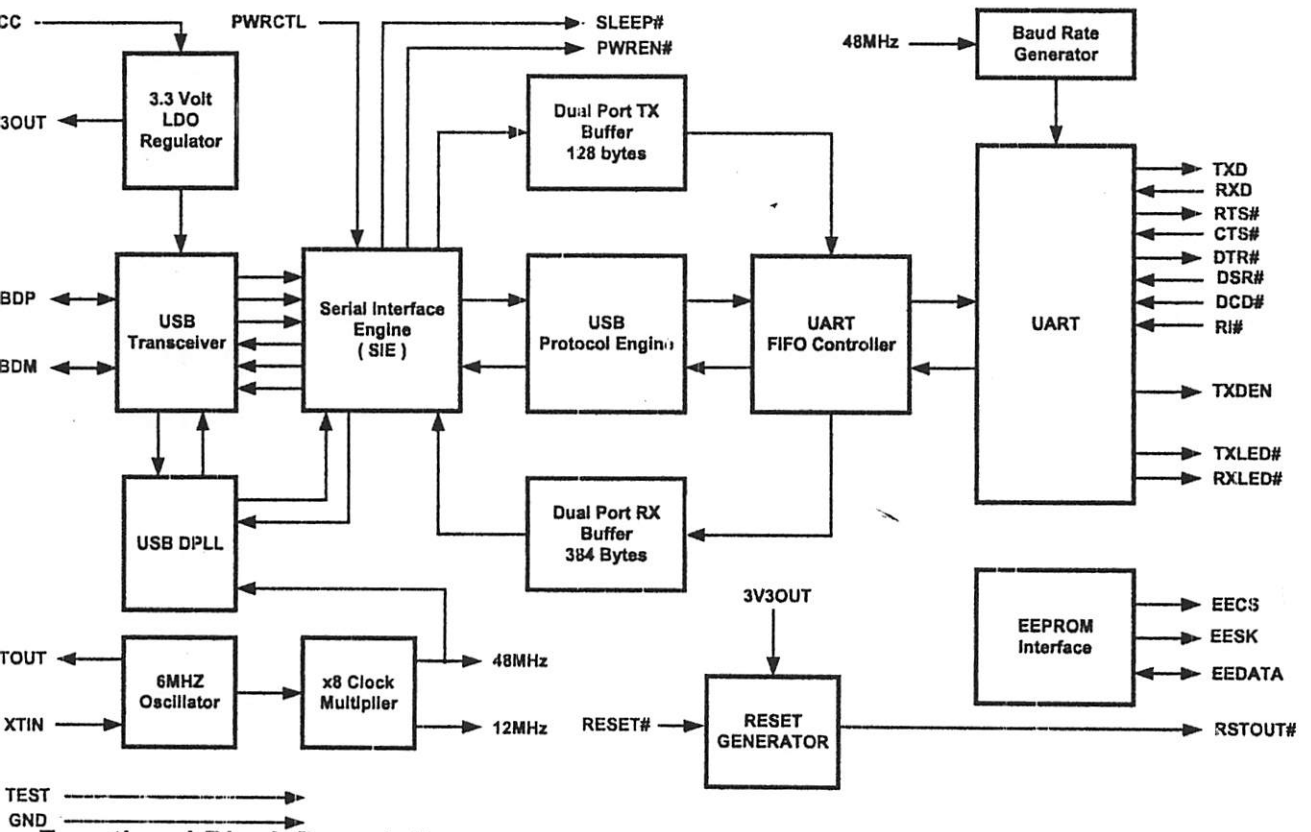
USB 2.0 (full speed option)

A new EEPROM based option allows the FT232BM to return a USB 2.0 device descriptor as opposed to USB 1.1. Note : The device would be a USB 2.0 Full Speed device (12Mb/s) as opposed to a USB 2.0 High Speed device (480Mb/s).

Multiple Device Support without EEPROM

When no EEPROM (or a blank or invalid EEPROM) is attached to the device, the FT232BM no longer gives a serial number as part of it's USB descriptor. This allows multiple devices to be simultaneously connected to the same PC. However, we still highly recommend that EEPROM is used, as without serial numbers a device can only be identified by which hub port in the USB tree it is connected to which can change if the end user re-plugs the device into a different port.

Block Diagram (simplified)



Functional Block Descriptions

3.3V LDO Regulator

The 3.3V LDO Regulator generates the 3.3 volt reference voltage for driving the USB transceiver cell output buffers. It requires an external decoupling capacitor to be attached to the 3V3OUT regulator output pin. It also provides 3.3V power to the RSTOUT# pin. The main function of this block is to power the USB Transceiver and the Reset Generator Cells rather than to power external logic. However, external circuitry requiring 3.3V nominal at a current of not greater than 5mA could also draw it's power from the 3V3OUT pin if required.

USB Transceiver

The USB Transceiver Cell provides the USB 1.1 / USB 2.0 full-speed physical interface to the USB cable. The output drivers provide 3.3 volt level slew rate control signalling, whilst a differential receiver and two single ended receivers provide USB data in, SEO and USB Reset condition detection.

- **USB DPLL**

  The USB DPLL cell locks on to the incoming NRZI USB data and provides separate recovered clock and data signals to the SIE block.

- **6MHz Oscillator**

  The 6MHz Oscillator cell generates a 6MHz reference clock input to the x8 Clock multiplier from an external 6MHz crystal or ceramic resonator.

- **x8 Clock Multiplier**

  The x8 Clock Multiplier takes the 6MHz input from the Oscillator cell and generates a 12MHz reference clock for the SIE, USB Protocol Engine and UART FIFO controller blocks. It also generates a 48MHz reference clock for the USB DPPL and the Baud Rate Generator blocks.

- **Serial Interface Engine (SIE)**

  The Serial Interface Engine (SIE) block performs the Parallel to Serial and Serial to Parallel conversion of the USB data. In accordance to the USB 2.0 specification, it performs bit stuffing / un-

stuffing and CRC5 / CRC16 generation / checking on the USB data stream.

USB Protocol Engine

The USB Protocol Engine manages the data stream from the device USB control endpoint. It handles the low level USB protocol (Chapter 9) requests generated by the USB host controller and the commands for controlling the functional parameters of the UART.

Dual Port TX Buffer (128 bytes)

Data from the USB data out endpoint is stored in the Dual Port TX buffer and removed from the buffer to the UART transmit register under control of the UART FIFO controller.

Dual Port RX Buffer (384 bytes)

Data from the UART receive register is stored in the Dual Port RX buffer prior to being removed by the SIE on a USB request for data from the device data in endpoint.

UART FIFO Controller

The UART FIFO controller handles the transfer of data between the Dual Port RX and TX buffers and the UART transmit and receive registers.

UART

The UART performs asynchronous 7 / 8 bit Parallel to Serial and Serial to Parallel conversion of the data on the RS232 (RS422 and RS485) interface. Control signals supported by the UART include RTS, CTS, DSR , DTR, DCD and RI.
The UART provides a transmitter enable control signal (TXDEN) to assist with interfacing to RS485 transceivers. The UART supports RTS/ CTS, DSR/DTR and X-On/X-Off handshaking options. Handshaking, where required, is handled in hardware to ensure fast response times. The UART also supports the RS232 BREAK setting and detection conditions.

Baud Rate Generator

The Baud Rate Generator provides a x16 clock input to the UART from the 48MHz reference clock and consists of a 14 bit prescaler and 3 register bits which provide fine tuning of the baud rate (used to divide by a number plus a fraction). This determines the Baud Rate of the UART which is programmable from 183 baud to 3 million baud.

- **RESET Generator**

  The Reset Generator Cell provides a reliable power-on reset to the device internal circuitry on power up. An additional RESET# input and RSTOUT# output are provided to allow other devices to reset the FT232BM or the FT232BM to reset other devices respectively. During reset, RSTOUT# is driven low, otherwise it drives out at the 3.3V provided by the onboard regulator. RSTOUT# can be used to control the 1.5k pull-up on USB DP directly where delayed USB enumeration is required. It can also be used to reset other devices. RSTOUT# will stay high-impedance for approximately 5ms after VCC has risen above 3.5V AND the device oscillator is running AND RESET# is high. RESET# should be tied to VCC unless it is a requirement to reset the device from external logic or an external reset generator i.c.

- **EEPROM Interface**

  Though the FT232BM will work without the optional EEPROM, an external 93C46 (93C56 or 93C66) EEPROM can be used to customise the USB VID, PID, Serial Number, Product Description Strings and Power Descriptor value of the FT232BM for OEM applications. Other parameters controlled by the EEPROM include Remote Wake Up, Isochronous Transfer Mode, Soft Pull Down on Power-Off and USB 2.0 descriptor modes.
  The EEPROM should be a 16 bit wide configuration such as a MicroChip 93LC46B or equivalent capable of a 1Mb/s clock rate at VCC = 4.35V to 5.25V. The EEPROM is programmable-on board over USB using a utility available from FTDI's web site (http://www.ftdichip.com). This allows a blank part to be soldered onto the PCB and programmed as part of the manufacturing and test process.
  If no EEPROM is connected (or the EEPROM is blank), the FT232BM will use it's built-in default VID, PID Product Description and Power Descriptor Value. In this case, the device will not have a serial number as part of the USB descriptor.

Device Pin-Out

Figure 1
Pin-Out
(LQFP-32 Package)

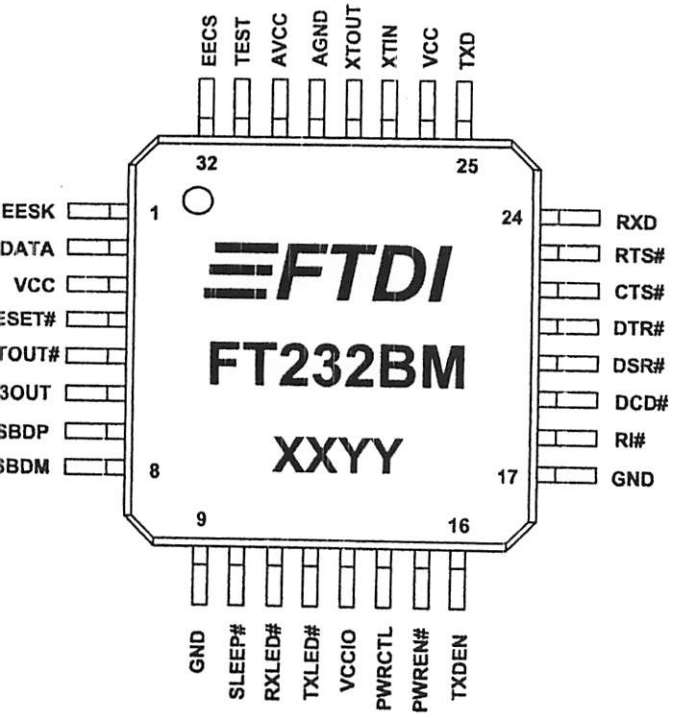
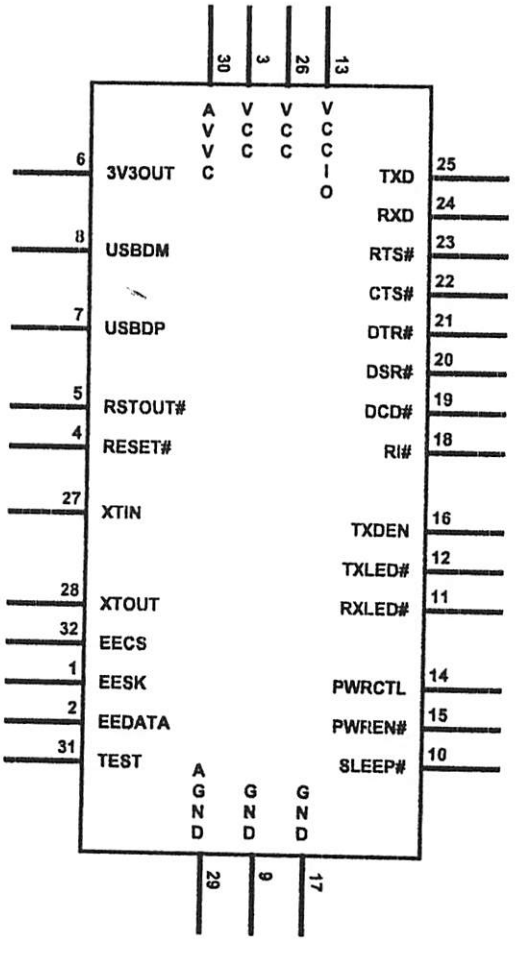


Figure 2
Pin-Out
(Schematic Symbol)



Signal Descriptions

ble 1 - FT232BM - PINOUT DESCRIPTION

RT INTERFACE GROUP

n#	Signal	Type	Description
	TXD	OUT	Transmit Asynchronous Data Output
	RXD	IN	Receive Asynchronous Data Input
	RTS#	OUT	Request To Send Control Output / Handshake signal
	CTS#	IN	Clear To Send Control Input / Handshake signal
	DTR#	OUT	Data Terminal Ready Control Output / Handshake signal
	DSR#	IN	Data Set Ready Control Input / Handshake signal
	DCD#	IN	Data Carrier Detect Control Input
	RI#	IN	Ring Indicator Control Input. When the Remote Wakeup option is enabled in the EEPROM, taking RI# low can be used to resume the PC USB Host controller from suspend.
	TXDEN	OUT	Enable Transmit Data for RS485

