PENGGUNAAN LOCAL AREA NETWORK (LAN) UNTUK PENGENDALIAN SHUT DOWN PERSONAL COMPUTER (PC)

TUGAS AKHIR

Diajukan sebagai Salah Satu syarat Untuk

Memperoleh Gelar Sarjana Strata 1 Jurusan Teknik Elektro

Disusun oleh :

Nama : Faisal

No.Mahasiswa : 99524018

JURUSAN TEKNIK ELEKTRO

FAKULTAS TEKNOLOGI INDUSTRI

UNIVERSITAS ISLAM INDONESIA

JOGJAKARTA

2007

LEMBAR PENGESAHAN PEMBIMBING

PENGGUNAAN LOCAL AREA NETWORK (LAN) UNTUK PENGENDALIAN SHUT DOWN PERSONAL COMPUTER (PC)

TUGAS AKHIR



Oleh:

Nama : Faisal

No.Mahasiswa: 99524018

Jogjakarta, 15 Maret 2007

Pembimbing I,	Pembimbing II
Tito Yuwono, ST, MSc	Dwi Ana Ratna Wati, ST

LEMBAR PENGESAHAN PENGUJI

PENGGUNAAN LOCAL AREA NETWORK (LAN) UNTUK PENGENDALIAN SHUTDOWN PERSONAL COMPUTER (PC)

TUGAS AKHIR

Oleh:

Nama : Faisal

No.Mahasiswa : 99524018

Telah Dipertahankan di Depan Sidang Penguji sebagai Salah Satu Syarat untuk Memperoleh Gelar Sarjana Teknik Elektro Fakultas Teknologi Industri Universitas Islam Indonesia

Jogjakarta, 30 Maret 2007

Tim Penguji,

Drs.Abdul Halim
Ketua

Tito Yuwono, ST, MSc
Anggota I

Dwi Ana Ratna Wati, ST
Anggota II

Mengetahui,

Ketua Jurusan Teknik Elektro Fakultas Teknologi Industri

Universitas Islam Indonesia

Tito Yuwono, ST, Msc

HALAMAN PERSEMBAHAN

Ku persembahkan tugas akhir ini kepada

Bapak dan Ibu tersayang

Kedua adikku Ashar dan Faikah yang tersayang

Dewi S Bonitasari yang selalu menemaniku dalam suka dan duka.

Saudaraku Eko dan Nurul terima kasih segalanya

Teman – temanku penghuni Asrama Bontang terima kasih atas bantuan kalian

Pengurus Himpunan Mahasiswa Bontang Jogjakarta

Dan semua yang tidak kusebutkan namanya

Terima kasih doa dan bantuannya

MOTTO

Apapun yang dapat dilihat dengan mata,

Sesulit apapun kita pasti dapat mengerjakannya, dan berani memulai

Adalah suatu kemenangan.

Sesungguhnya sesudah kesulitan ada kemudahan ,

maka apabila kamu telah selesai mengerjakan sesuatu urusan,

kerjakanlah sungguh-sungguh urusan yang lain.

Dan hanya kepada tuhanmulah hendaknya kamu berharap

(Q.S Alam Nasyrah : 6-8)

KATA PENGANTAR

Puji syukur penulis panjatkan kehadirat Allah SWT dan sholawat serta salam teruntuk Nabi Muhammad SAW yang telah memberikan rahmat, hidayah dan syafaat-Nya kepada penulis sehingga dapat menyelesaikan tugas akhir ini dengan judul " Penggunaan Local Area Network (LAN) untuk pengendalian shutdown Personal Computer (PC) "

Tujuan penulisan tugas akhir ini adalah sebagai syarat kelulusan pada Program Strata 1 (S1) Jurusan Teknik Elektro Fakultas Teknologi Industri pada Universitas Islam Indonesia Jogjakarta.

Dengan selesainya tugas akhir ini penulis menyampaikan ucapan terima kasih dan penghargaan sebesar-besarnya kepada :

1. Bapak Fathul Wahid, ST, MSc selaku Dekan Fakultas Teknologi Industri Universitas Islam Indonesia Jogjakarta
2. Bapak Tito Yuwono, ST, MSc selaku Kepala Jurusan Teknik Elektro Universitas Islam Indonesia Jogjakarta serta dosen pembimbing I.
3. Ibu Dwi Ana Ratnawati, ST selaku dosen pembimbing II
4. Bapak dan Ibu dosen Teknik Elektro Universitas Islam Indonesia Jogjakarta yang telah mengajarkan dan memberikan ilmu dibidang Elektro.
5. Kedua Orang tuaku yang selalu memberikan doa, membiayai kuliah dan sabar menunggu hingga dapat menyelesaikan pendidikan Strata 1.
6. Kedua adikku yang selalu memberikan motivasi
7. Dewi S.Bonitasari yang telah sabar menunggu dan memberikan doa.
8. Teman2 seperjuangan di Asrama Bontang yang selalu mendukung susah dan senang.

Bagaimanapun juga penulis adalah seorang manusia biasa dan penuh dengan kekurangan. Hanyalah terima kasih yang dapat kami berikan apabila pembaca turut memberikan saran dan kritik agar tugas akhir ini selalu mendekati sempurna. Tugas akhir ini akan sangat berarti jika dapat memberikan manfaat yang tiada pernah habis bagi penulis khususnya dan bagi semua pihak yang bersangkutan pada umumnya.

Semoga

Wassalam

Jogjakarta, Maret 2007

Penulis

ABSTRAKSI

Aktivitas dalam suatu gedung perkantoran atau pun lembaga pendidikan tidak dapat dipisahkan dengan komputer. Komputer-komputer tersebut dapat dihubungkan satu dengan lainnya dalam suatu jaringan. Jaringan dalam lokasi tertentu disebut LAN (*Local Area Network*). Salah satu keuntungan dalam jaringan adalah memungkinkan pengendalian terpusat atas komputer-komputer yang terhubung. Pengendalian pada penelitian ini lebih ditekankan untuk melakukan *shutdown* komputer dalam jaringan, sehingga komputer-komputer tadi tidak perlu dimatikan satu persatu pada komputer bersangkutan.

Pada penelitian yang berjudul Penggunaan Local Area Network (LAN) untuk pengendalian shut down Personal Computer (PC) ini, sistem operasi yang digunakan adalah windows XP dan mensyaratkan tiap komputer dalam jaringan mempunyai LAN *card* agar dapat terhubung satu dengan lainnya. Komputer yang bertindak sebagai pengendali adalah komputer yang diinstal program Shutdown. Prosedur pengendalian *shutdown*, yaitu mula-mula komputer pengendali mendeteksi semua komputer yang terpasang dalam jaringan. Komputer yang terdeteksi akan diketahui informasi *IP address*, dan statusnya. Untuk kemudian, *IP address* sebagai penentu komputer yang akan dimatikan.

Kata kunci : LAN, *shutdown*, *IP address*

DAFTAR ISI

Lembar Pengesahan Pembimbing.....ii
Lembar Pengesahan Penguji.....iii
Halaman Persembahan.....iv
Halaman Motto.....v
Kata Pengantar.....vi
Abstrak.....viii
Daftar Isi.....ix
Daftar Tabel.....xii
Daftar Gambar..... xiii

BAB I PENDAHULUAN 1
1.1 Latar Belakang Masalah 1
1.2 Rumusan Masalah 3
1.3 Batasan Masalah 3
1.4 Tujuan Tugas Akhir 3
1.5 Manfaat 4
1.6 Sistematika Penulisan 4

BAB II LANDASAN TEORI 6
2.1 Pengenalan LAN (Local Area Network) 6
2.1.1 Sejarah Singkat LAN 6
2.1.2 Kategori Utama pada LAN 7
2.2 Prinsip Dalam komunikasi data 8
2.2.1. Protokol Komunikasi 9
2.3 Beberapa Hadware jaringan yang penting 10
2.3.1 Kabel 10
2.3.2 Network Card 12
2.3.3 Bride 13
2.3.4 Router 14

DAFTAR TABEL

Tabel 2.1 Kategori kabel twisted pair 11

Tabel 2.2. Contoh IP address 15

Table 2.3 Kelas IP Address 15

BAB I

PENDAHULUAN

1.1 Latar Belakang Masalah

Peradaban dunia saat ini menunjukkan adanya perkembangan ilmu pengetahuan dan teknologi. Jika diamati, teknologi komputer memiliki perkembangan yang sangat cepat di mana begitu banyak teknologi-teknologi yang bermunculan. Salah satu contoh perkembangan teknologi komputer yang sangat terlihat adalah di bidang jaringan, yaitu jaringan lokal *LAN* (*Local Area Network*).

LAN beruang lingkup sebuah bangunan, LAN merupakan jaringan milik pribadi di dalam sebuah gedung atau bangunan yang berukuran sampai beberapa kilometer. LAN sering kali digunakan untuk menghubungkan komputer-komputer pribadi dan workstation dalam kantor, perusahaan atau pabrik-pabrik untuk memakai bersama *resource* (misalnya printer) dan saling bertukar informasi. LAN merupakan jaringan yang tidak mempunyai *router*. Sebelum merancang sebuah jaringan, ada beberapa hal yang harus diperhatikan untuk memutuskan tentang kebutuhan dari jaringan tersebut. Dalam perecanaan sebuah jaringan berdasarkan pada hardware yang sudah ada, namun cara ini bersifat jangka pendek karena mengasumsikan jaringan hanya sebagai sistem fisik. Cara yang lain adalah dengan melihat aliran kerja dan informasi dalam sebuah organisasi, sehingga nantinya jaringan yang ada bisa membantu dalam pengumpulan data penyebarluasan

informasi yang mendukung proses pengambilan keputusan. Keputusan yang harus dibuat ketika menset-up sebuah jaringan adalah apakah sesorang akan diberikan hak kontrol terhadap jaringan tersebut atau tidak. Salah satu tipe jaringan yang ada adalah *client/server*, sebuah komputer difungsikan khusus menjadi server yang mengkontrol semua akses ke jaringan, sehingga hanya ada satu orang saja (*administrator*) yang mempunyai hak untuk mengontrol semua akses ke jaringan tersebut.

Komunikasi data yang terjadi antar komputer dapat terkontrol dengan adanya jaringan lokal dengan topologi jaringan yang sederhana sekalipun. Jaringan lokal ini dapat digunakan sebagai alat pemindahan data dari satu komputer ke komputer lain, sehingga pengguna dapat berkomunikasi dengan mudah dalam ruang lingkup satu gedung dari masing-masing komputer (PC). Dengan sistem piranti lunak (*software*) tertentu berbasis *client/server*, jaringan lokal LAN mampu untuk mengendalikan secara automatis komputer lainnya dengan sistem yang terintegrasi. Di dalam penelitian ini digunakan piranti lunak *Microsoft Visual Basic 6.0* sebagai alat komunikasi shutdown komputer klien, dengan pengendalian topologi jaringan sederhana. *Micrososft Visual Basic 6.0* merupakan salah satu piranti lunak yang dapat membangun topologi jaringan dalam pengendalian sistem jaringan, dikarenakan di dalamnya terdapat komponen pendukung untuk berkomunikasi antar komputer dengan fasilitas komponen *Winshock*.

BAB II

LANDASAN TEORI

2.1 Pengenalan LAN (Local Area Network)

Jaringan *(network)* komputer merupakan hal yang populer di kalangan masyarakat modern saat ini. Biasanya dipakai oleh perkantoran-perkantoran, baik perkantoran kecil maupun perkantoran besar yang berbasiskan komunikasi jaringan. Pada dasarnya jaringan computer merupakan interkoneksi antara komputer satu dengan komputer lainnya di dalam sebuah gedung. Jaringan ini dikenal dengan sebutan LAN (*Local Area Network*)

Dengan menggunakan LAN, seseorang dapat berkomunikasi dengan orang lain dalam waktu yang relative sangat cepat dan efisien. Bentuk dan tujuan LAN sangat menguntungkan, karena bisa mudah mengkomunikasikan dan menghubungkan segala sesuatu dengan cepat, sehingga pemanfaatan waktu kerja sangat efisien.

Perkembangan internet dan intranet, kemampuan jaringan LAN dapat ditingkatkan untuk keperluan yang lebih universal. Yaitu internet yang merupakan suatu jaringan yang dapat berkomunikasi secara global di seluruh Negara dengan perantara satelit.