B INTERFACE GROUP

n#	Signal	Type	Description
	USBDP	I/O	USB Data Signal Plus (Requires 1.5k pull-up to 3V3OUT or RSTOUT#)
	USBDM	I/O	USB Data Signal Minus

PROM INTERFACE GROUP

n#	Signal	Type	Description
	EECS	I/O	EEPROM – Chip Select. For 48MHz operation pull EECS to GND using a 10k resistor. For 6MHz operation no resistor is required. Tri-State during device reset. **Note 1
	EESK	OUT	Clock signal to EEPROM. Tri-State during device reset, else drives out. **Note 1
	EEDATA	I/O	EEPROM – Data I/O Connect directly to Data-In of the EEPROM and to Data-Out of the EEPROM via a 2.2k resistor. Also, pull Data-Out of the EEPROM to VCC via a 10k resistor for correct operation. Tri-State during device reset. **Note 1

WER CONTROL GROUP

#	Signal	Type	Description
	SLEEP#	OUT	Goes Low during USB Suspend Mode. Typically used to power-down an external TTL to RS232 level converter i.c. in USB -> RS232 converter designs.
	PWREN#	OUT	Goes Low after the device is configured via USB, then high during USB suspend. Can be used to control power to external logic using a P-Channel Logic Level MOSFET switch. Enable the Interface Pull-Down Option in EEPROM when using the PWREN# pin in this way.
	PWRCTL	IN	Bus Powered – Tie Low / Self Powered – Tie High (to VCCIO)

SCELLANEOUS SIGNAL GROUP

in#	Signal	Type	Description
	RESET#	IN	Can be used by an external device to reset the FT232BM. If not required, tie to VCC.
	RSTOUT#	OUT	Output of the internal Reset Generator. Stays high impedance for ~ 5ms after VCC > 3.5V and the internal clock starts up, then clamps it's output to the 3.3v output of the internal regulator. Taking RESET# low will also force RSTOUT# to drive low. RSTOUT# is NOT affected by a USB Bus Reset.
2	TXLED#	O.C.	LED Drive - Pulses Low when Transmitting Data via USB
1	RXLED#	O.C.	LED Drive - Pulses Low when Receiving Data via USB
7	XTIN	IN	Input to 6MHz Crystal Oscillator Cell. This pin can also be driven by an external 6MHz clock if required. Note : Switching threshold of this pin is VCC/2, so if driving from an external source, the source must be driving at 5V CMOS level or a.c. coupled to centre around VCC/2.
8	XTOUT	OUT	Output from 6MHz Crystal Oscillator Cell. XTOUT stops oscillating during USB suspend, so take care if using this signal to clock external logic.
1	TEST	IN	Puts device in i.c. test mode – must be tied to GND for normal operation.

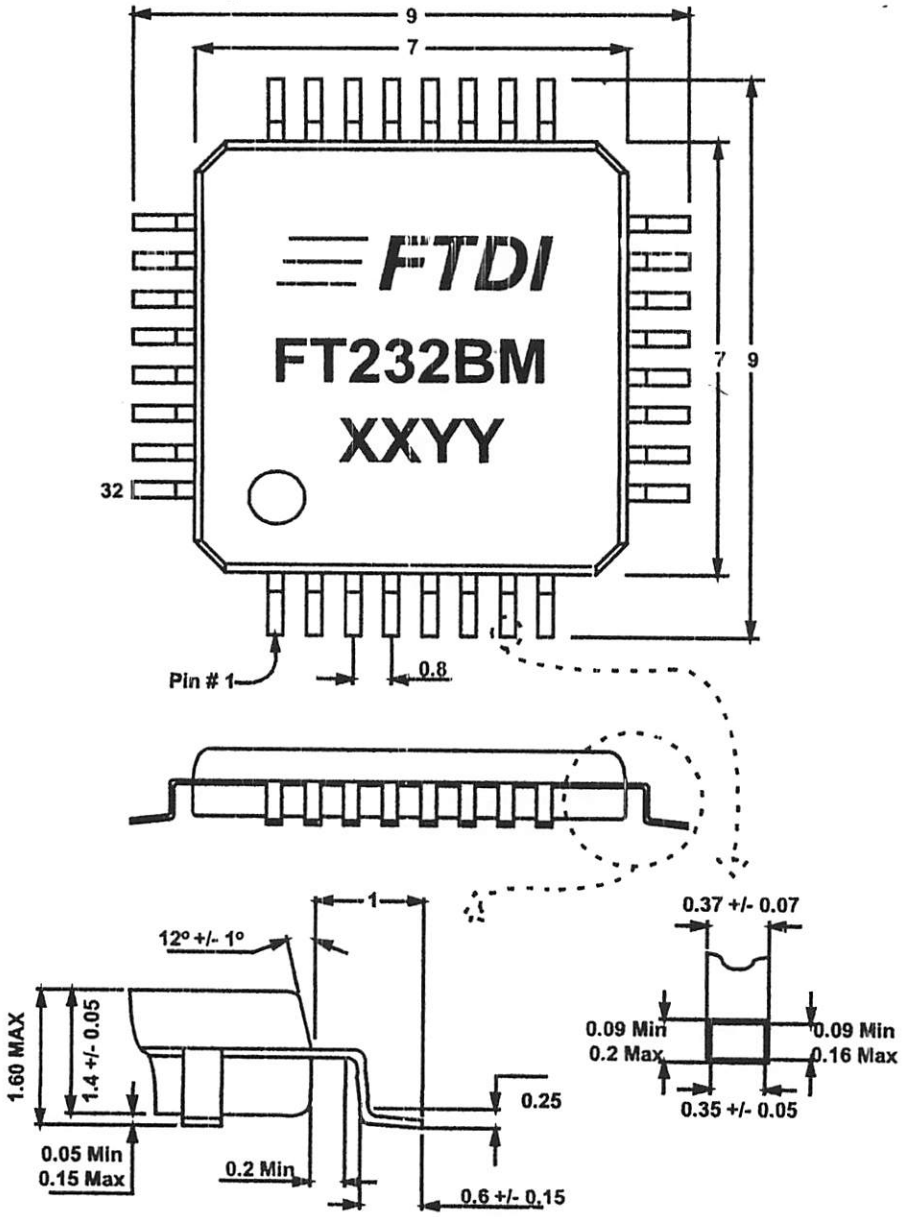
OWER AND GND GROUP

in#	Signal	Type	Description
	3V3OUT	OUT	3.3 volt Output from the integrated L.D.O. regulator This pin should be decoupled to GND using a 33nF ceramic capacitor in close proximity to the device pin. It's prime purpose is to provide the internal 3.3V supply to the USB transceiver cell and the RSTOUT# pin. A small amount of current (<= 5mA) can be drawn from this pin to power external 3.3v logic if required.
,26	VCC	PWR	+4.35 volt to +5.25 volt VCC to the device core, LDO and non-UART interface pins.
3	VCCIO	PWR	+3.0 volt to +5.25 volt VCC to the UART interface pins 10..12, 14..16 and 18..25. When interfacing with 3.3V external logic connect VCCIO to the 3.3V supply of the external logic, otherwise connect to VCC to drive out at 5V CMOS level.
,17	GND	PWR	Device– Ground Supply Pins
0	AVCC	PWR	Device - Analog Power Supply for the internal x8 clock multiplier
9	AGND	PWR	Device - Analog Ground Supply for the internal x8 clock multiplier

Note 1 - During device reset, these pins are tri-state but pulled up to VCC via internal 200k resistors.

) Package Outline

gure 3 – 32 LD LQFP Package Dimensions



The FT232BM is supplied in a 32 LD LQFP package as standard. This package has a 7mm x 7mm body (9mm x 9mm including leads) with leads on a 0.8mm pitch. A lead free version is available on special request. alternative 5mm x 5mm leadless chip scale package (C.S.P.) is also available on special request for projects where package area is critical.

Contact support@ftdichip.com for package details and availability.

The above drawing shows the LQFP-32 package – all dimensions are in millimetres.

XXYY = Date Code (XX = 1 or 2 digit year number, YY = 2 digit week number.

0 Absolute Maximum Ratings

ese are the absolute maximum ratings for the FT232BM device in accordance with the Absolute Maximum Rating System C 60134). Exceeding these may cause permanent damage to the device.

Storage Temperature	–65°C to + 150°C
Ambient Temperature (Power Applied)	0°C to + 70°C
VCC Supply Voltage	-0.5V to +6.00V
DC Input Voltage - Inputs	-0.5V to VCC + 0.5V
DC Input Voltage - High Impedance Bidirectionals	-0.5V to VCC + 0.5V
DC Output Current – Outputs	24mA
DC Output Current – Low Impedance Bidirectionals	24mA
Power Dissipation (VCC = 5.25V)	500mW
Electrostatic Discharge Voltage (I < 1uA)	+/- 2000V
Latch Up Current (Vi < 0 or Vi > Vcc)	100mA

1 D.C. Characteristics

Characteristics (Ambient Temperature = 0 .. 70°C)

erating Voltage and Current

arameter	Description	Min	Typ	Max	Units	Conditions
cc1	VCC Operating Supply Voltage	4.35	5.0	5.25	V	
cc2	VCCIO Operating Supply Voltage	3.0	-	5.25	V	
c1	Operating Supply Current	-	25	-	mA	Normal Operation
c2	Operating Supply Current	-	180	200	uA	USB Suspend **Note 1

ote 1 – *Supply current excludes the 200uA nominal drawn by the external pull-up resistor on USB DP.*

RT IO Pin Characteristics (VCCIO = 5.0V)

rameter	Description	Min	Typ	Max	Units	Conditions
oh	Output Voltage High	3.2	4.1	4.9	V	I source = 2mA
ol	Output Voltage Low	0.3	0.4	0.6	V	I sink = 2mA
n	Input Switching Threshold	1.3	1.6	1.9	V	**Note 2
lys	Input Switching Hysteresis	50	55	60	mV	

RT IO Pin Characteristics (VCCIO = 3.0 - 3.6V)

rameter	Description	Min	Typ	Max	Units	Conditions
oh	Output Voltage High	2.2	2.7	3.2	V	I source = 1mA
ol	Output Voltage Low	0.3	0.4	0.5	V	I sink = 2 mA
n	Input Switching Threshold	1.0	1.2	1.5	V	**Note 2
lys	Input Switching Hysteresis	20	25	30	mV	

ote 2 – *Inputs have an internal 200k pull-up resistor to VCCIO.*

N / XTOUT Pin Characteristics

rameter	Description	Min	Typ	Max	Units	Conditions
oh	Output Voltage High	4.0	-	5.0	V	Fosc = 6MHz
ol	Output Voltage Low	0.1	-	1.0	V	Fosc = 6MHz
n	Input Switching Threshold	1.8	2.5	3.2	V	

SET#, TEST, EECS, EESK, EEDATA Pin Characteristics

rameter	Description	Min	Typ	Max	Units	Conditions
oh	Output Voltage High	3.2	4.1	4.9	V	I source = 2mA
ol	Output Voltage Low	0.3	0.4	0.6	V	I sink = 2 mA
n	Input Switching Threshold	1.3	1.6	1.9	V	**Note 3
Hys	Input Switching Hysteresis	50	55	60	mV	

ote 3 – *EECS, EESK and EEDATA pins have an internal 200k pull-up resistor to VCC*

TOUT Pin Characteristics

rameter	Description	Min	Typ	Max	Units	Conditions
oh	Output Voltage High	3.0	-	3 6	V	I source = 2mA
ol	Output Voltage Low	0.3	-	0.6	V	I sink = 2mA

B IO Pin Characteristics

rameter	Description	Min	Typ	Max	Units	Conditions
Voh	IO Pins Static Output (High)	2.8		3.6	V	RI = 1.5k to 3V3Out (D+) RI = 15k to GND (D-)
Vol	IO Pins Static Output (Low)	0		0.3	V	RI = 1.5k to 3V3Out (D+) RI = 15k to GND (D-)
Vse	Single Ended Rx Threshold	0.8		2.0	V	
Com	Differential Common Mode	0.8		2.5	V	
VDif	Differential Input Sensitivity	0.2			V	
DrvZ	Driver Output Impedance	29		44	Ohm	**Note 4

ote 4 – *Driver Output Impedance includes the external 27R series resistors on USBDP and USBDM pins.*

Device Configuration Examples

Oscillator Configurations

ure 4
Pin Ceramic Resonator
nfiguration

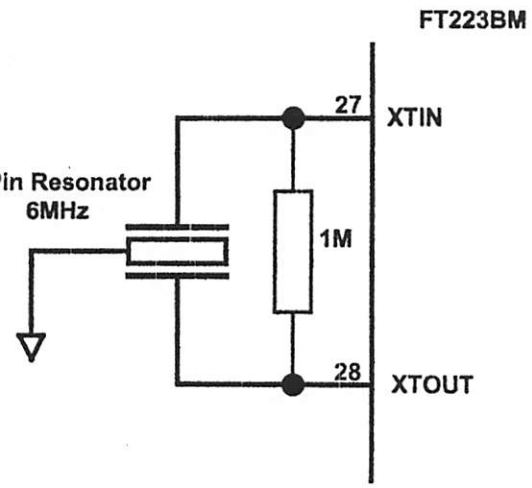
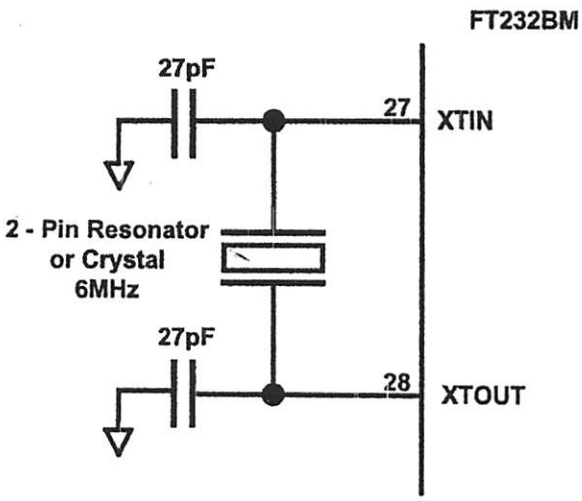


Figure 5
Crystal or 2-Pin Ceramic Resonator Configuration



ure 4 illustrates how to use the FT232BM with a 3-Pin Ceramic Resonator. A suitable part would be a ceramic onator from Murata's CERALOCK range. (Murata Part Number CSTCR6M00G15), or equivalent. 3-Pin ceramic onators have the load capacitors built into the resonator so no external loading capacitors are required. This kes for an economical configuration. The accuracy of this Murata ceramic resonator is +/- 0.1% and it is specifically igned for USB full speed applications. A 1 MOhm loading resistor across XTIN and XTOUT is recommended in er to guarantee this level of accuracy.

er ceramic resonators with a lesser degree of accuracy (typically +/- 5%) are technically out-with the USB cification, but it has been calculated that using such a device will work satisfactorily in practice with a FT232BM gn.

ure 5 illustrates how to use the FT232BM with a 6MHz Crystal or 2-Pin Ceramic Resonator. In this case, these ces do not have in-built loading capacitors so these have to be added between XTIN, XTOUT and GND as wn. A value of 27pF is shown as the capacitor in the example – this will be good for many crystals and some onators but do select the value based on the manufacturers recommendations wherever possible. If using a crystal, a parallel cut type. If using a resonator, see the previous note on frequency accuracy.

EEPROM Configuration

ure 6
Prom Configuration

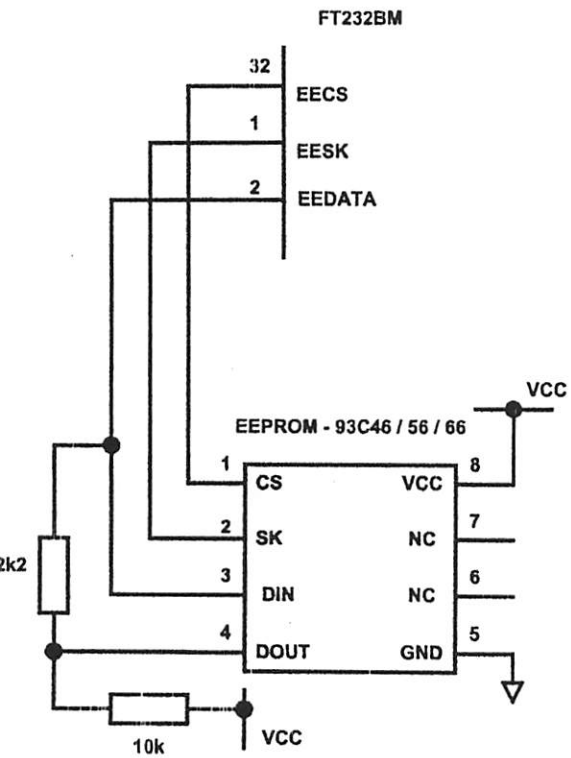


Figure 6 illustrates how to connect the FT232BM to the 93C46 (93C56 or 93C66) EEPROM. EECS (pin 32) is directly connected to the chip select (CS) pin of the EEPROM. EESK (pin 1) is directly connected to the clock (SK) pin of the EEPROM. EEDATA (pin 2) is directly connected to the Data In (Din) pin of the EEPROM. There is a potential condition whereby both the Data Output (Dout) of the EEPROM can drive out at the same time as the EEDATA pin of the FT232BM. To prevent potential data clash in this situation, the Dout of the EEPROM is connected to EEDATA of the FT232BM via a 2.2k resistor.

Following a power-on reset or a USB reset, the FT232BM will scan the EEPROM to find out (a) if an EEPROM is attached to the Device and (b) if the data in the device is valid. If both of these are the case, then the FT232BM will use the data in the EEPROM, otherwise it will use it's built-in default values. When a valid command is issued to the EEPROM from the FT232BM, the EEPROM will acknowledge the command by pulling it's Dout pin low. In order to check for this condition, it is necessary to pull Dout high using a 10k resistor. If the

mand acknowledge doesn't happen then EEDATA will be pulled high by the 10k resistor during this part of the e and the device will detect an invalid command or no EEPROM present.

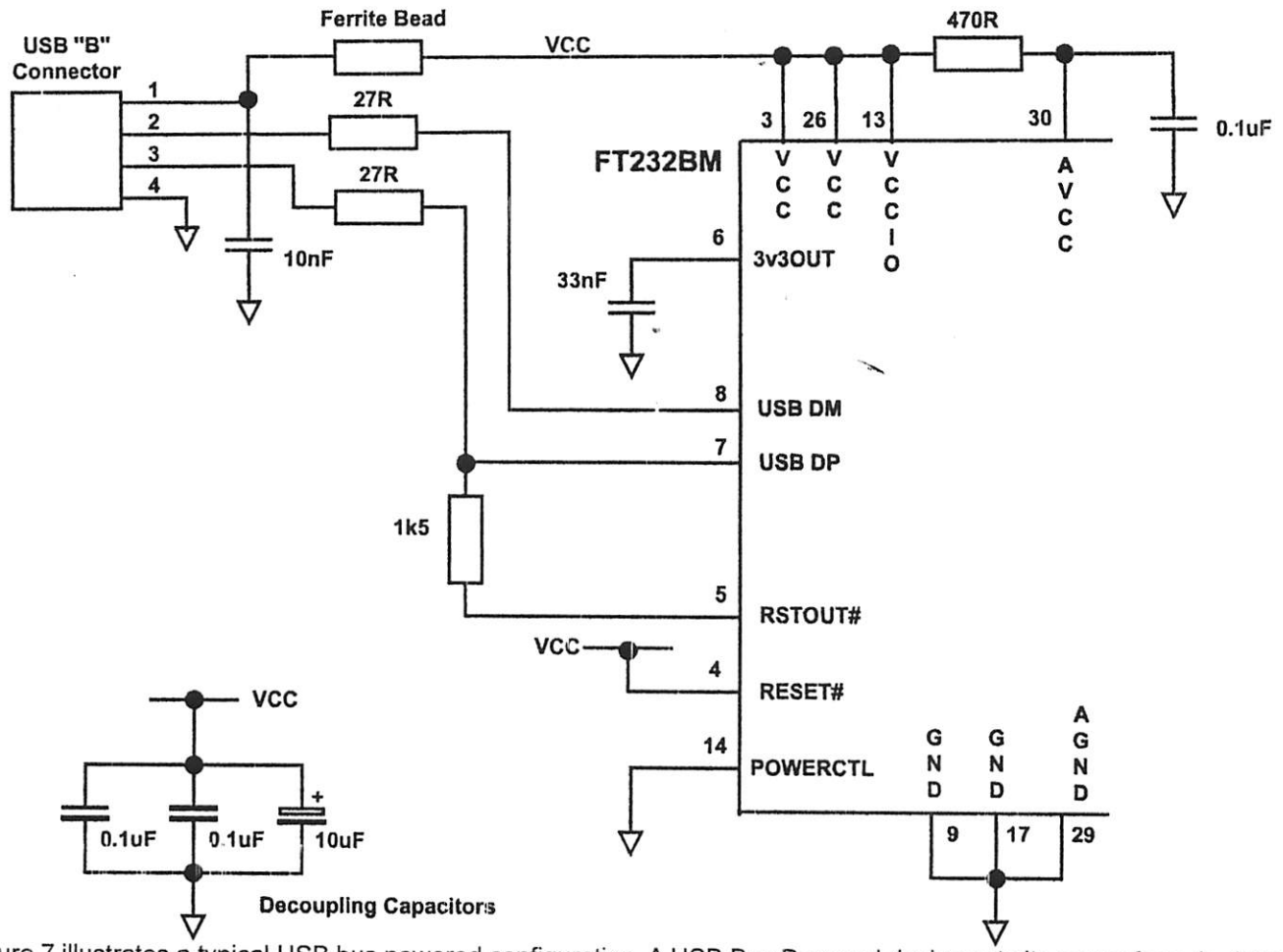
re are two varieties of these EEPROM's on the market – one is configured as being 16 bits wide, the other is figured as being 8 bits wide. These are available from many sources such as Microchip, STMicro, ISSI etc. The 32BM requires EEPROM's with a 16-bit wide configuration such as the Microchip 93LC46B device. The EEPROM st be capable of reading data at a 1Mb clock rate at a supply voltage of 4.35V to 5.25V. Most available parts are able of this.

ck the manufacturers data sheet to find out how to connect pins 6 and 7 of the EEPROM. Some devices specify se as no-connect, others use them for selecting 8 / 16 bit mode or for test functions. Some other parts have their ut rotated by 90° so please select the required part and it's options carefully.

possible to "share" the EEPROM between the FT232BM and another external device such as an MCU. However, can only be done when the FT232BM is in it's reset condition as it tri-states it's EEPROM interface at that time. pical configuration would use four bit's of an MCU IO Port. One bit would be used to hold the FT232BM reset ng RESET#) on power-up, the other three would connect to the EECS, EESK and EEDATA pins of the FT232BM rder to read / write data to the EEPROM at this time. Once the MCU has read / written the EEPROM, it would take SET# high to allow the FT232BM to configure itself and enumerate over USB.

USB Bus Powered and Self Powered Configuration

gure 7
SB Bus Powered Configuration



ure 7 illustrates a typical USB bus powered configuration. A USB Bus Powered device gets its power from the USB . Basic rules for USB Bus power devices are as follows –

On plug-in, the device must draw no more than 100mA

On USB Suspend the device must draw no more than 500uA.

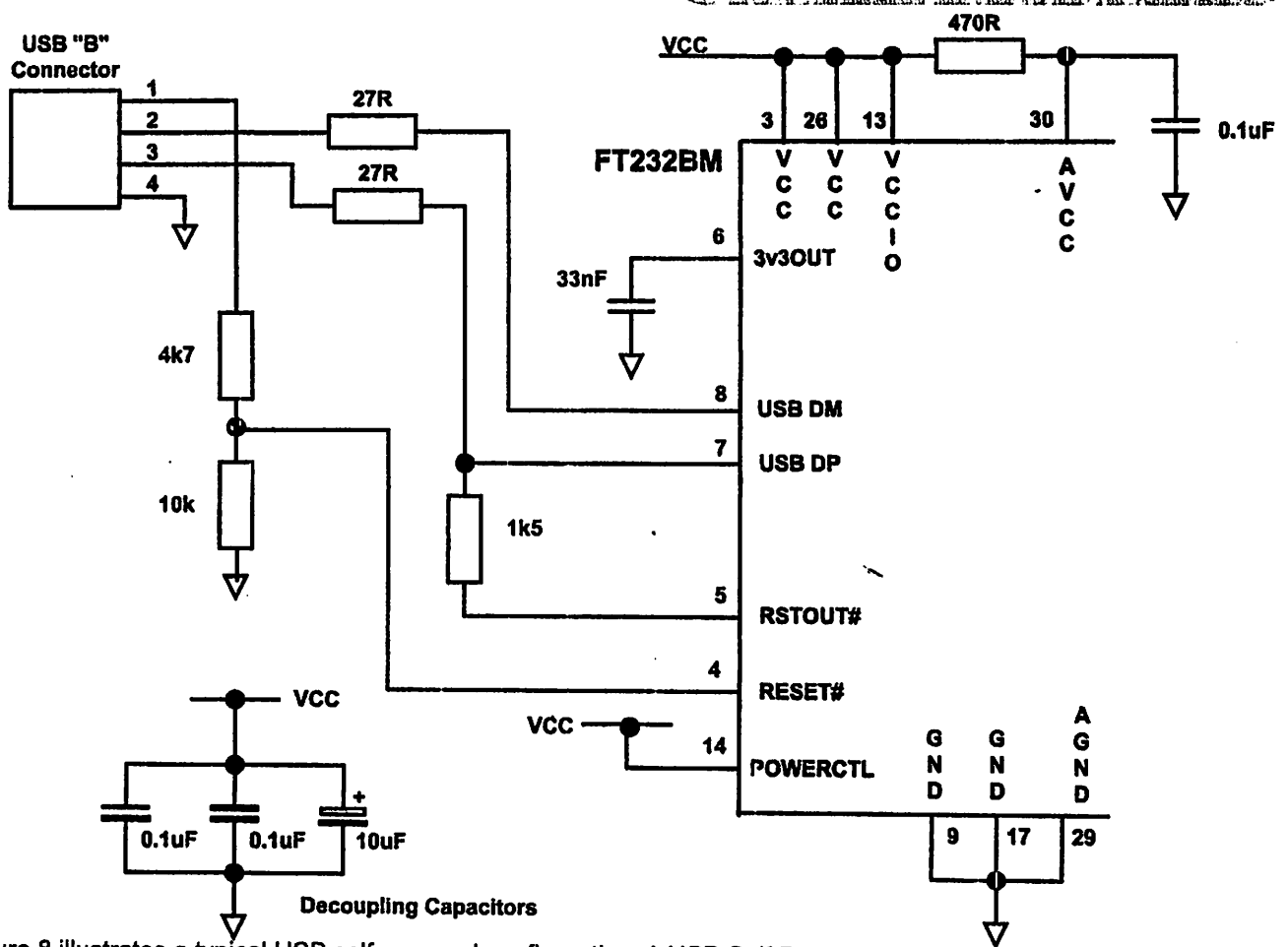
A Bus Powered High Power Device (one that draws more than 100mA) should use the PWREN# pin to keep the current below 100mA on plug-in and 500uA on USB suspend.