2.1.1 Sejarah Singkat LAN

Tahun 1984 Microsoft Corporation dan IBM mengeluarkan produk terbarunya, yakni Microsoft dengan *MS-NET* serta IBM dengan PC network program. Keduanya masih berbasis *MS-DOS*. Pada tahun yang sama perusahaan yang bergerak di bidang software lainnya, yakni Novel.Inc memperkenalkan hasil

merupakan media guided atau media yang menggunakan suatu perantara dalam pengiriman data yang digunakan, yang diantaranya adalah *coaxial cable* dan fiber optic. Biasanya dalam penggunaannya broadband dapat membawa lima sampai enem saluran komunikasi yang berbeda.

b. *Baseband LAN*; merupakan suatu media juga yang pada dasarnya hampir sama dengan broadband, tetapi baseband mempunyai perbedaan hanya menggunakan satu saluran komunikasi pada kabel untuk mengirimkan data yang dimungkinkan untuk menunjang pengiriman sinyal digital yang lebih cepat dari pada pengiriman sinyal analog.

LAN juga mempunyai ciri khas yang mudah sekali diketahui, yaitu penggunaan *data, printer, scanner*, serta harddisk secara bersama-sama. LAN juga mempunyai keuntungan, diantaranya dapat men-*share* data, dapat men-*share software*, dapat digunakan untuk *electronic mail*, dapat mengkomunikasikan data dengan kecepatan tinggi, serta proses LAN tersebut sudah terdistribusi.

2.2 Prinsip dalam Komunikasi Data

Komunikasi data merupakan suatu interaksi atau hubungan yang diawali dengan adanya pertukaran data dari lokasi yang satu dengan lokasi yang lainnya, sehingga interaksi ini terjadi hubungan yang disebut komunikasi data. Komponen-komponen utama yang digunakan dalam komunikasi data diantaranya :

2.2.1. Protokol Komunikasi

Protokol komunikasi merupakan suatu aturan transmisi data yang akan keluar ataupun masuk. Dalam pengaturan ini dibutuhkan alat yang dinamakan Gateway yang berfungsi sebagai penerjemah protocol komunikasi kompatibel agar jaringan dapat berfungsi dengan baik.

a. *Host Computer*; merupakan seperangkat komputer yang digunakan untuk menyimpan informasi, dari sinilah informasi yang ada ditransmisikan. Host Computer memiliki kemampuan pemrosesan data yang dapat menjadi suatu komputer mainframe atau komputer mini. Host yang berupa mainframe bekerja dengan peralatan yang dikenal sebagai *Front-End-Processor (FEP)* yang merupakan komputer mini yang mengelola komunikasi data dari jaringan-jaringan untuk host mainframe.
b. *Receiving Computer*; merupakan suatu dumb terminal yang digunakan sebagai suatu penghubung antara dunia luar dan dunia dalam, yaitu sebagai input dan output yang diimplementasikan sebagai komponen dasar, yaitu keyboard, display monitor, dan beberapa port untuk komunikasi penghubungnya.
c. Data; merupakan informasi yang sudah diolah dan siap dikirimkan dari suatu lokasi ke lokasi lainnya. Misalnya data kepegawaian, administrasi perusahaan, inventarisasi perusahaan, dan sebagainya.
d. Komponen Transmisi; berfungsi untuk mempersiapkan data yang akan dikirim dari suatu lokasi tertentu ke lokasi lain. Biasanya komponen transmisi ini berbentuk saluran komunikasi, modem, dan mode transmisi.

2.3 Beberapa Hardware Jaringan Yang Penting

2.3.1. Kabel

Ada beberapa tipe pengkabelan untuk penggunaan jaringan yang dapat digunakan:

a. Thin Ethernet (Thinnet)

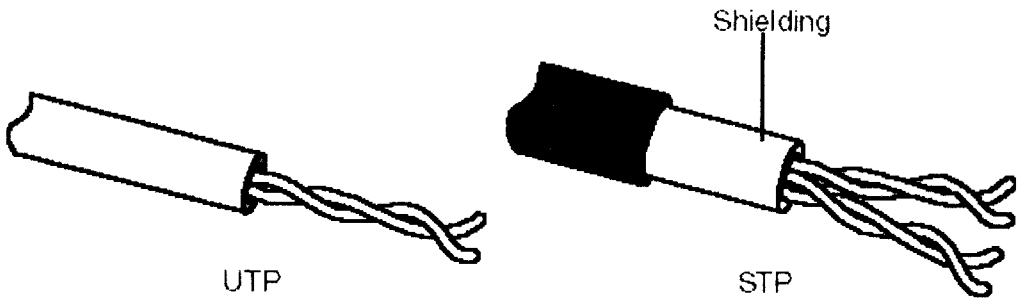
Thin Ethernet atau Thinnet memiliki keunggulan dalam hal biaya yang relatif lebih murah dibandingkan dengan tipe pengkabelan lain, serta pemasangan komponennya lebih mudah. Panjang kabel thin coaxial/RG-58 antara 0.5 – 185 m dan maksimum 30 komputer terhubung.

b. Thick Ethernet (Thicknet)

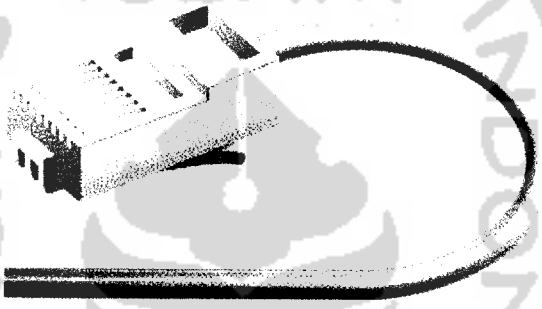
Dengan thick Ethernet atau thicknet, jumlah komputer yang dapat dihubungkan dalam jaringan akan lebih banyak dan jarak antara komputer dapat diperbesar. Pada Thicknet digunakan transceiver untuk menghubungkan setiap komputer dengan sistem jaringan dan konektor yang digunakan adalah konektor tipe DIX. Panjang kabel transceiver maksimum 50 m, panjang kabel Thick Ethernet maksimum 500 m dengan maksimum 100 transceiver terhubung.



Gambar 2.1 Thicknet core dan Thinnet core



Gambar 2.2 Kabel UTP dan STP



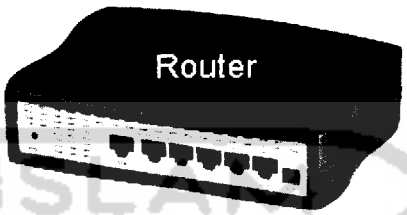
Gambar 2.3 Konektor RJ-45

2.3.2. *Network Card*

Network card berfungsi sebagai penghubung antara komputer dengan kabel jaringan agar sinyalnya dapat diolah oleh komputer. Kecepatan standart saat ini adalah 100 Mbps. Serta ada beberapa tipe yaitu ISA dan PCI

2.3.4. Router

Router merupakan piranti pendukung yang dapat diprogram untuk menghasilkan jaringan komputer berkualitas tinggi yang sangat mahal. Router bekerja dengan cara yang mirip dengan switch dan bridge. Perbedaanya router menyaring (filter) lalu lintas data.



Gambar 2.6 Router

2.4 TCP/IP

TCP/IP *(Transmission Control Protocol/Internet Protocol)* adalah nama protocol jaringan, Protocol ini sering digunakan sebagai protocol jaringan yang universal terutama pada internet, dan mempunyai kemampuan jaringan yang sangat baik, tetapi protocol ini membutuhkan konfigurasi yang cukup sulit. Pengalamatan jaringan tidak seperti physical addresses yang melebur ke dalam hardware manapun.

Pengalamatan jaringan diberikan oleh administrator jaringan dan secara logika dikonfigurasikan kedalam device jaringan. Karena penting peranannya pada sistem operasi Windows dan juga karena protokol TCP/IP merupakan protokol pilihan (default) dari Windows. Protokol TCP berada pada lapisan Transport model OSI (Open System Interconnection), sedangkan IP berada pada lapisan Network mode OSI

IP address adalah alamat yang diberikan pada jaringan komputer dan peralatan jaringan yang menggunakan protokol TCP/IP. *IP address* terdiri atas 32 bit angka biner yang dapat dituliskan sebagai empat kelompok angka desimal yang dipisahkan oleh tanda titik seperti 192.168.0.1.

Tabel 2.2. Contoh *IP address*

Network ID			Host ID
192	168	0	1

IP address terdiri atas dua bagian yaitu *network ID* dan *host ID*, dimana *network ID* menentukan alamat jaringan komputer, sedangkan *host ID* menentukan alamat host (komputer, router, *switch*). Oleh sebab itu *IP address* memberikan alamat lengkap suatu host beserta alamat jaringan di mana host itu berada.

Kelas-kelas *IP Address*

Untuk mempermudah pemakaian, bergantung pada kebutuhan pemakai, *IP address* dibagi dalam tiga kelas seperti diperlihatkan pada tabel 2.3.

Table 2.3 Kelas *IP Address*

Kelas	**Network ID**	**Host ID**	**Default Subnet Mask**
A	xxx.0.0.1	xxx.255.255.254	255.0.0.0
B	xxx.xxx.0.1	xxx.xxx.255.254	255.255.0.0
C	xxx.xxx.xxx.1	xxx.xxx.xxx.254	255.255.255.0

IP address kelas A diberikan untuk jaringan dengan jumlah host yang sangat besar. Range IP 1.xxx.xxx.xxx. – 126.xxx.xxx.xxx, terdapat 16.777.214 (16 juta) *IP address* pada tiap kelas A. *IP address* kelas A diberikan untuk jaringan dengan jumlah host yang sangat besar. Pada *IP address* kelas A, *network ID* ialah 8 bit pertama, sedangkan *host ID* ialah 24 bit berikutnya. Dengan demikian, cara membaca *IP address* kelas A, misalnya 113.46.5.6 ialah: Network ID = 113

Host ID = 46.5.6 Sehingga *IP address* diatas berarti *host* nomor 46.5.6 pada *network* nomor 113.

IP address kelas B biasanya dialokasikan untuk jaringan berukuran sedang dan besar. Pada *IP address* kelas B, network ID ialah 16 bit pertama, sedangkan *host ID* ialah 16 bit berikutnya. Dengan demikian, cara membaca *IP address* kelas B, misalnya 132.92.121.1 *Network ID* = 132.92

Host ID = 121.1 Sehingga *IP address* di atas berarti host nomor 121.1 pada *network* nomor 132.92. dengan panjang *host ID* 16 bit, *network* dengan *IP address* kelas B dapat menampung sekitar 65000 host. *Range* IP 128.0.xxx.xxx – 191.155.xxx.xxx

IP address kelas C awalnya digunakan untuk jaringan berukuran kecil (LAN). *Host ID* ialah 8 bit terakhir. Dengan konfigurasi ini, bisa dibentuk sekitar 2 juta network dengan masing-masing network memiliki 256 IP address. *Range IP* 192.0.0.xxx – 223.255.255.x. Pengalokasian *IP address* pada dasarnya ialah proses memilih *network Id* dan *host ID* yang tepat untuk suatu jaringan. Tepat atau tidaknya konfigurasi ini tergantung dari tujuan yang hendak dicapai, yaitu mengalokasikan *IP address* seefisien mungkin.

2.5 Microsoft Visual Basic 6.0

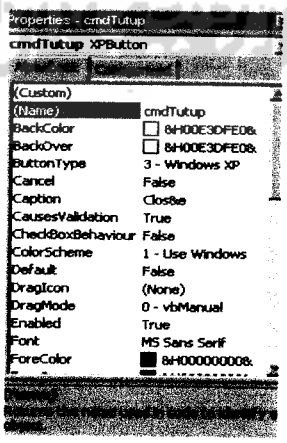
Microsoft Visual Basic 6.0 merupakan *software* penghasil aplikasi yang bekerja dalam *platform* Windows. Bidang kerja dalam pembuatan aplikasi adalah *form*. Komponen-komponen penyusun aplikasi, antara lain tombol perintah (*command button*), kotak teks (*text box*) dan lain-lain, diletakkan di dalam *form*. Tahap pembuatan aplikasi meliputi :

a. Membuat antar muka

Tahap ini merupakan desain *layout* komponen-komponen di dalam *form*. Desain aplikasi yang menarik dan *user friendly* (kemudahan dalam penggunaan aplikasi bagi semua pengguna, bahkan pengguna yang belum terbiasa dengan komputer sekalipun) dapat menjadi dasar baik atau buruknya aplikasi.

b. Mengatur properti

Properti merupakan sifat atau tampilan yang dimiliki komponen. Pengaturan properti komponen dapat melalui jendela *properties* sebelum program dijalankan atau di *run*, dapat juga melalui kode program. Komponen yang akan diatur propertinya kita pilih terlebih dahulu dengan *mouse* hingga terlihat ada 8 kotak kecil yang mengelilinginya.