A device that consumes more than 100mA can not be plugged into a USB Bus Powered Hub

No device can draw more that 500mA from the USB Bus.

RCTL (pin 14) is pulled low to tell the device to use a USB Bus Power descriptor. The power descriptor in the PROM should be programmed to match the current draw of the device.

errite Bead is connected in series with USB power to prevent noise from the device and associated circuitry (EMI) ng radiated down the USB cable to the Host. The value of the Ferrite Bead depends on the total current required he circuit – a suitable range of Ferrite Beads is available from Steward (www.steward.com) for example Steward t # **MI0805K400R-00 also available as DigiKey Part #** 240-1035-1.



ire 8 illustrates a typical USB self powered configuration. A USB Self Powered device gets its power from its own WER SUPPLY and does not draw current from the USB bus. The basic rules for USB Self power devices are as ws –

A Self-Powered device should not force current down the USB bus when the USB Host or Hub Controller is powered down.

A Self Powered Device can take as much current as it likes during normal operation and USB suspend as it has its own POWER SUPPLY.

A Self Powered Device can be used with any USB Host and both Bus and Self Powered USB Hubs

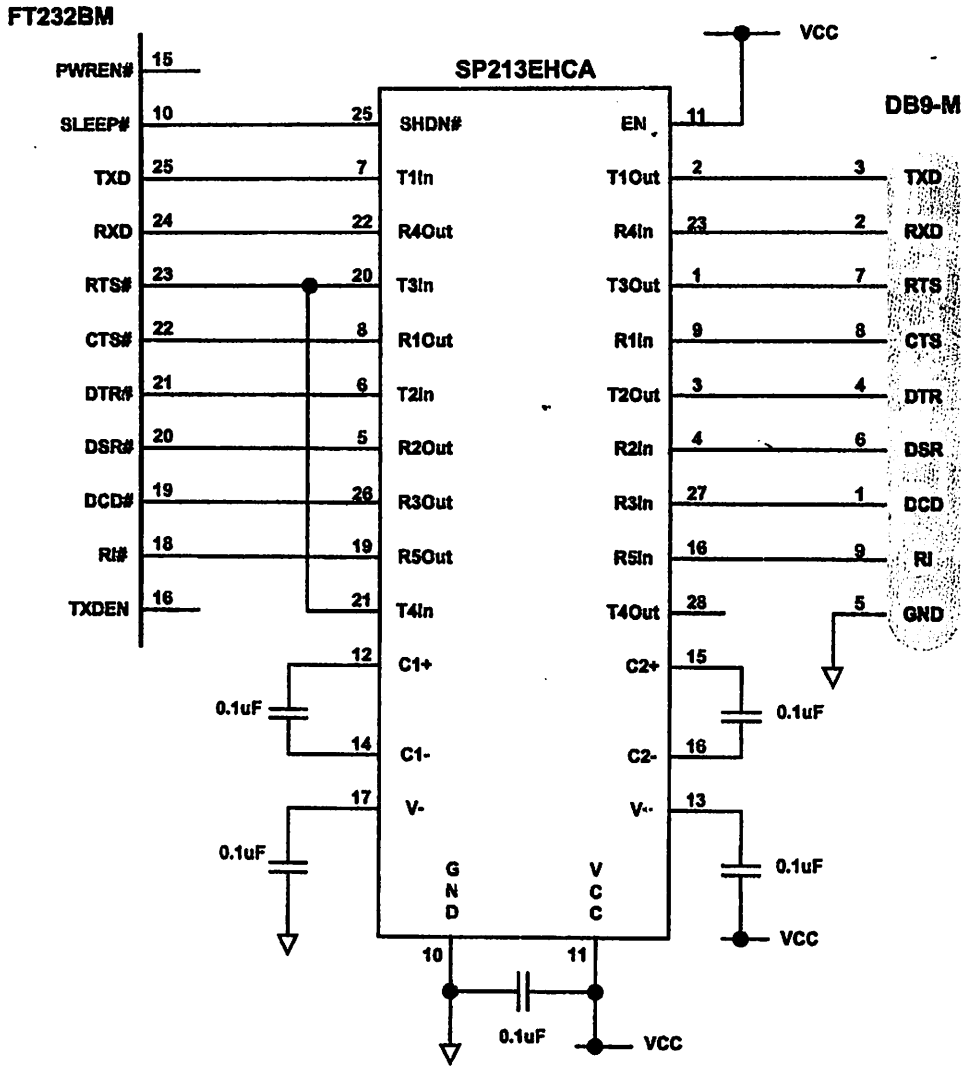
RCTL (pin 14) is pulled high to tell the device to use a USB Bus Power descriptor. The power descriptor in the 'ROM should be programmed to a value of zero. The USB power descriptor option in the EEPROM should be ıramm ed to a value of zero (self powered).

ıeet requirement a) the 1.5k pull-up resistor on USB DP is connected to RSTOUT# as per the bus-power circuit. ever, the USB Bus Power is used to control the RESET# Pin of the FT232BM device. When the USB Host or is powered up RSTOUT# will pull the 1.5k resistor on USB DP to 3.3V, thus identifying the device as a full speed ce to USB. When the USB Host or Hub power is off, RESET# will go low and the device will be held in reset. As ET# is low, RSTOUT# will also be low, so no current will be forced down USB DP via the 1.5k pull-up resistor n the host or hub is powered down. Failure to do this may cause some USB host or hub controllers to power up tically.

: When the FT232BM is in reset, the UART interface pins all go tri-state. These pins have internal 200k pull-up tors to VCCIO, so they will gently pull high unless driven by some external logic.

UART Interface Configuration

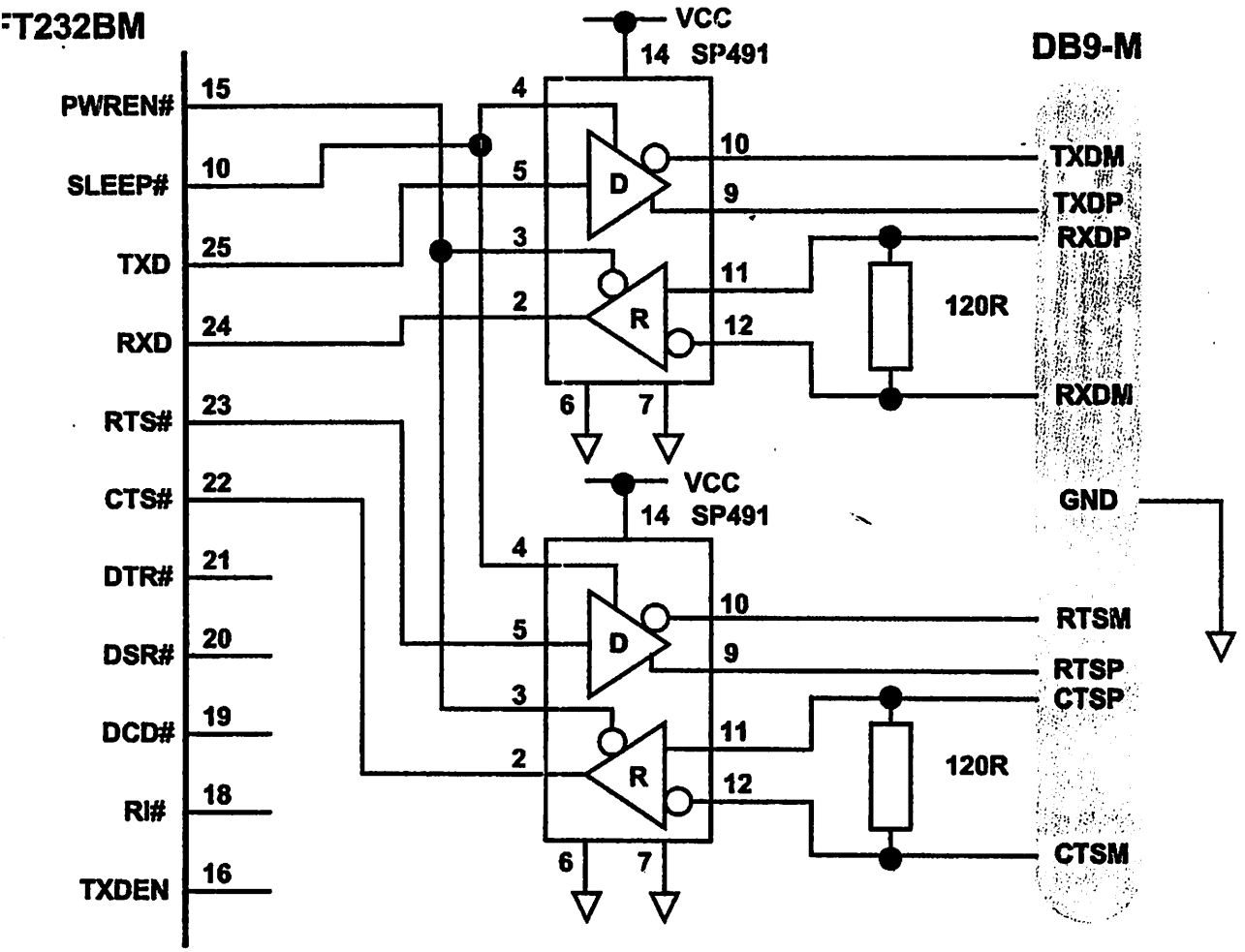
gure 9
iB <=> RS232 Converter Configuration



ure 9 illustrates how to connect the UART interface of the FT232BM to a TTL – RS232 Level Converter I.C. to ke a USB <=> RS232 converter using the popular "213" series of TTL to RS232 level converters. These devices 'e 4 transmitters and 5 receivers in a 28 LD SSOP package and feature an in-built voltage converter to convert the (nominal) VCC to the +/- 9 volts required by RS232. An important feature of these devices is the SHDN# pin which power down the device to a low quiescent current during USB suspend mode

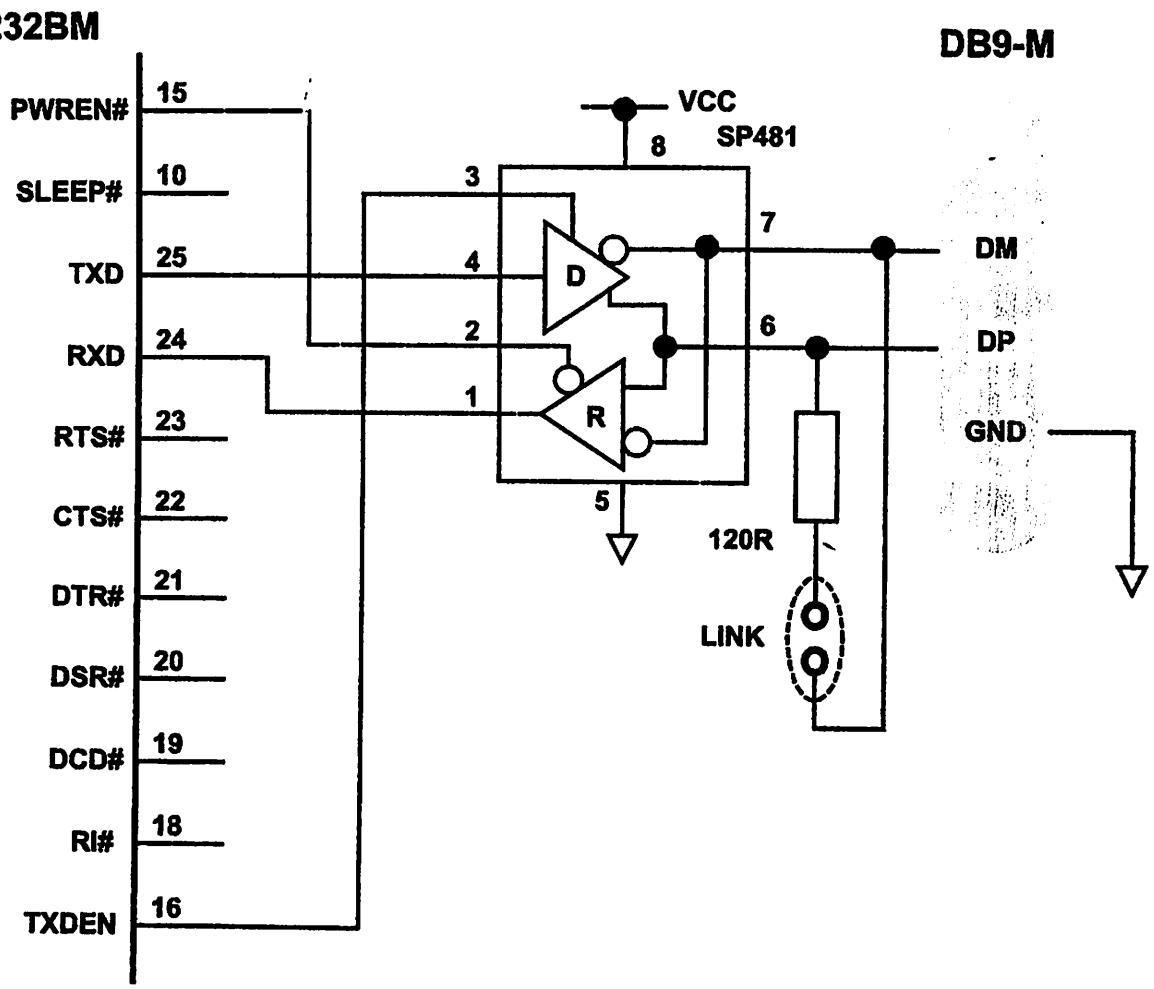
device used in the example is a Sipex SP213EHCA which is capable of RS232 communication at up to 500k id. If a lower baud rate is acceptable, then several pin compatible alternatives are available such as Sipex 213ECA , Maxim MAX213CAI and Analog Devices ADM213E which are good for communication at up to 115,200 id. If a higher baud rate is desired, use a Maxim MAX3245CAI part which is capable of RS232 communication at s of up to 1M baud. The MAX3245 is not pin compatible with the 213 series devices, also it's SHDN pin is active n so connect this to PWREN# instead of SLEEP#.