Gambar 2.7 Jendela *Properties*

BAB III
PERANCANGAN

3.1. Bahan dalam Sistem Jaringan

Kebutuhan yang diperlukan untuk rencana membangun instalasi jaringan meliputi

3.1.1. Penempatan Server

Pada komputer server bisa disiapkan antara lain :

a. 1 unit komputer *workstation* untuk mengontrol aktivitas jaringan yang sekaligus sebagai komputer server.
b. Hub
c. Modem
d. Line telepon jika diperlukan

3.1.2. Kriteria Komputer Server

Komputer server merupakan komponen inti dalam membangun sebuah jaringan, ada beberapa kriteria dalam memilih komputer yang akan dijadikan server antara lain :

a. Casing diusahakan yang berbentuk tower
b. Motherboard harus memenuhi syarat untuk server
c. Memory minimal 512 Mbyte
d. Prosesor minimal Pentium III
e. Harddisk minimal 40 Gbyte, disarankan menggunakan SCSI
f. Kartu grafiknya minimal 16 Mbyte
g. Kartu jaringan minimal 100 Mbps

h. Modem minimal 56 Kbps

i. Peralatan standar seperti *keyboard, mouse, monitor, disk drive*, dan *CD-Rom*

3.1.3. Kriteria Komputer Client

Kriteria komputer yang akan digunakan sebagai client adalah sebagai berikut :

a. Casing diusahakan yang berbentuk tower

b. Memori minimal 128 Mbyte

c. Kartu grafik dengan memori minimal 32 Mbyte

d. Kartu jaringan 10/100 Mbps

e. Prosesor AMD K6-2 400 Mhz

f. Harddisk minimal 10 Gbyte

g. Peralatan standar seperti *keyboard, mouse, monitor, disk drive dan CD-Rom.*

3.2. Alat yang dibutuhkan

Ketika membangun suatu jaringan, peralatan yang dibutuhkan adalah sebagai berikut :

a. Obeng plus dan obeng minus

b. Obeng plus bermagnet

c. Test pen

d. Tang pemotong

e. Pinset

f. Tang penjepit

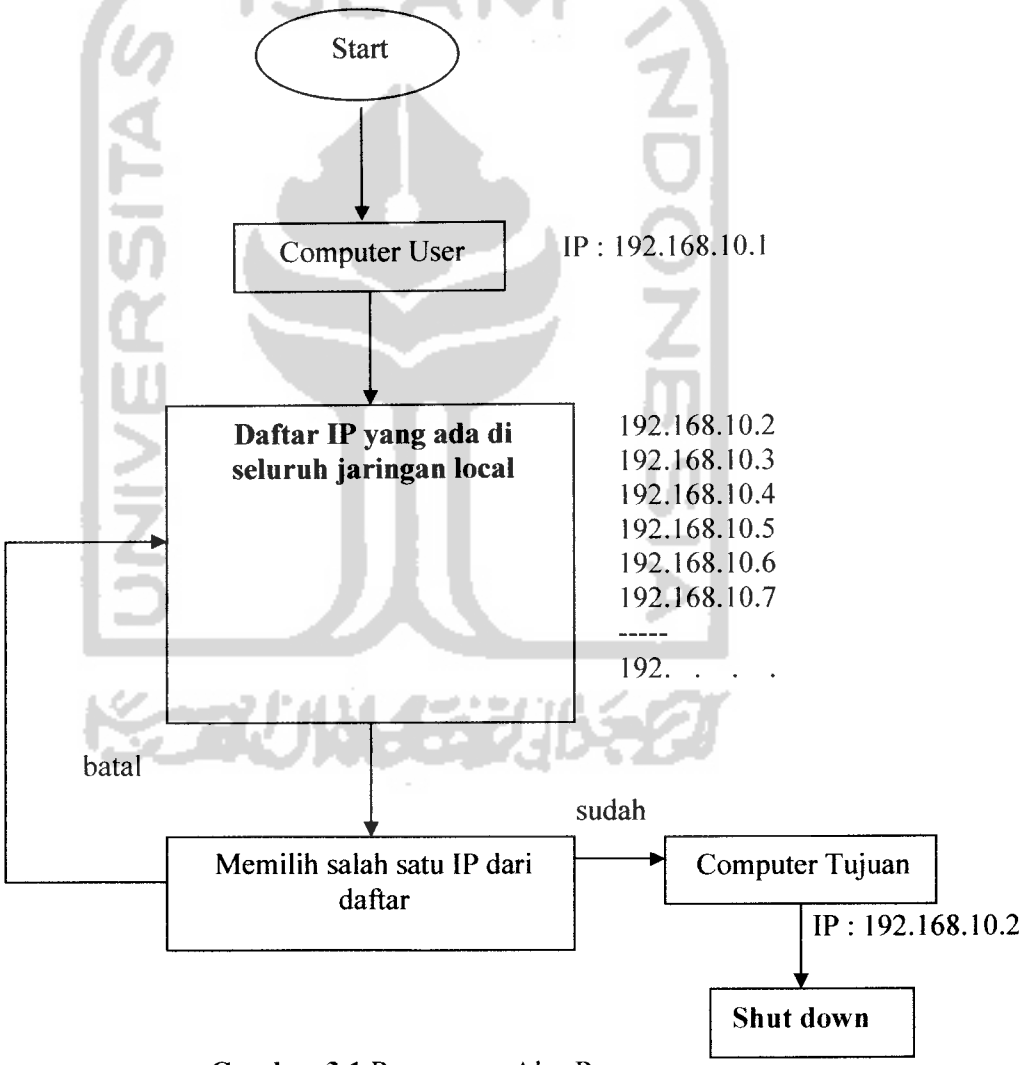
g. Avometer (multitester)

h. Clipper (tang crimping) untuk memasang konektor RJ45 (kabel UTP)

i. Alat tester untuk mengetahui konektisitas kabel jaringan UTP

3.3. Diagram Rancangan

Pada penelitian ini direncanakan akan menggunakan Microsoft Visual Basic 6.0 sebagai software pada *client-server*, dengan rencana program sebagai berikut :



Gambar 3.1 Rancangan Alur Program

Diskripsi Alur Program :

a. Pada rancangan program Microsoft Visual Basic 6.0 dapat digunakan apabila komputer server dan komputer client pada posisi hidup semua.
b. Komputer server dibuat kode IP : 192.168.10.1, sedangkan komputer client dibuat kode IP : 192.168.10.2 dan 192.168.10.3 dan seterusnya apabila akan digunakan komputer client lebih dari 2 unit.
c. Pada layar monitor akan selalu ditampilkan menu program shut down komputer client.
d. Apabila menu program shut down diaktifkan, maka pada layer monitor komputer server akan tampil daftar IP yang ada di seluruh jaringan local.
e. Pada layer monitor tersebut akan ditampilkan kotak dialog untuk memilih salah satu IP dari daftar keseluruhan
f. Apabila batal atau tidak jadi memilih kode IP komputer client, maka tampilan akan kembali pada daftar IP yang ada di seluruh jaringan local.
g. Tetapi jika pada kotak dialog dipilih "Ya", maka layar komputer akan tampil untuk memilih kode IP komputer client yang mau di shut down.
h. Setelah diisi kode IP komputer client, maka tinggal mengeklik kotak pilihan shut down. Pada komputer client yang dipilih akan terjadi shut down secara otomatis.

3.4 Kontrol dan Monitoring

Untuk mengontrol dan memonitor dari hasil rancangan penelitian ini maka dapat dijelaskan sesuai gambar diagram di bawah :

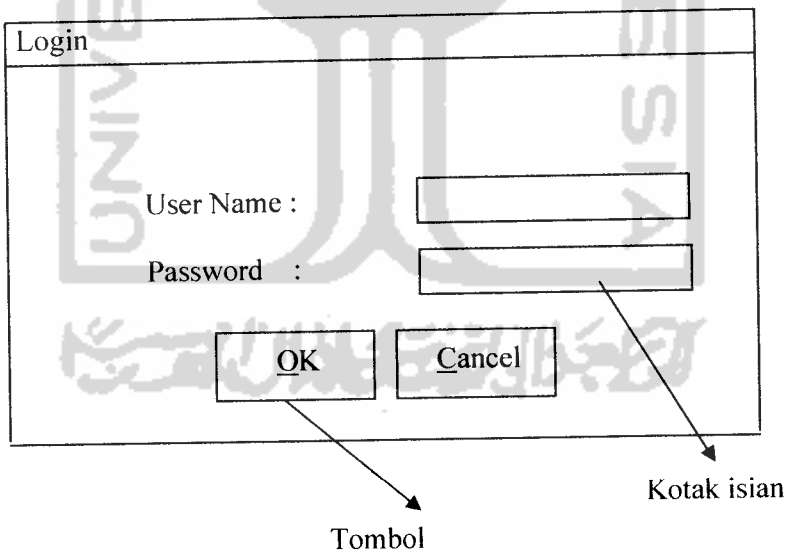
Keadaan seperti ini telah dinyatakan pada gambar diagram di atas dengan garis lurus.

e. Untuk proses kerja selanjutnya telah dijelaskan pada alur rancangan program.

Pada dasarnya komputer 1 sebagai komputer server berfungsi untuk mengontrol dan memonitor keadaan komputer 2 dan komputer 3 sebagai komputer client yang dijadikan obyek pada penelitian ini.

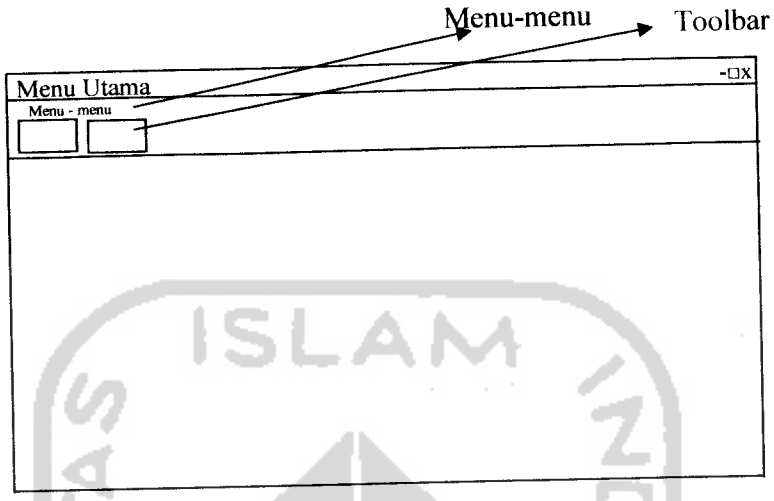
3.5 Alur Program

Pertama kali akan ditampilkan form login sebagai keamanan program, agar hak akses pada program ini hanya ada pada administrator.