ıre 10
3 <=> RS422 Converter Configuration



'e 10 illustrates how to connect the UART interface of the FT232BM to a TTL – RS422 Level Converter I.C. to
a USB <=> RS422 converter. There are many such level converter devices available – this example uses Sipex
91 devices which have enables on both the transmitter and receiver. Because the transmitter enable is active
it is connected to the SLEEP# pin. The receiver enable is active low and is connected to the PWREN# pin. This
res that both the transmitters and receivers are enabled when the device is active, and disabled when the device
USB suspend mode. If the design is USB BUS powered, it may be necessary to use a P-Channel logic level
FET (controlled by PWREN#) in the VCC line of the SP491 devices to ensure that the USB standby current of
A is met.
SP491 is good for sending and receiving data at a rate of up to 5M Baud – in this case the maximum rate is
d to 3M Baud by the FT232BM.

ure 11
B <=> RS485 Converter Configuration



re 11 illustrates how to connect the UART interface of the FT232BM to a TTL – RS485 Level Converter I.C. to e a USB => RS485 converter. This example uses the Sipex SP491 device but there are similar parts available Maxim and Analog Devices amongst others. The SP491 is a RS485 device in a compact 8 pin SOP package. s separate enables on both the transmitter and receiver. With RS485, the transmitter is only enabled when a acter is being transmitted from the UART. The TXDEN pin on the FT232BM is provided for exactly that purpose so the transmitter enable is wired to TXDEN. The receiver enable is active low, so it is wired to the PWREN# pin sable the receiver when in USB suspend mode.

35 is a multi-drop network – i.e. many devices can communicate with each other over a single two wire cable ection. The RS485 cable requires to be terminated at each end of the cable. A link is provided to allow the cable terminated if the device is physically positioned at either end of the cable.

s example the data transmitted by the FT232BM is also received by the device that is transmitting. This is a non feature of RS485 and requires the application software to remove the transmitted data from the received stream. With the FT232BM it is possible to do this entirely in hardware – simply modify the schematic so that of the FT232BM is the logical OR of the SP481 receiver output with TXDEN using an HC32 or similar logic gate.

ure 12
al LED Configuration

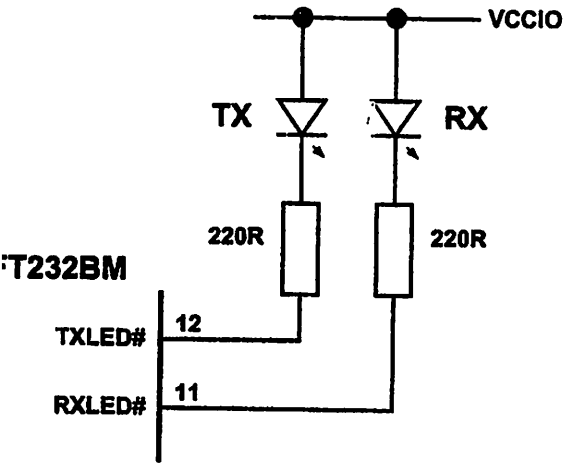
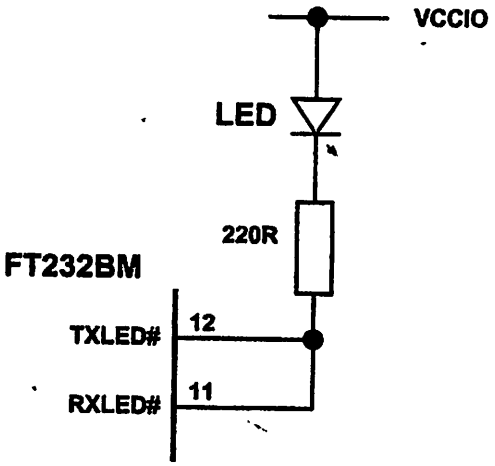


Figure 13
Single LED Configuration

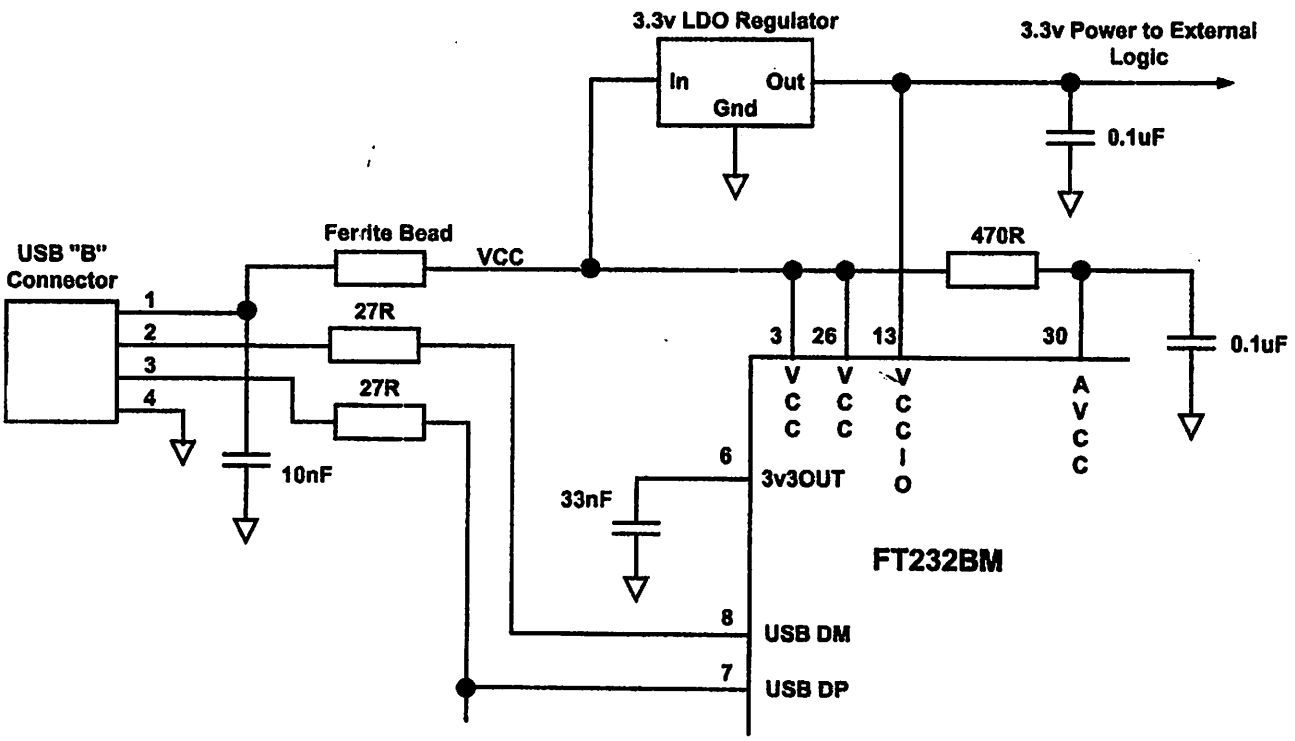


FT232BM has two IO pins dedicated to controlling LED status indicators, one for transmitted data the other
ceived data. When data is being transmitted / received the respective pins drive from tri-state to low in order
ovide indication on the LEDs of data transfer. A digital one-shot timer is used so that even a small percentage
ta transfer is visible to the end user. Figure 12 shows a configuration using two individual LED's – one for
mitted data the other for received data. In Figure 13, the transmit and receive LED indicators are wire-or'd
her to give a single LED indicator which indicates any transmit or receive data activity.
ner possibility (not shown here) is to use a 3 pin common anode tri-color LED based on the circuit in Figure 13 to
a single LED that can display activity in a variety of colors depending on the ratio of transmit activity compared to
ve activity.

that the LED's are connected to VCCIO.

Interfacing to 3.3v Logic

ure 14
s Powered Circuit with 3.3V logic drive / supply voltage

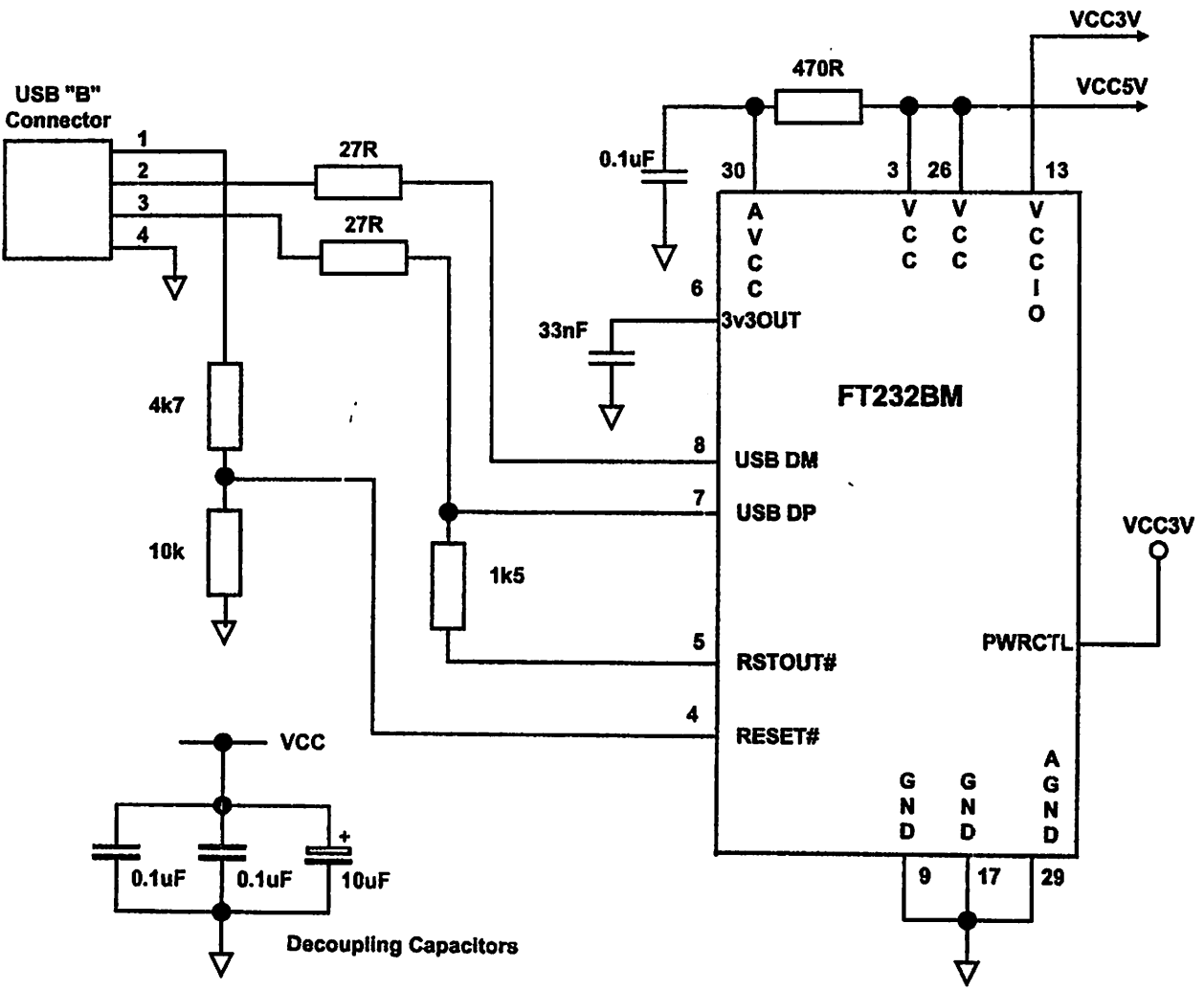


re 14 shows how to configure the FT232BM to interface with a 3.3V logic device. In this example, a discrete
' regulator is used to supply the 3.3V logic from the USB supply. VCCIO is connected to the output of the 3.3V
lator, which in turn will cause the UART interface IO pins to drive out at 3.3V level. For USB bus powered circuits
e considerations have to be taken into account when selecting the regulator –
The regulator must be capable of sustaining its output voltage with an input voltage of 4.35 volts. A Low Drop Out
(LDO) regulator must be selected.
The quiescent current of the regulator must be low in order to meet the USB suspend total current requirement of
<= 500uA during USB suspend.
xample of a regulator family that meets these requirements is the MicroChip (Telcom) TC55 Series. These
ces can supply up to 250mA current and have a quiescent current of under 1uA.

me cases, where only a small amount of current is required (< 5mA) , it may be possible to use the in-built
lator of the FT232BM to supply the 3.3v without any other components being required. In this case, connect
IO to the 3v3OUT pin of the FT232BM.