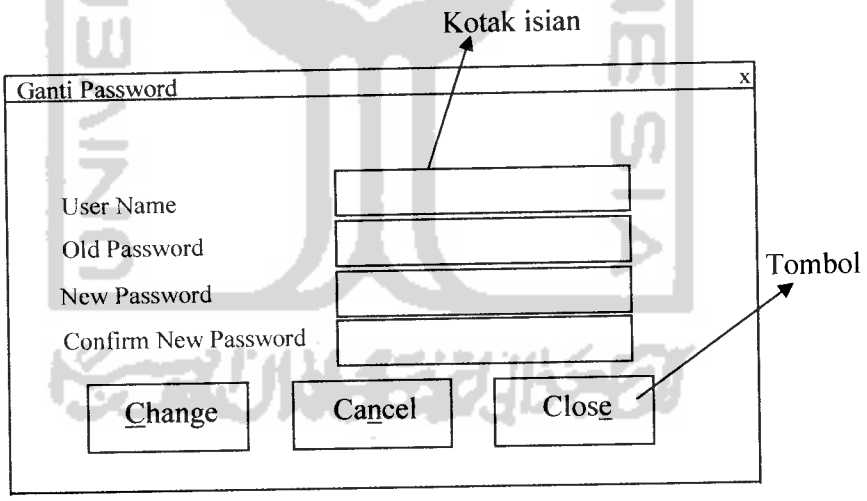
Gambar 3.3 Rancangan *Form Login*

Setelah *login* akan masuk pada menu utama, yang mempunyai tampilan beberapa menu toolbar seperti : *Password, Scan, Fun, About, Exit*



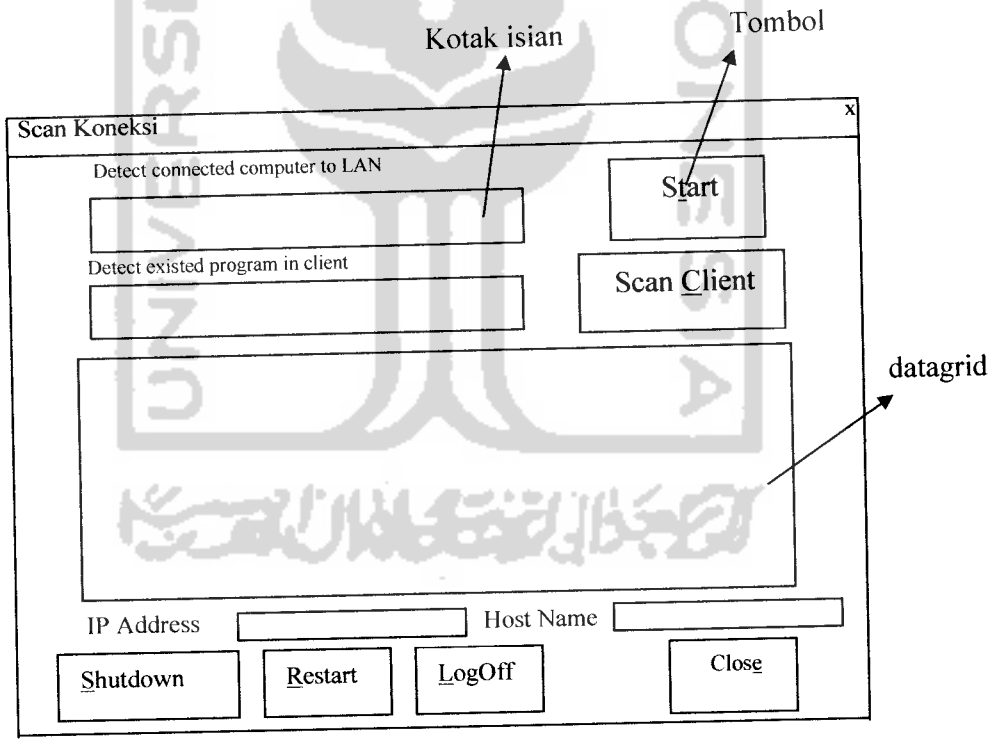
Gambar 3.4 Rancangan *Form Menu Utama*

Pada menu Toolbar password berfungsi untuk merubah user name dan password



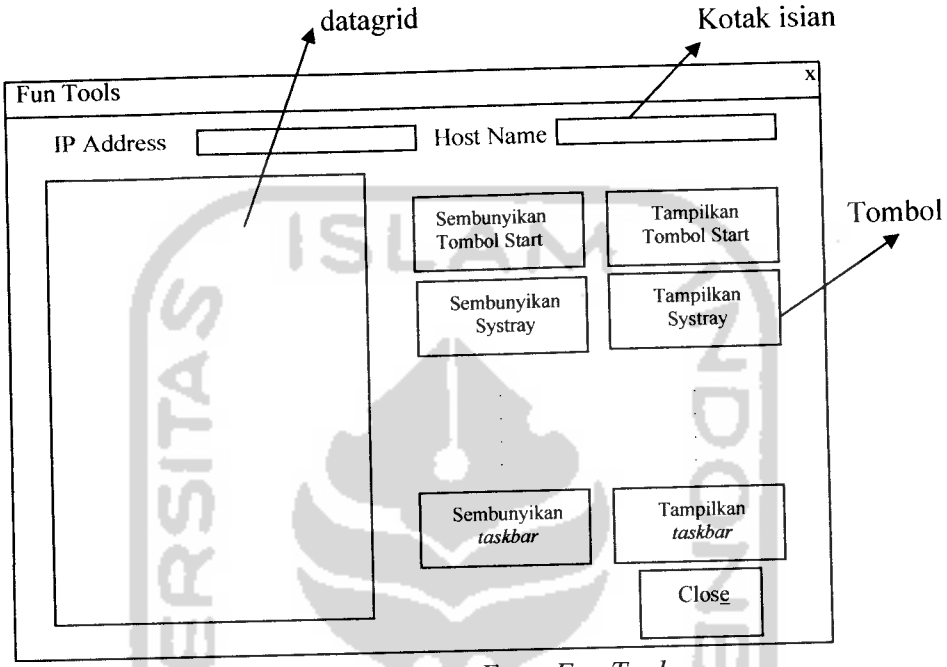
Gambar 3.5 Rancangan *Form ganti password*

Kemudian pada menu Tollbar Scan dan Fun, Tampilan Scan komputer merupakan form untuk mendaftar komputer klien yang akan dimonitor dan dikontrol. Di dalam aplikasi ini, daftar komputer yang terkoneksi jaringan akan muncul di kolom Alamat IP sebagai alamat komputer disertai dengan kolom nama komputernya dan dilengkapi dengan kolom status koneksi dari komputer klien cara kerja pada from Scan adalah klik tombol *start* untuk mengetahui alamat IP yang sedang terhubung ke server dan selanjutnya klik tombol *scan* untuk mengetahui apakah personal Computer sedang aktif atau tidak. Jika pada kolom tampilan telah bermunculan alamat IP klik salah satu alamat IP kemudian lakukan perintah *shutdown, restart atau logoff* tergantung keinginan admin.



Gambar 3.6 Rancangan *Form Scan Koneksi*

Sedangkan pada menu Fun terdiri dari beberapa perintah untuk melakukan atau bermain – main pada tampilan desktop seperti menghilangkan dan menampilkan kembali seperti *tombol start, IE Bar , system systray dan taskbar.*



Gambar 3.7 Rancangan *Form Fun Tools*

Sebagai jembatan interkoneksi digunakan pula sebuah program yang diberi nama "target" tujuannya agar dapat menghubungkan *IP address computer client* pada server sehingga program kontrol shut down dapat dilakukan.



Target.exe

Gambar 3.8 *Icon target*

BAB IV

PENGAMATAN DAN ANALISA

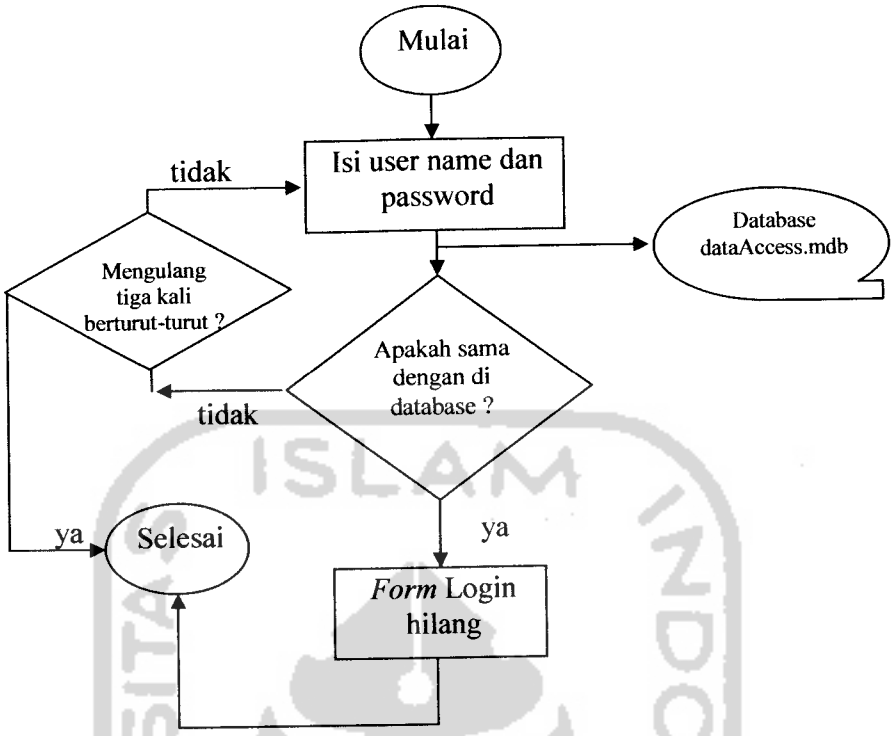
4.1 Pengamatan

4.1.1 *Form* Login



Gambar 4.1 *Form* Login

Program apabila dijalankan, akan muncul dua *form* bersamaan. Kedua *form* itu adalah *form* menu Utama dan *form* Login, dengan sifat kemunculan *form* Login ini adalah *vbmodal.* Yang artinya pemakai tidak dapat menjalankan perintah pada *form* lain, sebelum *form* yang bersifat *vbmodal* tadi sudah dihilangkan atau ditutup. Adapun diagram alir dari *form* Login sebagai berikut :



Gambar 4.2 *Flowchart form* Login

4.1.2 *Form* Menu Utama

Pada *form* menu Utama terdapat fasilitas menu-menu dan *toolbar* yang dapat digunakan untuk menjalankan aplikasi yang ada di program ini. Rincian menu-menu yang ada di *form* Utama sebagai berikut :

a. Menu File mempunyai sub menu Password

Sub menu Password jika dijalankan akan menampilkan frmGanti Passwd. *Form* ini berguna untuk mengubah *user nam*e maupun *password* yang ada agar tetap terjaga kerahasiaannya.

b. Menu Registrastation mempunyai sub menu Scan

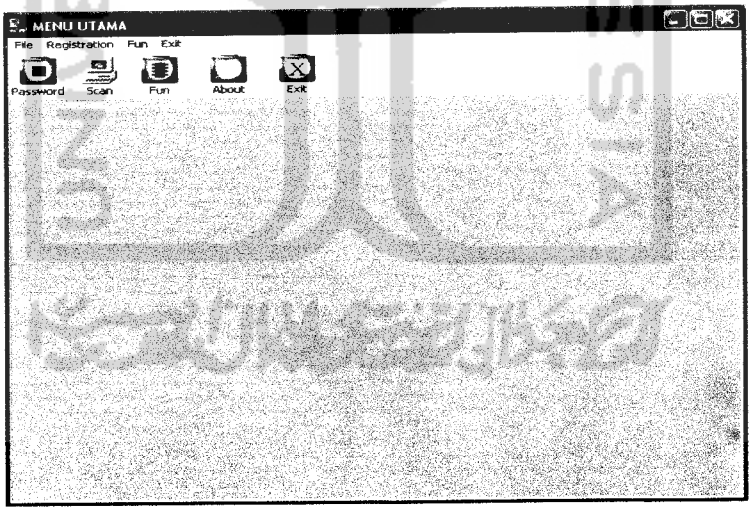
Sub menu Scan apabila dijalankan menampilkan frmScanClient. *Form* ini berguna untuk melakukan deteksi komputer klien yang telah terkoneksi dengan jaringan dan deteksi keberadaan program shutdown di komputer klien yang terkoneksi tadi.

c. Menu Fun

Menu ini berguna untuk menampilkan frmFun. Adapun *form* frmFun ini berisi aplikasi untuk mengubah tampilan windows yaitu menyembunyikan atau menampilkan *Taskbar, IEBar,* tombol *Start* dan *Systray.*

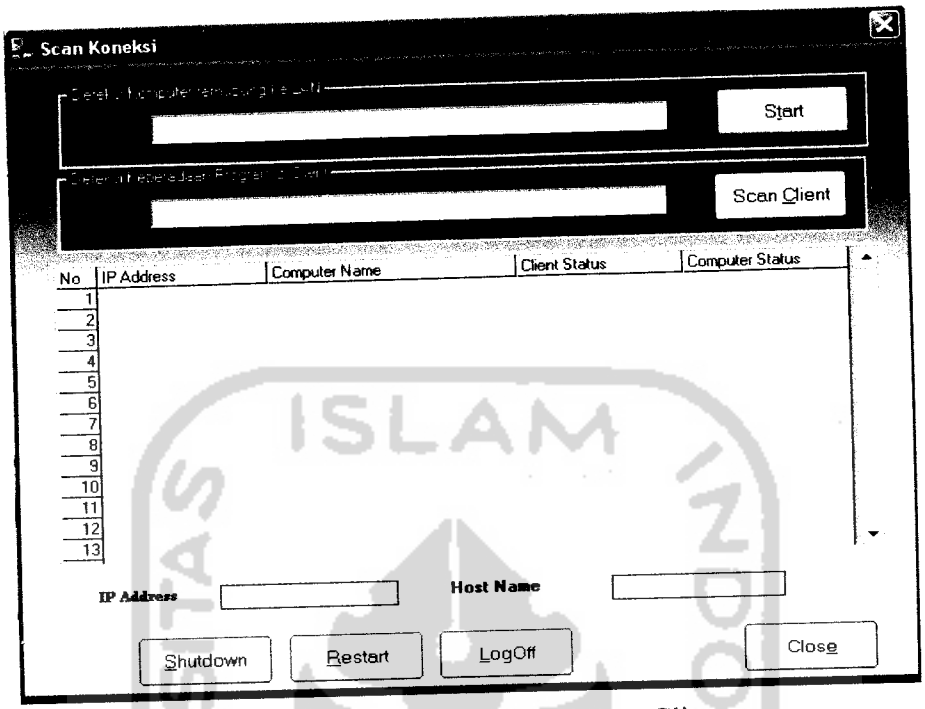
d. Menu Exit

Menu yang digunakan apabila akan mengakhiri aplikasi ini. Sebelum aplikasi hilang dari layar, akan muncul kotak pesan untuk menanyakan kepada pengguna apakah benar akan keluar dari aplikasi ini.



Gambar.4.3 *Form* Menu Utama

4.1.3 *Form* ScanClient



Gambar 4.5 *Form* Scan Client

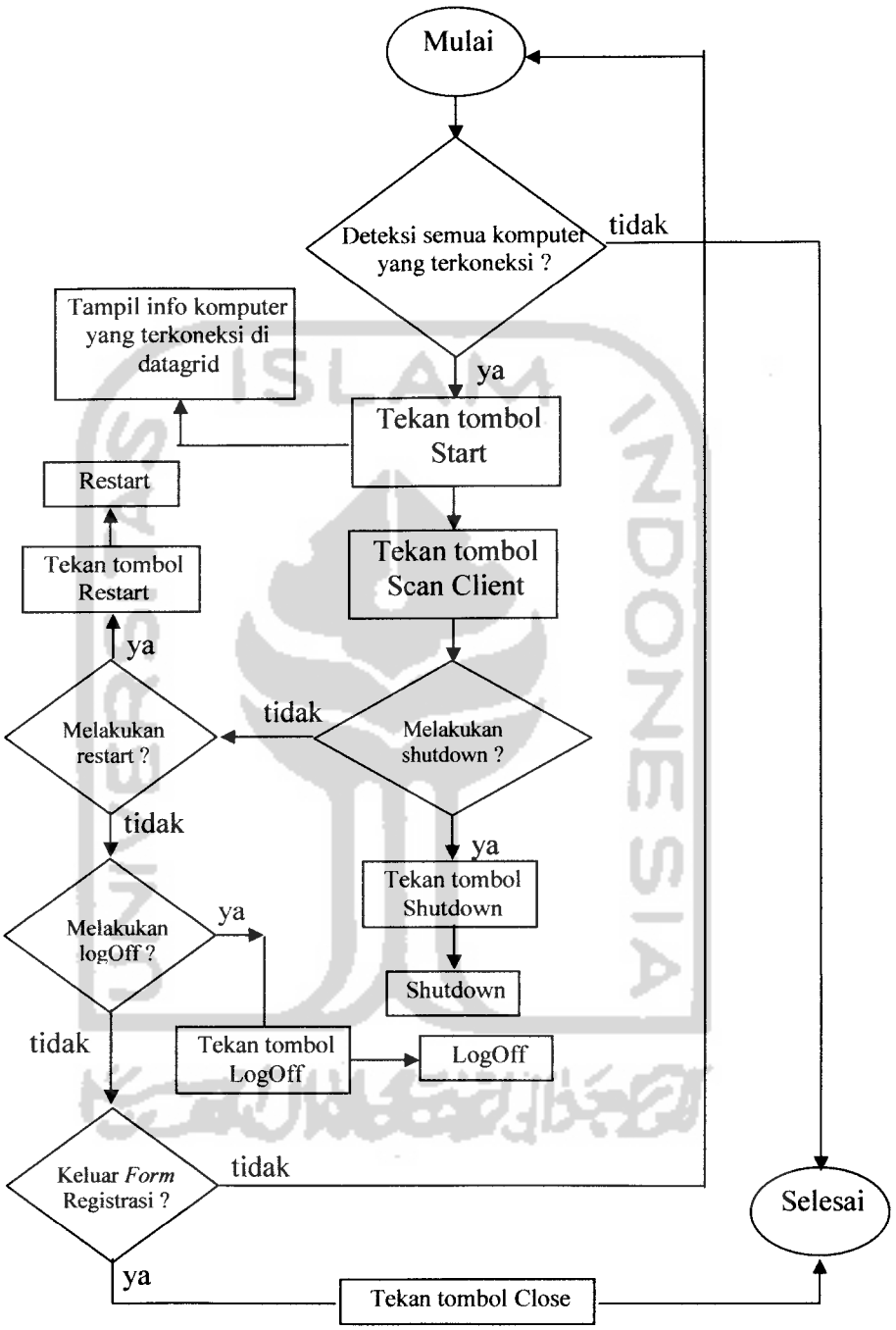
Form ini digunakan untuk mendeteksi semua komputer *client* yang terhubung dengan komputer *server*. Dalam aplikasi ini jumlah maksimal komputer yang dapat terhubung sebesar 255 buah. Hal ini dapat terlihat pada kotak input IP address yang menampilkan alamat komputer *client* dari 127.0.0.1 sampai dengan 127.0.0.255. Komputer yang terkoneksi dengan jaringan akan ditampilkan dalam tabel dengan informasi: no urut, IP address, Computer name, Status client dan Computer status.

Pada Gambar 4.7 memperlihatkan, keadaan sebelum tombol ScanClient ditekan. Hampir sama dengan kejadian setelah tombol Start dijalankan, tetapi ada perbedaan pada tabel kolom ke-3 dan ke-4. Pada kolom tersebut, mula-mula bewarna biru dan tidak ada keterangan di dalamnya (lihat Gambar 4.7). Kejadian setelah tombol ini dijalankan, kolom-kolom tadi berubah bewarna hijau muda dan terdapat keterangan pada kolom ke-3 menjadi Client Installed dan kolom ke-4 menjadi RUNNING (lihat Gambar 4.8). Keterangan ini mempunyai arti bahwa komputer-komputer yang terdapat pada tabel telah terkoneksi dan dalam status siap dikendalikan.

Urutan penggunaan harus melalui tahap-tahap tadi, yaitu menjalankan tombol Start, kemudian mengecek status komputer dengan menekan tombol ScanClient. Penggunaan program pada *form* ini tidak dapat diubah urutannya. Program akan menampilkan kotak pesan, apabila pengguna tidak sesuai urutan penggunaannya. Selengkapnya, perhatikan diagram alir pada Gambar 4.9.

Untuk selanjutnya, pengguna dapat melakukan perintah-perintah pada salah satu komputer yang terdapat pada tabel dengan memilih tombol perintah yang mempunyai fungsi sesuai dengan *caption* nya.

Diagram alir pada *form* ini sebagai berikut :



Gambar 4.9 *Flowchart form* FrmRegistrasi

4.1.6 Program koneksi "target"

Pada penelitian ini, disamping program utama yang berguna untuk mengendalikan komputer klien baik *shutdown,logoff* dan *restart*, masih ada program bantu yang berguna untuk memasang "target" pada komputer klien. Maksud target adalah suatu file berekstensi *.dll, yang diinstalkan pada komputer klien, sehingga komputer klien mempunyai file tadi di *path* sistem komputer tersebut, C:\windows\system32. Nama file pada penelitian ini adalah zlib.dll, dan dapat dinamakan sesuai keinginan pengguna. File zlib.dll secara otomatis akan terbentuk pada *path* C:\windows\system32\zlib.dll setiap file *executable* dari program bantu ini dijalankan, yang pada penelitian ini bernama target.exe.

4. 2 Analisa

4.2.1 Kejadian pada Tombol Start

Fungsi tombol ini untuk mendeteksi komputer yang terhubung dalam LAN atau tidak. Adapun prosedur yang penting adalah fScanLAN

```
Private Sub fScanLAN()
Dim i As Integer
Dim txtCurrentIPx As String
Call fDisableButton
indexGrid = 1
PB.Value = 0
PB.Max = maxScan
keluarScan = False
For i = 0 To maxScan
   txtCurrentIPx = txtIPs + CStr(i)
   txtCurrentIP.Text = txtCurrentIPx
   IPRespon = Ping_IP(txtCurrentIPx)
   If Not fCheckInGrid(IPRespon) Then
      txtCurrentHost.Text = ResolveHostx(txtCurrentIP.Text)
      Call fIsiGrid(indexGrid, IPRespon, txtCurrentHost.Text, "", "", vbBlue)
   End If
   PB.Value = i
   If keluarScan = True Then Exit For
   DoEvents
Next
cmdScanClient.Enabled = True
PB.Value = 0
End Sub
```

Gambar 4.10 Kode program prosedur fScanLan

Dalam prosedur ini, semua komputer dalam LAN dicek koneksinya dengan menjalankan fungsi Ping_IP. Fungsi ini berada dalam module ModPing. Jumlah maksimal komputer yang dicek status koneksinya adalah sebesar properti Max pada objek *progress bar*. Pada penelitian ini sebesar 255. Setiap komputer yang terdeteksi akan ditampilkan pada tabel, dengan keterangan IP address, Computer Name, Client Status dan Computer Status, yaitu dengan memanggil prosedur flsiGrid. Nilai IP Address antara 127.0.0.1 sampai dengan 127.0.0.255, hal ini dikarenakan variabel txtIPs mempunyai nilai keluaran dari WS.LocalIP (127.0.0.xxx). Variabel tadi diberi nilai pada prosedure Form_Load, seperti pada keterangan di bawah ini:

```
txtip = WS.LocalIP
txtIPx = Split(txtip, ".", , vbTextCompare)
txtIPs = txtIPx(0) + "." + txtIPx(1) + "." + txtIPx(2) + "."
```

Gambar 4.11 Variabel IP Address

4.2.2 Kejadian pada Tombol Scan Client

Fungsi tombol ini untuk mengetahui status komputer yang telah terdeteksi pada jaringan LAN. Fungsi yang penting pada tombol scanClient ini adalah fScanClient. Prosedur fScanClient, akan memeriksa semua komputer-komputer yang ada pada daftar tabel. Pengisian status komputer (*Client Installed* atau *Client Not Installed*), merupakan hasil prosedur retKon. Prosedur retKon mempunyai nilai kembalian *True* atau *False*, prosedur ini merupakan hasil dari fungsi ProsedurKoneksi2.(lihat Gambar 4.13). Apabila nilai kembalian prosedur