: It should be emphasised that the 3.3V supply for VCCIO in a bus powered design with a 3.3V logic interface
ld come from an LDO which is supplied by the USB bus, or from the 3V3OUT pin of the FT232BM, and not from
ther source.

ure 15
f Powered Circuit with 3.3V logic drive / supply voltage



re 15 is an example of a USB self powered design with 3.3V interface. In this case VCCIO is supplied by an rnal 3.3V supply in order to make the device IO pins drive out at 3.3V logic level, thus allowing it to be connected 3.3V MCU or other external logic. A USB self powered design uses its own power supplies, and does not draw of it's power from the USB bus. In such cases, no special care need be taken to meet the USB suspend current mA) as the device does not get it's power from the USB port.

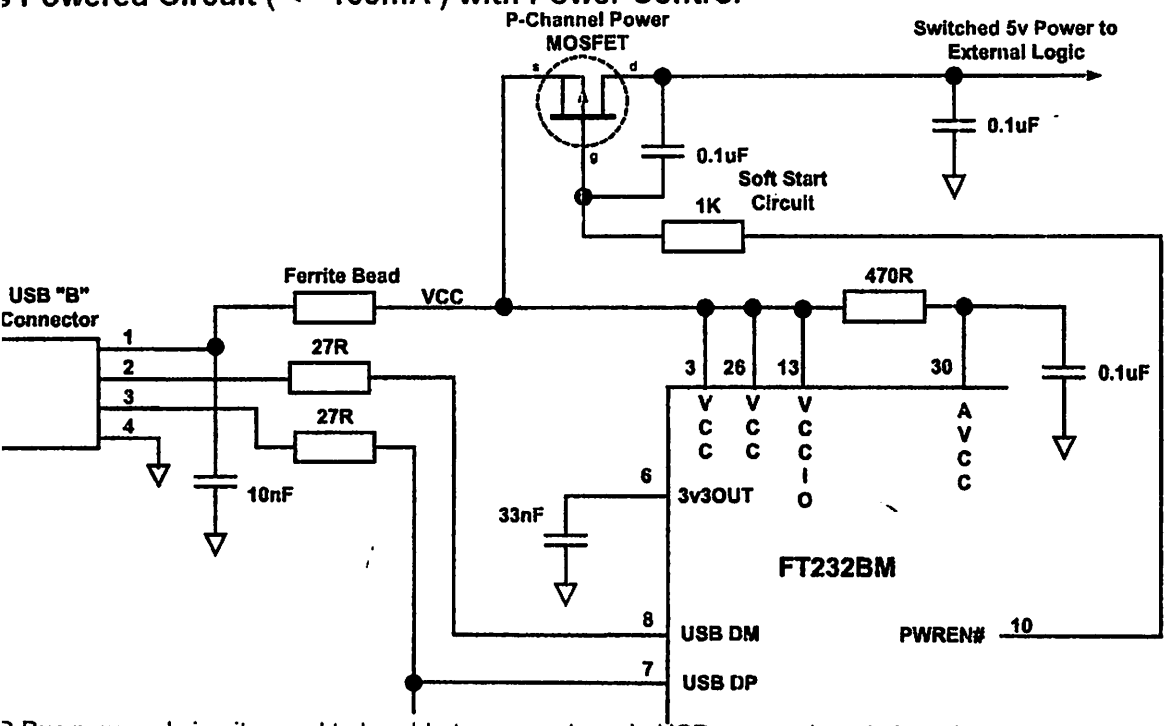
ith bus powered 3.3V interface designs, in some cases, where only a small amount of current is required (<5mA), y be possible to use the in-built regulator of the FT232BM to supply the 3.3V without any other components being ired. In this case, connect VCCIO to the 3v3OUT pin of the FT232BM.

that in this case PWRCTL is pulled up to VCCIO, not VCC.

Power Switching

ure 16

s Powered Circuit (<= 100mA) with Power Control



B Bus powered circuits need to be able to power down in USB suspend mode in order to meet the <= 500uA total end current requirement (including external logic). Some external logic can power itself down into a low current e by monitoring the PWREN# pin. For external logic that cannot power itself down in that way, the FT232BM ides a simple but effective way of turning off power to external circuitry during USB suspend.

re 16 shows how to use a discrete P-Channel Logic Level MOSFET to control the power to external logic iits. A suitable device could be a Fairchild NDT456P, or International Rectifier IRLML6402, or equivalent. It is mmended that a "soft start" circuit consisting of a 1K series resistor and a 0.1 uF capacitor are used to limit the ent surge when the MOSFET turns on. Without the soft start circuit there is a danger that the transient power e of the MOSFET turning on will reset the FT232BM, or the USB host / hub controller. The values used here v attached circuitry to power up with a slew rate of ~12.5 V per millisecond, in other words the output voltage will sition from GND to 5 V in approximately 400 microseconds.

natively, a dedicated power switch i.c. with inbuilt "soft-start" can be used instead of a MOSFET. A suitable power ch i.c. for such an application would be a Micrel (www.micrel.com) MIC2025-2BM or equivalent.

se note the following points in connection with power controlled designs –

The logic to be controlled must have it's own reset circuitry so that it will automatically reset itself when power is e-applied on coming out of suspend.

Set the Pull-down on Suspend option in the FT232BM's EEPROM.

For USB high-power bus powered device (one that consumes greater than 100 mA, and up to 500 mA of current rom the USB bus), the power consumption of the device should be set in the max power field in the EEPROM. A high-power bus powered device must use this descriptor in the EEPROM to inform the system of it's power equirements.

For 3.3V power controlled circuits VCCIO must not be powered down with the external circuitry (PWREN# gets t's VCC supply from VCCIO). Either connect the power switch between the output of the 3.3V regulator and the external 3.3V logic OR if appropriate power VCCIO from the 3V3OUT pin of the FT232BM.

Document Revision History

!32B Version 1.0 – Initial document created 30 April 2002.

!32B Version 1.1 – Updated 04 August 2002

Section 4.1 RESET# Pin description corrected (RESET# does not have an internal 200k pull-up to VCC as previously stated).

Figure 2 pin-out corrected (EECS = Pin 32).

!32B Version 1.2 – Updated 27 October 2002

Pin and package naming made consistent throughout data sheet.

Section 1.0 Updated to reflect availability of Mac OS X driver.

Section 2.0 Minor corrections.

Section 3.1 Minor changes to functional block descriptions of SIE, RESET Generator, and EEPROM interface.

Section 4.1 Note added to EEPROM interface group.

Section 4.1 RSTOUT# Pin description amended.

Section 6.1 Minimum operating supply voltage adjusted.

Section 6.1 EESK added to Note 3.

Section 6.1 UART IO pin characteristics amended.

Section 6.1 RESET#, TEST, EECS, EESK, and EEDATA pin characteristics amended.

Section 6.1 RSTOUT pin characteristics amended.

Section 7.1 Updated recommended ceramic resonator part number and circuit configuration.

Section 7.3 "USB Self Powered Configuration (1)" (original figure 8) removed. Recommended circuit for USB self powered designs updated. Subsequent figure numbers have changed as a result.

Section 7.6 Note added to description of Bus powered circuit with 3.3V logic drive / supply voltage.

Section 7.6 Self Powered Circuit with 3.3V logic drive / supply voltage added (new figure 16).

Disclaimer

) Contact Information

re Technology Devices Intl. Limited
Seorge's Studios
7 St. George's Road,
gow G3 6JA,
ed Kingdom.
+44 (0)141 353 2565
: +44 (0)141 353 2656

E-Mail (Sales) : sales@ftdichip.com
E-Mail (Support) : support@ftdichip.com
E-Mail (General Enquiries) : admin@ftdichip.com
Web Site URL : http://www.ftdichip.com

nts and Sales Representatives

e time of writing our Sales Network covers over 40 different countries world-wide. Please visit the Sales Network of our Web site for the contact details our distributor(s) in your country.

atures

- ompatible with MCS-51® Products
- K Bytes of In-System Programmable (ISP) Flash Memory
  - Endurance: 1000 Write/Erase Cycles
- 0V to 5.5V Operating Range
- lly Static Operation: 0 Hz to 33 MHz
- ree-level Program Memory Lock
- 6 x 8-bit Internal RAM
- Programmable I/O Lines
- ree 16-bit Timer/Counters
- ght Interrupt Sources
- ll Duplex UART Serial Channel
- w-power Idle and Power-down Modes
- errupt Recovery from Power-down Mode
- tchdog Timer
- al Data Pointer
- wer-off Flag
- st Programming Time
- xible ISP Programming (Byte and Page Mode)

scription

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

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

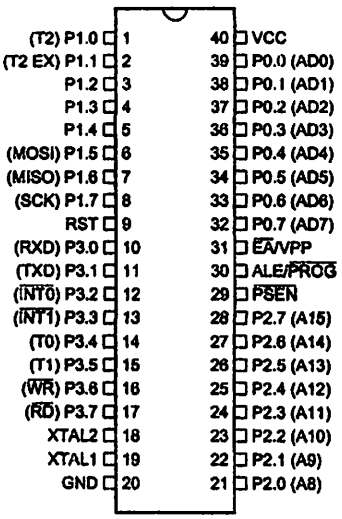
8-bit Microcontroller with 8K Bytes In-System Programmable Flash