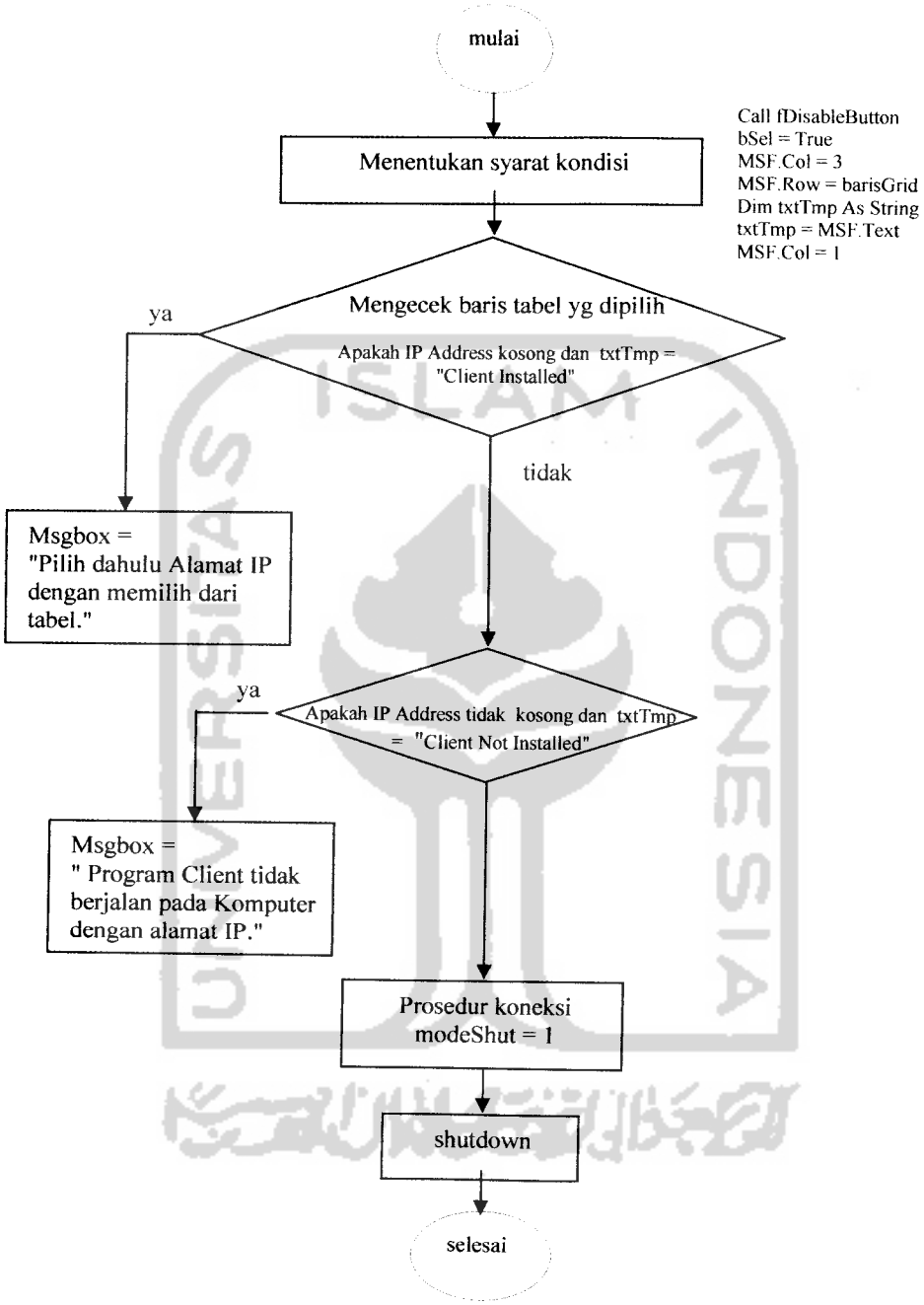
retKon adalah benar maka kolom Status Client akan tertulis *Client Installed* dan diblok warna hijau muda, sebaliknya apabila nilai kembalian prosedur retKon adalah salah maka kolom Status Client akan tertulis *Client Not Installed* dan diblok warna merah.

```
Private Sub fScanClient()
Dim i As Integer
Dim txtCurrentIPx As String
Dim txtCurrentHostx As String
Dim retKon As Boolean
Call fDisableButton

indexGrid = 1
PB2.Value = 0
PB2.Max = 50
keluarScan = False
With MSF
.Col = 1
For i = 1 To 50
   .Row = i
   txtCurrentIPx = .Text
   txtCurrentIP.Text = txtCurrentIPx
   .Col = 2
   txtCurrentHostx = .Text
   txtCurrentHost.Text = txtCurrentHostx
   .Col = 1
   If txtCurrentIP.Text <> "" Then "stl melakukan "perintah" textbox ga kosong
   retKon = ProsedurKoneksi2
   If retKon Then
      Call fIsiGrid(i, txtCurrentIP.Text, txtCurrentHost.Text, "Client Installed",
"RUNNING", vbGreen)
   Else
     Call fIsiGrid(i, txtCurrentIP.Text, txtCurrentHost.Text, "Client Not Installed", "",
vbRed)
   End If
   End If
   PB2.Value = i
   If keluarScan = True Then Exit For
   DoEvents
Next
End With
cmdScanClient.Enabled = True
PB.Value = 0
Call fEnableButton
```

Gambar 4.12 Kode program prosedur fScanClient

4.2.3 Proses shutdown dalam tombol Shutdown



Gambar 4.14 Diagram alir proses pada tombol Shutdown

digunakan sebagai syaraat kondisi. Kedua informasi ini sebagai syarat kondisi komputer client yang akan dilakukan kendali memakai program system shutdown ini. Pada msFlexGrid terlihat computer yang akan dikendalikan terblok warna merah dengan keterangan sesuai tombol kendali, dalam hal ini adalah shutdown. Untuk selanjutnya, prosedur ProsedurKoneksi dijalankan. Prosedur ini berfungsi untuk mendeteksi computer yang terhubung dengan jaringan LAN. Adapun kode program sebagai berikut:

```
Private Sub ProsedurKoneksi()
On Error GoTo err_Connect
   If txtCurrentIP.Text = "" Then
      MsgBox "Pilih dahulu Alamat IP dengan memilih dari tabel.", vbOKOnly +
vbExclamation, "Pesan Kesalahan"
      Call fEnableButton
   Else
      With Me
         If  WS.State > 0 Then  "'syarat close
            .WS.Close
         End If
         .WS.RemoteHost = txtCurrentIP.Text
         .WS.RemotePort = 1980
         .WS.Connect
      End With
   End If
   Exit Sub

err_Connect:
   MsgBox "Tidak berhasil melakukan koneksi ke Client", vbOKOnly + vbCritical,
"Pesan Kesalahan"
   Call fEnableButton
   Exit Sub
End Sub
```

Gambar 4.16 Kode program prosedur ProsedurKoneksi

Prosedur ini dapat berjalan dengan syarat kotak isian IP address dan host name komputer telah terisi. Jika kotak isian IP address dan host name komputer belum terisi, maka akan muncul pesan peringatan. Pengisian kotak isian tersebut

dimasukkan dan dikeluarkan media). Nilai port harus sama dengan properti RemotePort winshock, yaitu sebesar 1980.

(no. 7) menjalankan koneksi antar komputer pada LAN

Kejadian *shutdown* ditentukan dengan nilai variabel modeshut. Penentuan nilai variabel dan fungsi diperoleh dengan menjalankan prosedur WS_connect (lihat Gambar 4.18). Semua kejadian pada tombol-tombol perintah untuk *shutdown, restart* dan *logoff*, ada pada prosedur ini, dengan memperhatikan nilai modeshut yang diberikan. Kejadian *shutdown* apabila modeshut = 1, untuk *restart* modeshut = 2 dan *logoff* modeshut = 0.

```
Case 1:
   On Error GoTo WSError1
   If txtCurrentIP.Text <> "" Then
      WS.SendData "SHUTDOWN;" & "1"
   Else
      MsgBox "Pilih dahulu Alamat IP dengan memilih dari tabel.",
vbOKOnly + vbExclamation, "Pesan Kesalahan"
   End If
   Exit Sub
WSError1:
      MsgBox "Tidak berhasil mengirim perintah Shutdown", vbOKOnly +
vbCritical, "Pesan Kesalahan"
```

Gambar 4.18 proses *shutdown* pada WS_Connect

Modeshut merupakan variabel yang mengarahkan pada pilihan proses koneksi yang dilakukan oleh winshock. Pada Gambar 4.18, dengan variabel modeshut = 1, maka program menuju Case 1, pada bagian Select – Case prosedur WS_Connect. Adapun kode program yang menyebabkan komputer *shutdown* yaitu WS.SendData "SHUTDOWN;" & "1". Winshock mengirim perintah bernomor 1,

```
Case 1:
   On Error GoTo WSError1
   If txtCurrentIP.Text <> "" Then
      WS.SendData "SHUTDOWN;" & "1"
   Else
      MsgBox "Pilih dahulu Alamat IP dengan memilih dari tabel.", vbOKOnly +
vbExclamation, "Pesan Kesalahan"
   End If
   Exit Sub
WSError1:
      MsgBox "Tidak berhasil mengirim perintah Shutdown", vbOKOnly + vbCritical, "Pesan
Kesalahan"
Case 2:
   On Error GoTo WSError2
   If txtCurrentIP.Text <> "" Then
      WS.SendData "SHUTDOWN;" & "2"
   Else
      MsgBox "Pilih dahulu Alamat IP dengan memilih dari tabel.", vbOKOnly +
vbExclamation, "Pesan Kesalahan"
   End If
   Exit Sub
WSError2:
      MsgBox "Tidak berhasil mengirim perintah Restart", vbOKOnly + vbCritical, "Pesan
Kesalahan"
Case 3:
   On Error GoTo WSError3
   If txtCurrentIP.Text <> "" Then
      WS.SendData "SHUTDOWN;" & "0" "3"

   Else
      MsgBox "Pilih dahulu Alamat IP dengan memilih dari tabel.", vbOKOnly +
vbExclamation, "Pesan Kesalahan"
   End If
   Exit Sub
WSError3:
     MsgBox "Tidak berhasil mengirim perintah LogOff", vbOKOnly + vbCritical, "Pesan
Kesalahan"
End Select
End Sub
```

Gambar 4.20 Prosedur WS_Connect

BAB V

KESIMPULAN dan SARAN

5.1 Kesimpulan

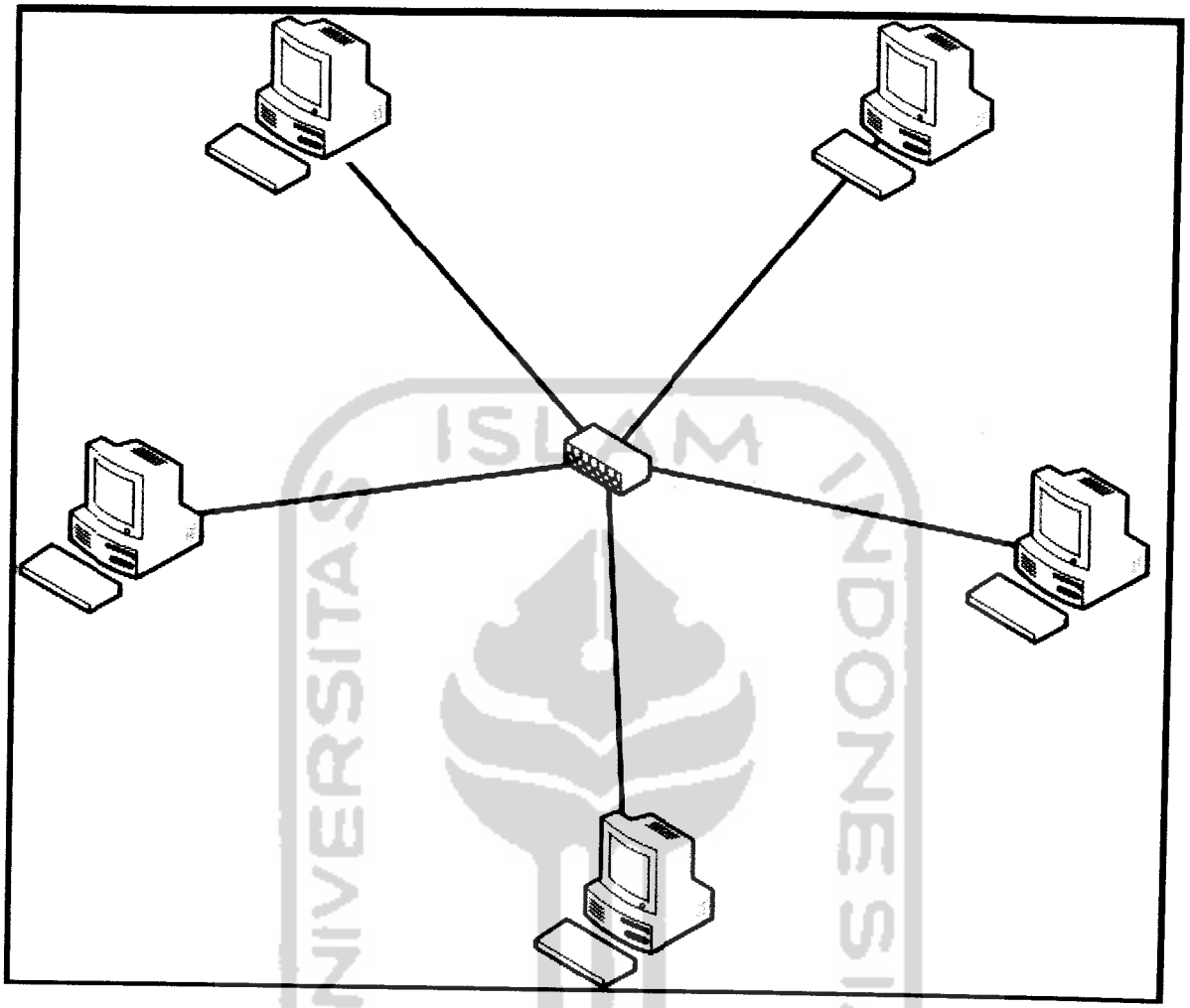
1. Komputer yang dikendalikan adalah komputer dalam jaringan yang hanya diinstal file target.exe, sedangkan komputer pengendali adalah komputer dalam jaringan yang diinstal program aplikasi ini.
2. Hasil Penelitian yang telah dilakukan :
    a) Komputer pengendali dapat melakukan pendeteksian (*scanning*) keberadaan komputer lain dalam jaringan.
    b) Komputer pengendali dapat melakukan pengendalian pada komputer lain dalam jaringan, baik untuk *shutdown*, *restart* dan *logoff*.