AT89S52

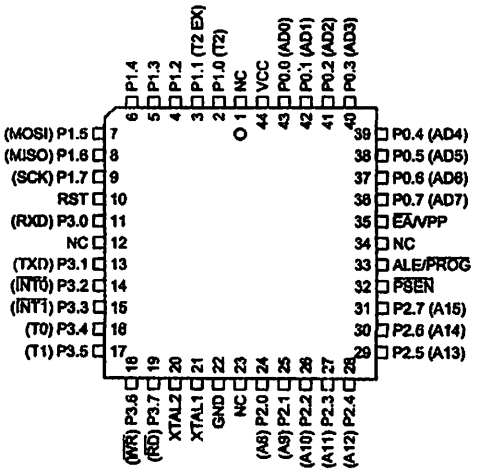
1919B–MICRO–11/03

Configurations

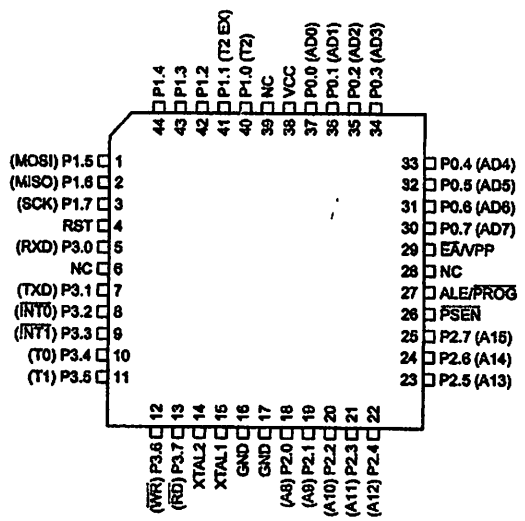
PDIP



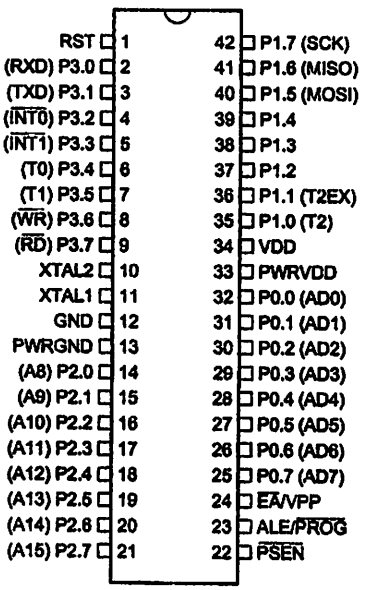
PLCC



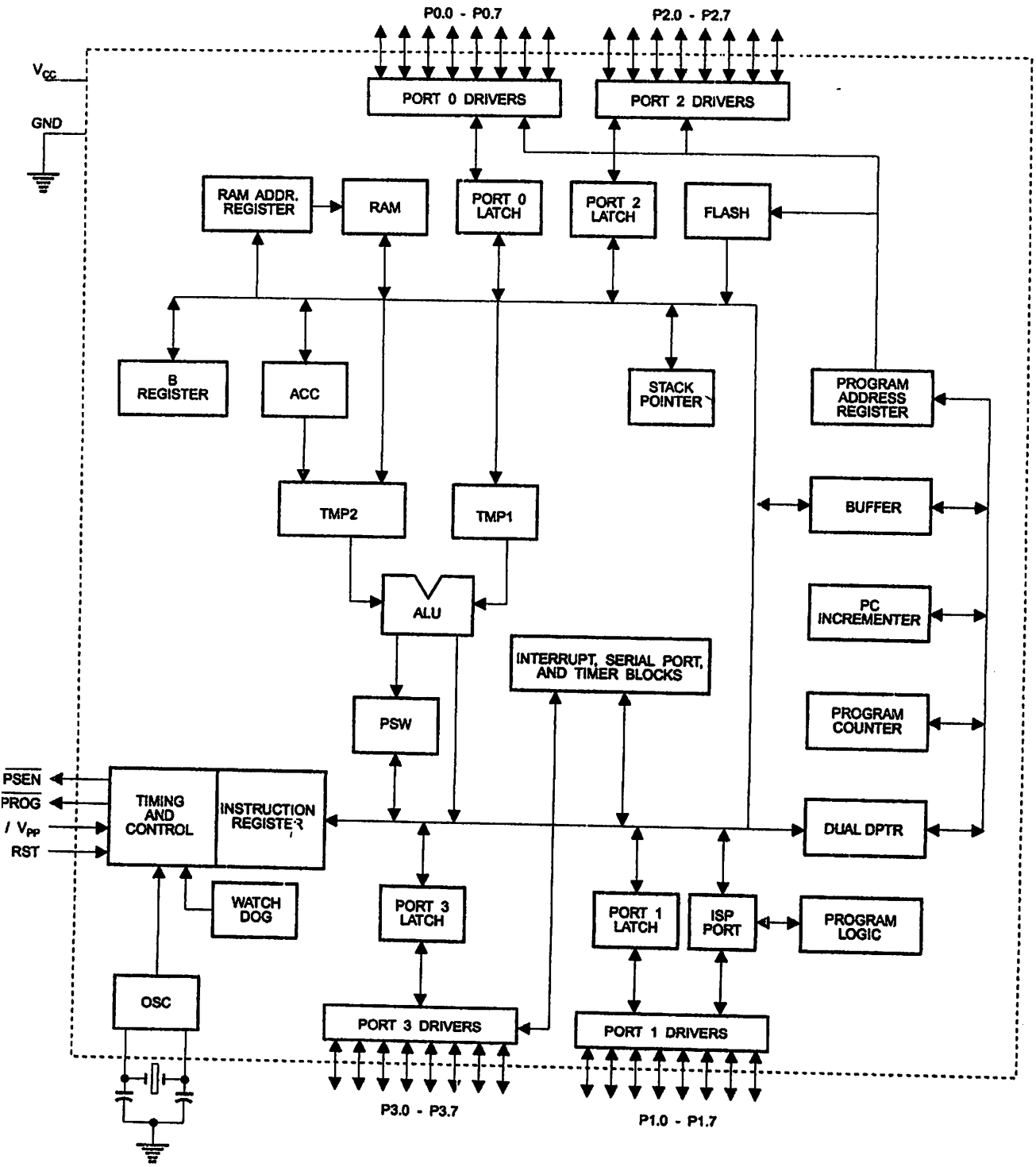
TQFP



PDIP



ock Diagram


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

Description

Supply voltage.

Ground.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

/$\overline{PROG}$

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

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

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

N

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

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

/PP

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

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

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

.1

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

.2

Output from the inverting oscillator amplifier.

le 2. T2CON – Timer/Counter 2 Control Register

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

Bit Addressable

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

mbol	Function
2	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.
F2	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).
LK	Receive clock enable. When set, causes the serial port to use Timer 2 overflow pulses for its receive clock in serial port Modes 1 and 3. RCLK = 0 causes Timer 1 overflow to be used for the receive clock.
.K	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.
EN2	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.
	Start/Stop control for Timer 2. TR2 = 1 starts the timer.
$\overline{2}$	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).
$\overline{RL2}$	Capture/Reload select. CP/$\overline{RL2}$ = 1 causes captures to occur on negative transitions at T2EX if EXEN2 = 1. CP/$\overline{RL2}$ = 0 causes automatic reloads to occur when Timer 2 overflows or negative transitions occur at T2EX when EXEN2 = 1. When either RCLK or TCLK = 1, this bit is ignored and the timer is forced to auto-reload on Timer 2 overflow.

le 3. AUXR: Auxiliary Register

XR Address = 8EH Reset Value = XXX00XX0B

Not Bit Addressable

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

Reserved for future expansion

ALE Disable/Enable ALE

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

RTO Disable/Enable Reset out

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

IDLE Disable/Enable WDT in IDLE mode

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

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

er 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 t can be set and rest under software control and is not affected by reset.

e 4. AUXR1: Auxiliary Register 1

(R1 Address = A2H Reset Value = XXXXXXX0B

Not Bit Addressable

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

Reserved for future expansion

Data Pointer Register Select

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

ɜmory Organization

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

ɔgram Memory

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

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

ta Memory

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

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

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

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

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

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

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

tchdog Timer ıe-time Enabled h Reset-out)

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

ng the WDT

To enable the WDT, a user must write 01EH and 0E1H in sequence to the WDTRST register (SFR location 0A6H). When the WDT is enabled, the user needs to service it by writing 01EH and 0E1H to WDTRST to avoid a WDT overflow. The 14-bit counter overflows when it reaches 16383 (3FFFH), and this will reset the device. When the WDT is enabled, it will increment every machine cycle while the oscillator is running. This means the user must reset the WDT at least every 16383 machine cycles. To reset the WDT the user must write 01EH and 0E1H to WDTRST. WDTRST is a write-only register. The WDT counter cannot be read or written. When WDT overflows, it will generate an output RESET pulse at the RST pin. The RESET pulse duration is 98xTOSC, where TOSC = 1/FOSC. To make the best use of the WDT, it should be serviced in those sections of code that will periodically be executed within the time required to prevent a WDT reset.

DT During Power-own and Idle

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

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

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

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

ART

The UART in the AT89S52 operates the same way as the UART in the AT89C51 and AT89C52. For further information on the UART operation, refer to the ATMEL Web site (http://www.atmel.com). From the home page, select "Products", then "8051-Architecture Flash Microcontroller", then "Product Overview".

ner 0 and 1

Timer 0 and Timer 1 in the AT89S52 operate the same way as Timer 0 and Timer 1 in the AT89C51 and AT89C52. For further information on the timers" operation, refer to the ATMEL Web site (http://www.atmel.com). From the home page, select "Products", then "8051-Architecture Flash Microcontroller", then "Product Overview".

ner 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 5. Timer 2 consists of two 8-bit registers, TH2 and TL2. In the Timer function, the TL2 register is incremented every machine cycle. Since a machine cycle consists of 12 oscillator periods, the count rate is 1/12 of the oscillator frequency.

Table 5. Timer 2 Operating Modes

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

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

ɔture 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.

:o-reload (Up or Down ınter)

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 6). 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.

ıre 1. Timer in Capture Mode

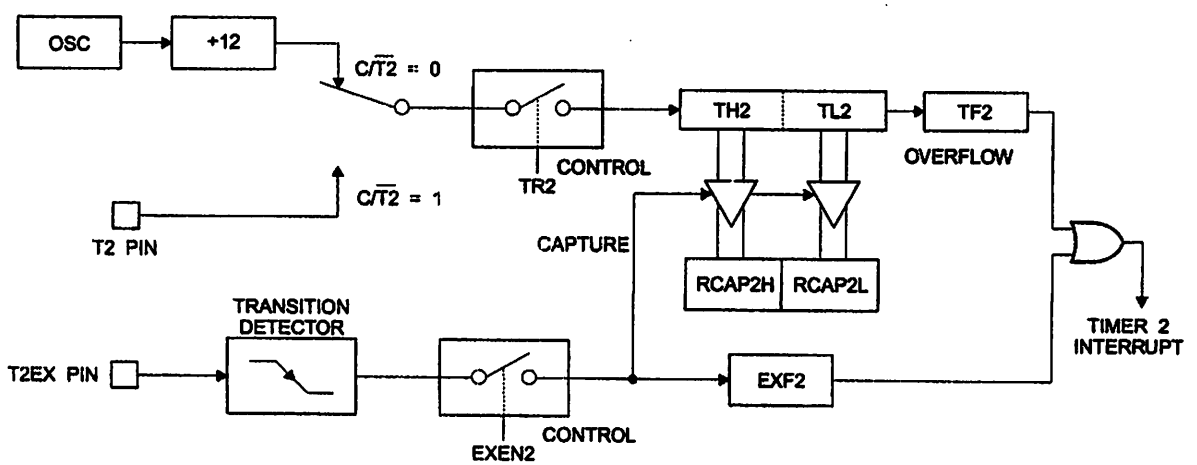


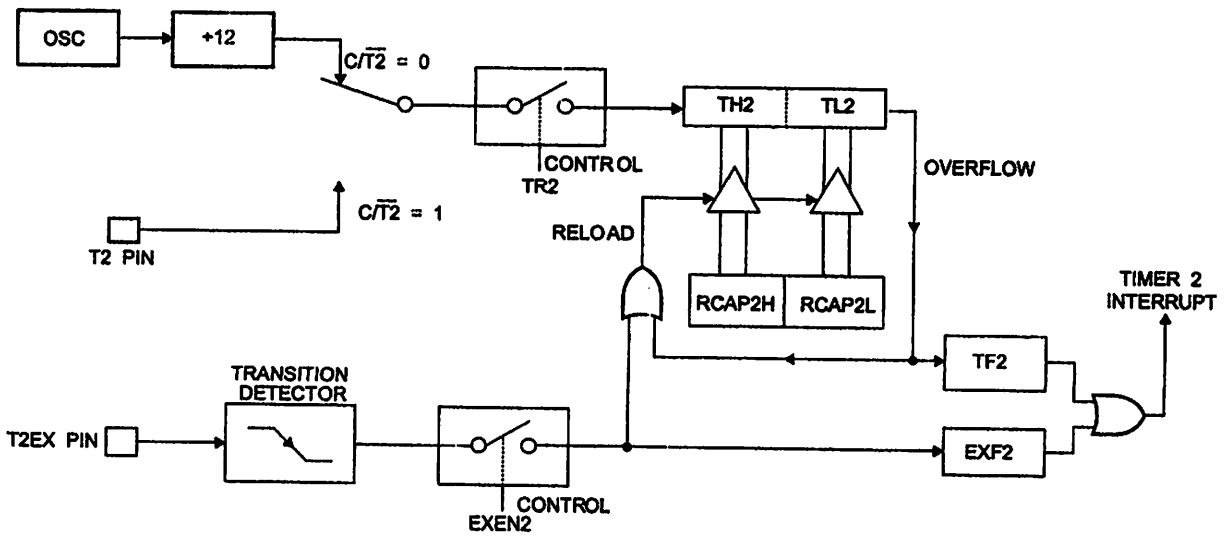
Figure 2 shows Timer 2 automatically counting up when DCEN = 0. In this mode, two options are selected by bit EXEN2 in T2CON. If EXEN2 = 0, Timer 2 counts up to 0FFFFH and then sets the TF2 bit upon overflow. The overflow also causes the timer registers to be reloaded with the 16-bit value in RCAP2H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

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

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

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

ıre 2. Timer 2 Auto Reload Mode (DCEN = 0)

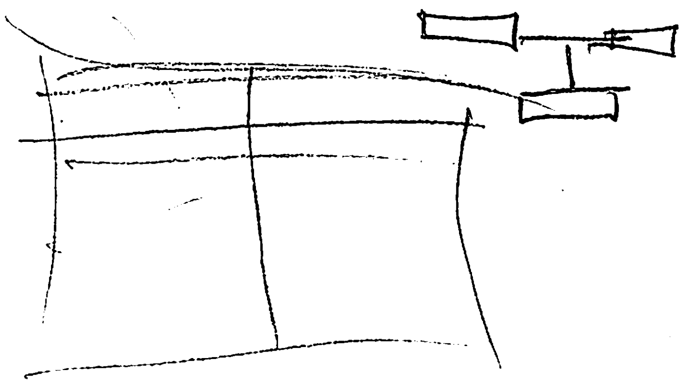


R=

HC: 80

1 a

R. 80. – 5g



le 6. T2MOD – Timer 2 Mode Control Register

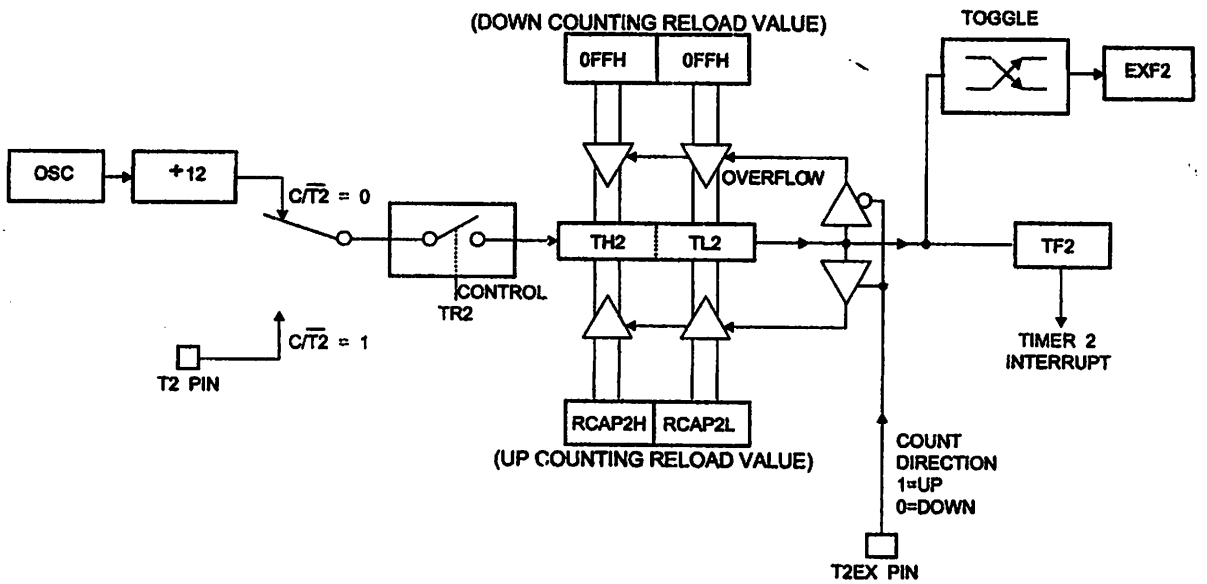
T2MOD Address = 0C9H — Reset Value = XXXX XX00B

Not Bit Addressable

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

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

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



ud Rate Generator

Timer 2 is selected as the baud rate generator by setting TCLK and/or RCLK in T2CON (Table 2). Note that the baud rates for transmit and receive can be different if Timer 2 is used for the receiver or transmitter and Timer 1 is used for the other function. Setting RCLK and/or TCLK puts Timer 2 into its baud rate generator mode, as shown in Figure 4.