5.2 Saran

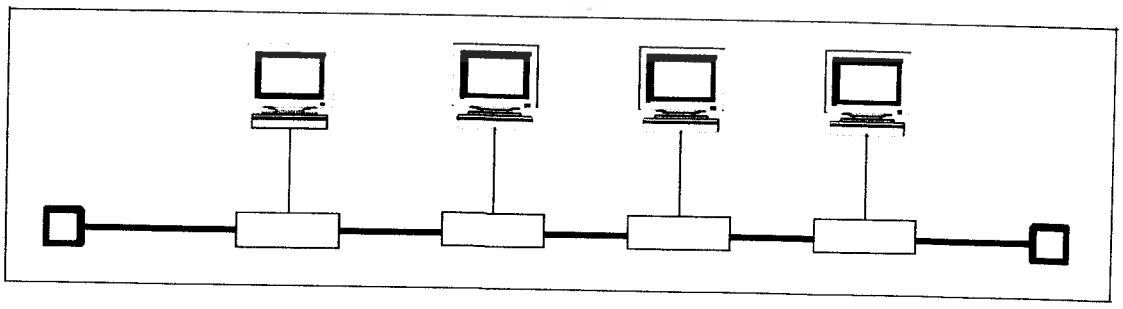
1. Perlu dilakukan penelitian yang dapat menghasilkan program aplikasi yang dapat digunakan dalam sistem operasi selain Windows.
2. Perlu dilengkapi fasilitas pengendalian jaringan, terutama untuk komunikasi data.

Lampiran 1

TOPOLOGI JARINGAN STAR

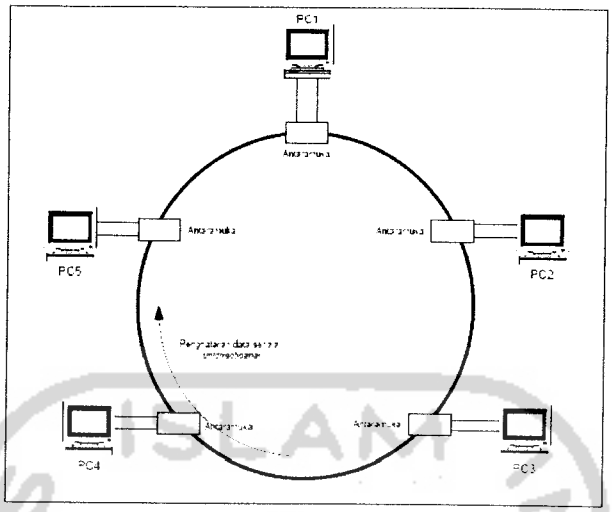


TOPOLOGI JARINGAN BUS

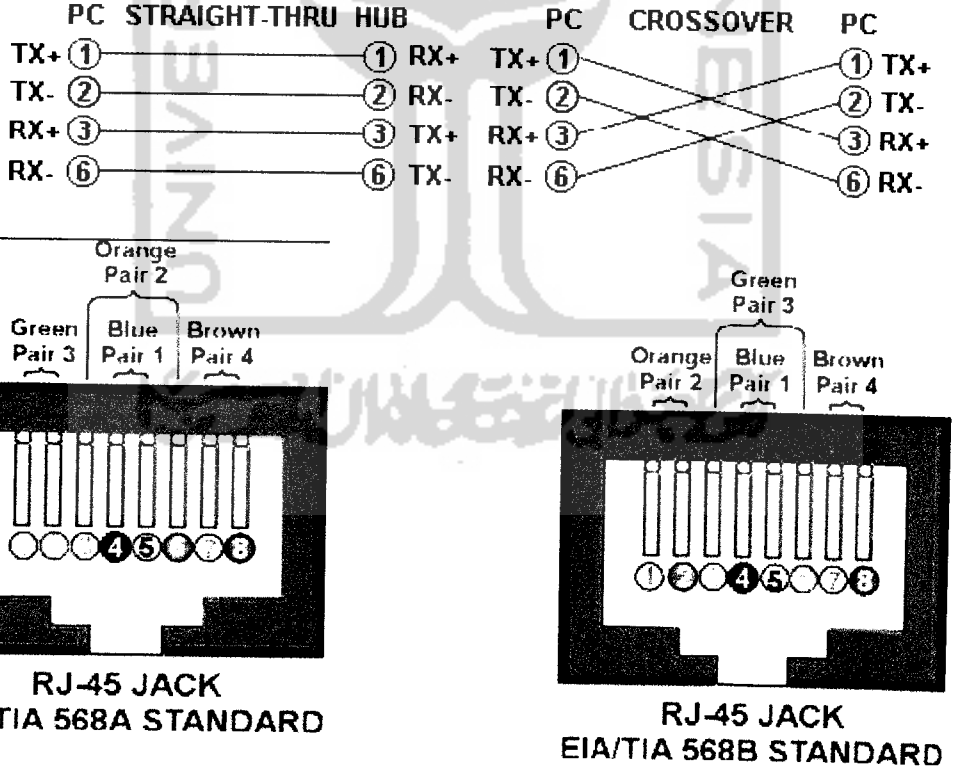


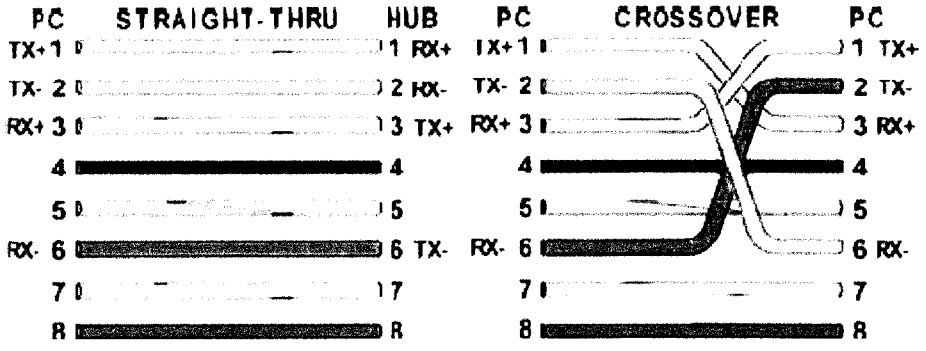
TOPOLOGI JARINGAN RING

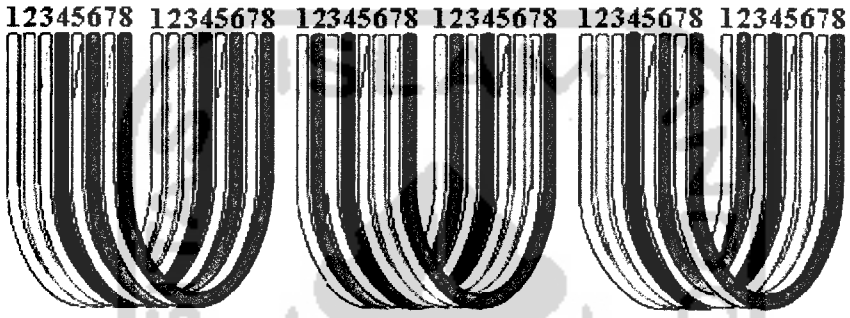
Topologi ring manghubungkan semua nod menggunakan medium penghatar seperti dibawah:



Crossed kabel Ke Hub dan PC to PC







STRAIGHT-THRU 568A | STRAIGHT-THRU 568B | CROSSOVER

Tampak Kabel yang sudah jadi di Konektor RJ-45



Tipe Standarisasi Kabel

Tipe Kabel	Keterangan
10Base2	10 Mbps baseband Ethernet dari IEEE 802.3, menggunakan kabel thin coaxial, jarak maksimal 606,8 feet – 185 meter per segmen. Mampu menghubungkan 5 segmen sehingga panjang keseluruhan mencapai 925 m. Sebuah segmen hanya mampu menampung max. 30 komputer saja.
10Base5	10 Mbps baseband 500 m, bekerja dilapisan physical, menggunakan kbel thinck coaxial 50 ohm berdiameter 0,5 inch (10 mm), jarak maksimal 1640 feet – 500 meter per segmen. Jika dipasang penghubung(repeater), sebuah jaringan bias mencapai maksimun 2,5 km.
10BaseF	Merujuk ke 10BaseFB, 10BaseFL, dan 10BaseFP, yang merupakan standar untuk kabel fiber optic. 10BaseF merupakan standar IEEE 802.3.Bentuk jaringan 10BaseF sama dengan 10BaseT, yakni berbentuk star. Karena menggunakan serat optic (fiberoptic) untuk media transmisinya, maka panjang antar NIC dan kosentratornya menjadi lebih panjang sampai 20 x (2000 m).Demikian pula dengan panjang total jaringannya. Pada 10BaseF, untuk transmisi output (TX) dan input (RX) menggunakan kabel/media yang berbeda.
10BaseFB	Tidak digunakan untuk koneksi antarworkstation (client), melainkan untuk jalur backbone, penambahan segmen dan repeater yang berhubungan ke jaringan.
10BaseFL	Dirancang untuk mengganti spesifikasi FOIRL, namun tetap mampu beroperasi dilingkungan FOIRL. Jangkauan segmen 10BaseFL dapat ditingkatkan hingga mencapai 3280 feet- 1000m jika digunakan dengan FOIRL. Jarak dapat ditingkatkan hingga 1.24 mil (2000m) jika digunakan secara eksklusif.
10BaseFP	10 Mbps fiber passive baseband Ethernet, menggunakan kabel fiber optic. 10BaseFL mengatur penomoran computer di dan kedalam topologi bintang tanpa meggunakan repeater. Jangkauan segmen 10BaseFL dapat ditingkatkan hingga 1640 feat – 500 meter.
10BaseT	Menggunakan kabel UTP category 3, 4 atau 5. satu pasang kabel digunakan untuk transmit data, satu pasang lainnya untuk receive data. Jangkauan maksimal 328 feet -100 meter per segmen. Menggunakan hub/switch sebagai pengganti konsentrator dan repeater pada topologi

	star. Setiap hub bisa dihubungkan untuk memperpanjang jaringan sampai 4 unit sehingga maksimal computer tersambung bisa mencapai 1024 unit
100BaseT	Merupakan standar IEEE 802.3. 100 Mbps basband Fast Ethernet, menggunakan kabel UTP seperti 10BaseT.100BaseT mengirimkan link pulse kesegmen jaringan lain ketika tidak ada traffic. Seri 100Base mempunyai beragam jenis berdasarkan metode akses datanya, diantaranya adalah 100BaseT,100Base-TX, dan 100Base-FX. Kecepatan transmisi seri 100Base bisa melebihi kecepatan chip pendahulunya(seri10Base) antara2-20 kali (20-200 Mbps). Ini dibuat untuk menyaingi jenis LAN berkecepatan tinggi lainnya seperti FDDI, 100VG-AnyLAN, dan lain sebagainya.
100BaseTX	100 Mbps baseband Fast Ethernet, menggunakan kabel UTP atau STP. Satu pasang kabel digunakan untuk transmit data, satu pasang lainnya untuk receive data.Bergaransi sesuai time signal.Sebuah segmen 100BaseTX jangkauannya melebihi jarak 328feet – 100 meter per segmen.
100BaseFX	Menggunakan 2 untai multimode kabel fiber optic. Seperti 100BaseFX jangkauannya bisa melebihi 1312 feet – 400 meter persegmen.
100BaseX	Standar untuk Fast Ethernet kabel Fiber Optic. Seperti 100BaseTX dan 100BaseFX. Berbasis standar IEEE 802.3