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

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

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

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

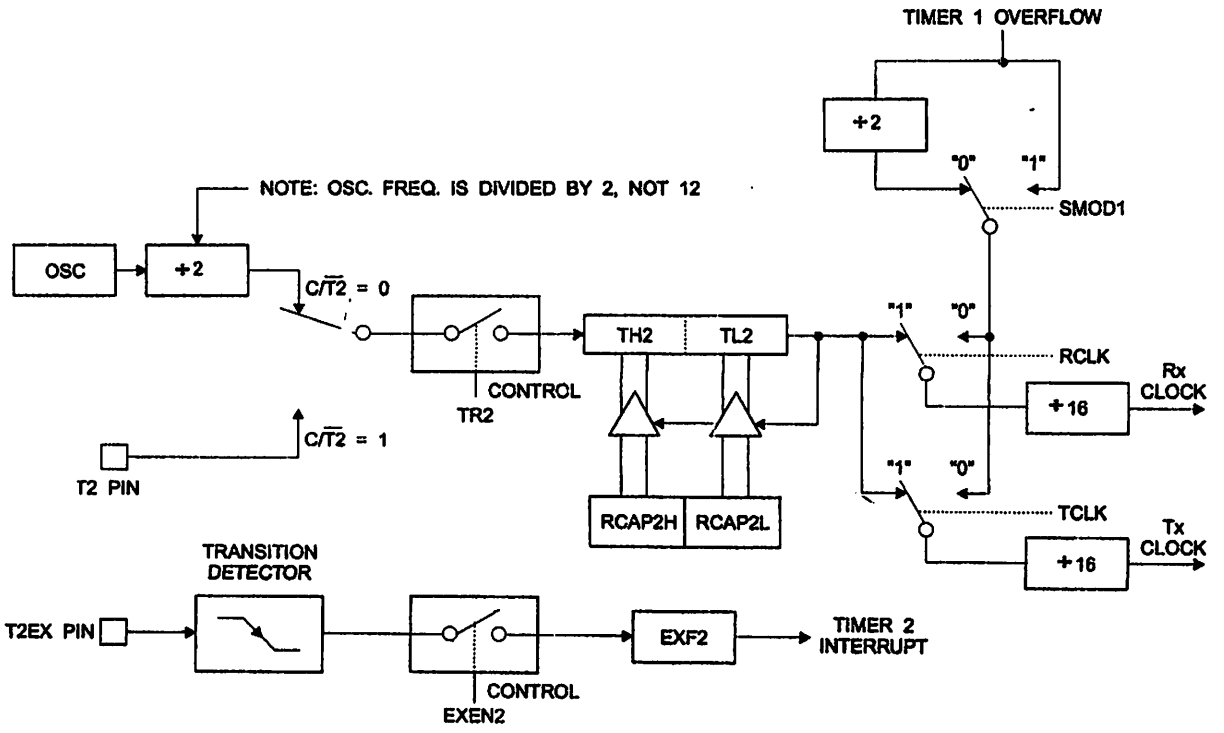
$$\frac{\text{Modes 1 and 3}}{\text{Baud Rate}} = \frac{\text{Oscillator Frequency}}{32 \times [65536\text{-RCAP2H,RCAP2L})]}$$

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

Timer 2 as a baud rate generator is shown in Figure 4. This figure is valid only if RCLK or TCLK = 1 in T2CON. Note that a rollover in TH2 does not set TF2 and will not generate an interrupt. Note too, that if EXEN2 is set, a 1-to-0 transition in T2EX will set EXF2 but will not cause a reload from (RCAP2H, RCAP2L) to (TH2, TL2). Thus, when Timer 2 is in use as a baud rate generator, T2EX can be used as an extra external interrupt.

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

ıre 4. Timer 2 in Baud Rate Generator Mode



›grammable ›ck Out

A 50% duty cycle clock can be programmed to come out on P1.0, as shown in Figure 5. This pin, besides being a regular I/O pin, has two alternate functions. It can be programmed to input the external clock for Timer/Counter 2 or to output a 50% duty cycle clock ranging from 61 Hz to 4 MHz (for a 16-MHz operating frequency).

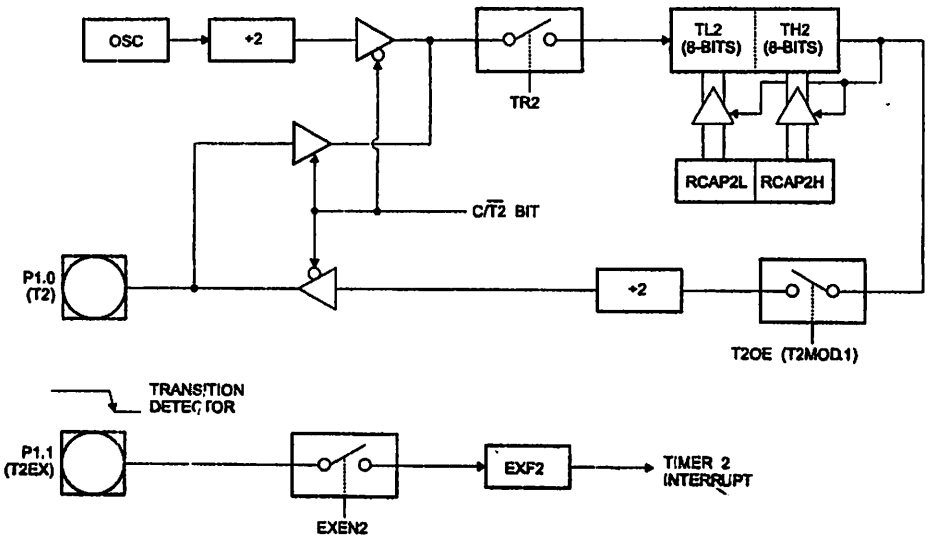
To configure the Timer/Counter 2 as a clock generator, bit C/$\overline{T2}$ (T2CON.1) must be cleared and bit T2OE (T2MOD.1) must be set. Bit TR2 (T2CON.2) starts and stops the timer.

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

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

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

ıre 5. Timer 2 in Clock-Out Mode



ɔrrupts

The AT89S52 has a total of six interrupt vectors: two external interrupts ($\overline{INT0}$ and $\overline{INT1}$), three timer interrupts (Timers 0, 1, and 2), and the serial port interrupt. These interrupts are all shown in Figure 6.

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

Note that Table 5 shows that bit position IE.6 is unimplemented. User software should not write a 1 to this bit position, since it may be used in future AT89 products.

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

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

le 7. Interrupt Enable (IE) Register

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

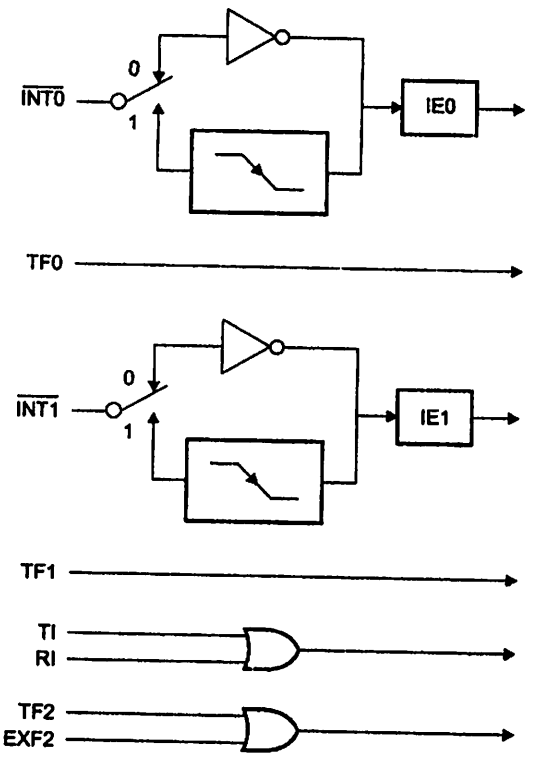
Enable Bit = 1 enables the interrupt.

Enable Bit = 0 disables the interrupt.

nbol	Position	Function
	IE.7	Disables all interrupts. If EA = 0, no interrupt is acknowledged. If EA = 1, each interrupt source is individually enabled or disabled by setting or clearing its enable bit.
	IE.6	Reserved.
?	IE.5	Timer 2 interrupt enable bit.
	IE.4	Serial Port interrupt enable bit.
	IE.3	Timer 1 interrupt enable bit.
	IE.2	External interrupt 1 enable bit.
	IE.1	Timer 0 interrupt enable bit.
	IE.0	External interrupt 0 enable bit.

r software should never write 1s to reserved bits, because they may be used in future AT89 products.

re 6. Interrupt Sources



cillator aracteristics

XTAL1 and XTAL2 are the input and output, respectively, of an inverting amplifier that can be configured for use as an on-chip oscillator, as shown in Figure 7. Either a quartz crystal or ceramic resonator may be used. To drive the device from an external clock source, XTAL2 should be left unconnected while XTAL1 is driven, as shown in Figure 8. There are no requirements on the duty cycle of the external clock signal, since the input to the internal clocking circuitry is through a divide-by-two flip-flop, but minimum and maximum voltage high and low time specifications must be observed.

e Mode

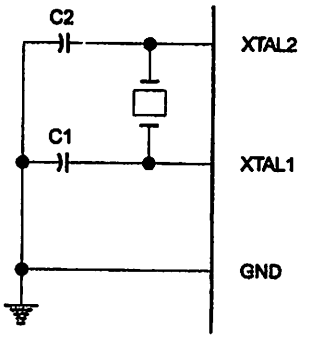
In idle mode, the CPU puts itself to sleep while all the on-chip peripherals remain active. The mode is invoked by software. The content of the on-chip RAM and all the special functions registers remain unchanged during this mode. The idle mode can be terminated by any enabled interrupt or by a hardware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

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

wer-down Mode

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

Figure 7. Oscillator Connections



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

Figure 8. External Clock Drive Configuration

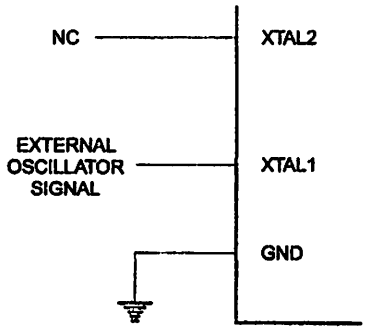


Table 8. Status of External Pins During Idle and Power-down Modes

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

›gram Memory ck Bits

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

Table 9. Lock Bit Protection Modes

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

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

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

PERMOHONAN PERSETUJUAN SKRIPSI

Yang betanda tangan dibawah ini :

Nama	: Muhammad. Bahesti.s
NIM	: ~~06.12.198~~ 06.12.318
Semester	: III.
Fakultas	: Teknologi Industri
Jurusan	: Teknik Elektro S-1
Konsentrasi	: Teknik Elektronika / ~~E. Energi Listrik~~
Alamat	:

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

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

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

Telah diteliti kebenaran data tersebut diatas
Recording Teknik Elektro

(Ivri Handayani)

Malang, 19 Desember 2007
Pemohon

(Muhammad - Bahesti)

Disetujui
Ketua Jurusan Teknik Elektro

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

Mengetahui
Dosen Wali

(Ir. F. Yudi Limpraptono, MT)
Nip. P. 1039500274

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

1. IPK 307.5 / 135 = 2.72
2. .....
3. - IMC PTR G.1.1.

prakt ?

Form. S-1a

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

BERITA ACARA SEMINAR PROPOSAL SKRIPSI
JURUSAN TEKNIK ELEKTRO S-1

Konsentrasi : Teknik Energi Listrik/Teknik Elektronika*)

1.	Nama Mahasiswa: Muhammad Bahesti S			Nim: 06.12.918
2.	Keterangan Pelaksanaan	Tanggal 19-01-08	Waktu	Tempat Ruang:
3.	Spesifikasi Judul (berilah tanda silang)**) a. Sistem Tenaga Elektrik b. Energi & Konversi Energi c. Tegangan Tinggi & Pengukuran d. Sistem Kendali Industri e. Elektronika & Komponen (f) Elektronika Digital & Komputer g. Elektronika Komunikasi h. lainnya			
4.	Judul Proposal yang diseminarkan Mahasiswa	Perancangan perangkat keras USB pada sistem pengamanan kode lisensi software menggunakan metode Kriptografi DES		
5.	Perubahan Judul yang diusulkan oleh Kelompok Dosen Keahlian	Perancangan Perangkat keras USB pada sistem pengamanan kode lisensi software menggunakan metode Kriptografi DES		
6.	Catatan: - Perubahan Judul			
7.	Catatan: Judul : Perancangan Hardware perangkat keras untuk sistem pengamanan kode lisensi software menggunakan metode kriptografi DES melalui USB			

Persetujuan Judul Skripsi

Disetujui, Dosen Keahlian I Ir. Yusuf Ismail Nakoela, MT	Disetujui, Dosen Keahlian II .....
Mengetahui, Ketua Jurusan. Ir. F. Yudi Limpraptono, MT NIP. P. 1039500274	Disetujui, Calon Dosen Pembimbing ybs M. ASHARI

Perhatian:

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

Form S-3c