LAMPIRAN 2

Kode program

File Program Utama SHUTDOWN

FORM MENUUTAMA

```
Private Sub MDIForm_Load()
On Error GoTo errBukaDatabase
   koneksi
   On Error GoTo filetdkKetemu
 If Not GetProcess("targetUII") Then Shell App.Path + "\targetUII.exe"
   Exit Sub
errBukaDatabase:
MsgBox "Failed to open database!" + vbCrLf + "Make sure file dataAccess.mdb is on your directory
program.", vbOKOnly + vbCritical, "Error Message"
End
filetdkKetemu:
MsgBox "File Target.exe is not found on path program.", vbOKOnly, "Error Message"
End Sub
```

```
Private Sub mnuExit_Click()
Dim pesan As String
pesan = MsgBox("Are you sure want to close this program?", vbQuestion + vbYesNo, "Close")
If pesan = vbYes Then
End
Else
Exit Sub
End
End If
End Sub
```

```
Private Sub mnuFun_Click()
If bTelahScan Then
 Call frmFUN.fCopyGrid
 frmFUN.Show vbModal
 Else
MsgBox "Scan first  Client computer!", vbOKOnly + vbExclamation, "Error Message"
End If
End Sub
```

```
Private Sub mnuPassword_Click()
frmGantiPasswd.Show vbModal
End Sub
```

```
Private Sub mnuScan_Click()
Me.Hide
frmRegistrasi.Show vbModal
End Sub
```

```
Private Sub Toolbar1_ButtonClick(ByVal Button As MSComctlLib.Button)
Select Case Button.Caption
Case "Scan":
Me.Hide
frmRegistrasi.Show vbModal
Case "Password": frmGantiPasswd.Show vbModal
Case "Fun":
        If bTelahScan Then
           Call frmFUN.fCopyGrid
           frmFUN.Show vbModal
        Else
           MsgBox "Scan dahulu komputer Client!", vbOKOnly + vbExclamation, "Pesan Kesalahan"
        End If
```

```
Case "About": frmAbout.Show vbModal

Case "Exit": mnuExit_Click
End Select
End Sub
```

FORM LOGIN

```
Dim i As Integer

Private Sub Form_Load()
i = 0
End Sub
Private Sub Form_Unload(Cancel As Integer)
Unload Me
Unload MENUUTAMA
End Sub
Private Sub txtPassword_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
   XPButton1.SetFocus
End If
End Sub
Private Sub txtUser_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then
   txtPassword.SetFocus
End If
End Sub
Private Sub XPButton1_Click()
On Error GoTo errhndlr
koneksi
StrSql = "select * from Pass" ' where katakunci='" & txtPassword & "'"
Set RS = Conn.Execute(StrSql)
If txtUser.Text = "" Then
   MsgBox "User name kosong.", vbOKOnly, "Pesan Kesalahan"
   txtUser.SetFocus
ElseIf (LCase(txtUser.Text) = LCase(RS!namauser)) And (LCase(txtPassword.Text) = LCase(RS!katakunci))
Then
   Me.Hide
Else

   i = i + 1 'increment
   If i = 3 Then
      MsgBox "Maaf, anda tidak berhak menggunakan program ini!", vbOKOnly + vbExclamation, "Login
gagal"
      Me.Hide
   End If
   MsgBox "Nama User atau Password anda salah!" + vbCrLf + "Anda masih punya " + CStr((3 - i)) + " kali
kesempatan memasukkan password."
   txtUser.SetFocus
End If
errhndlr:
   Exit Sub
End Sub
Private Sub XPButton2_Click()
End
End Sub
```

FORM FRMREGISTRASI

```
Dim barisGrid As Integer
```

```
Dim kolomGrid As Integer
Dim bSel As Boolean
Dim bCari As Boolean
Dim IPRespon As String
Dim indexGrid As Integer
Dim txtIPs As String
Dim keluarScan As Boolean
Dim bScanLAN As Boolean
Dim bScanClient As Boolean
Dim bRespon As Integer
Const maxScan = 255
```

Private Sub ProsedurKoneksi()

```
On Error GoTo err_Connect
   If txtCurrentIP.Text = "" Then
      MsgBox "Select first IP Address by choosing from tables.", vbOKOnly + vbExclamation, "Error
Message"
      Call fEnableButton
   Else
      With Me
         If WS.State > 0 Then  '"syarat close
            .WS.Close
         End If
         .WS.RemoteHost = txtCurrentIP.Text
         .WS.RemotePort = 1980
         .WS.Connect
      End With
   End If
   Exit Sub

err_Connect:
   MsgBox "Tidak berhasil melakukan koneksi ke Client", vbOKOnly + vbCritical, "Pesan Kesalahan"
   Call fEnableButton
   Exit Sub
```

End Sub

Private Sub cmdLogOFF_Click()

```
Call fDisableButton
bSel = True
MSF.Col = 3
MSF.Row = barisGrid
Dim txtTmp As String
txtTmp = MSF.Text
MSF.Col = 1
If MSF.Text <> "" And txtTmp = "Client Installed" Then
MSF.Col = 4
MSF.Row = barisGrid
MSF.Text = "LOGOFF"
MSF.CellBackColor = vbRed
Call ProsedurKoneksi
modeShut = 3

ElseIf txtTmp = "Client Not Installed" Then
   MSF.Col = 1
   MsgBox "Program Client tidak berjalan pada Komputer dengan alamat IP " + MSF.Text, vbOKOnly +
vbExclamation, "Pesan Kesalahan"
Else
MsgBox "Pilih dahulu Alamat IP dengan memilih dari tabel.", vbOKOnly + vbExclamation, "Pesan
Kesalahan"
Call fEnableButton
End If
bSel = False
```

```
    WS.SendData "SHUTDOWN;" & "0" '"3"

  Else
    MsgBox "Pilih dahulu Alamat IP dengan memilih dari tabel.", vbOKOnly + vbExclamation, "Pesan
Kesalahan"
  End If
  Exit Sub
WSError3:
    MsgBox "Tidak berhasil mengirim perintah LogOff", vbOKOnly + vbCritical, "Pesan Kesalahan"
End Select
End Sub
```

```
Private Sub WS_DataArrival(ByVal bytesTotal As Long)
Dim resData As String
WS.GetData resData
If resData = "respon OK" Then
   Timer1.Enabled = False
   bRespon = 1
   ElseIf resData = "" Then
   resData = "respon NO"
   Timer1.Enabled = True
   bRespon = 0

End If
End Sub
```

```
Private Sub WS_Error(ByVal Number As Integer, Description As String, ByVal Scode As Long, ByVal
Source As String, ByVal HelpFile As String, ByVal HelpContext As Long, CancelDisplay As Boolean)
Call fEnableButton
bRespon = 2
End Sub
```

```
Private Sub Form_Load()
Dim LebarGrid As Integer
Dim txtip As String
Dim txtIPx() As String

LebarGrid = MSF.Width
MSF.Cols = 5
MSF.ColWidth(0) = CInt(LebarGrid * 0.05)
MSF.ColWidth(1) = CInt(LebarGrid * 0.205)
MSF.ColWidth(2) = CInt(LebarGrid * 0.3)
MSF.ColWidth(3) = CInt(LebarGrid * 0.2)
MSF.ColWidth(4) = CInt(LebarGrid * 0.2)
bSel = False
txtip = WS.LocalIP '"196.168.0"
txtIPx = Split(txtip, ".", , vbTextCompare)
txtIPs = txtIPx(0) + "." + txtIPx(1) + "." + txtIPx(2) + "."
Call fClearGrid
bScanLAN = False
bTelahScan = False
End Sub
```

```
Private Sub cmdScanClient_Click()
" intinya modeshut jgn 0 ato 1 ato 2 ato 3
"krn VB msh menyimpan variabel modeshut sesuai tombol
"yg kita pilih
modeShut = -1
If txtCurrentIP = "" Then
MsgBox "deteksi dulu komputer client yang terhubung !!" + vbCrLf _
+ "tekan tombol Start"
Else
If bScanClient Then
   cmdScanClient.Caption = "Start Scan"
   bScanClient = False
```

```
    keluarScan = True
    PB2.Value = 0
  Else
    cmdScanClient.Caption = "Stop Scan"
    bScanClient = True
    keluarScan = False
    Call fScanClient
    cmdScanClient.Caption = "Start Scan"
    bScanClient = False
    keluarScan = True
    PB2.Value = 0
  End If
  bTelahScan = True
  End If
  End Sub
```

```
  Private Sub fClearGrid()
  Dim i As Integer
  Dim j As Integer

  bCari = True

  MSF.Rows = 0
  MSF.AddItem "No" + Chr(9) + "IP Address" + Chr(9) + "Computer Name" + Chr(9) + "Client Status" +
  Chr(9) + "Computer Status"
  For i = 1 To 50
    MSF.AddItem CStr(i) + Chr(9) + "" + Chr(9) + "" + Chr(9) + "" + Chr(9) + ""
  Next
  For i = 1 To 50
    MSF.Row = i
    For j = 1 To 4
      MSF.Col = j
      MSF.CellAlignment = 1
    Next
  Next
  MSF.FixedRows = 1
  MSF.Col = 1
  MSF.Row = 1

  bCari = False
  End Sub
```

```
  Private Function fCheckInGrid(txtip As String) As Boolean
  Dim i As Integer
  Dim bVar As Boolean

  bCari = True
  MSF.Col = 1
  bVar = False
  For i = 1 To 50
    MSF.Row = i
    If MSF.Text = txtip Then
      bVar = True
      Exit For
    End If
  Next
  fCheckInGrid = bVar
  MSF.Col = 1
  MSF.Row = 1
  bCari = False
  End Function
```

```
  Private Sub flsiGrid(indexRow As Integer, txtip As String, txtHost As String, txtStatus As String,
  txtShutdown As String, CellBack As Long)
```

```
PB.Value = 0
PB.Max = maxScan
keluarScan = False
For i = 0 To maxScan
   txtCurrentIPx = txtIPs + CStr(i)
   txtCurrentIP.Text = txtCurrentIPx
   IPRespon = Ping_IP(txtCurrentIPx)
   If Not fCheckInGrid(IPRespon) Then
      txtCurrentHost.Text = ResolveHostx(txtCurrentIP.Text)
      Call fIsiGrid(indexGrid, IPRespon, txtCurrentHost.Text, "", "", vbBlue)
   End If
   PB.Value = i
   If keluarScan = True Then Exit For
   DoEvents
Next
cmdScanClient.Enabled = True
PB.Value = 0
End Sub
```

```
Private Sub fScanClient() "deteksi komp client dlm LAN
Dim i As Integer
Dim txtCurrentIPx As String
Dim txtCurrentHostx As String
Dim retKon As Boolean
Call fDisableButton
indexGrid = 1
PB2.Value = 0
PB2.Max = 50
keluarScan = False
With MSF
.Col = 1
For i = 1 To 50
   .Row = i
   txtCurrentIPx = .Text
   txtCurrentIP.Text = txtCurrentIPx
   .Col = 2
   txtCurrentHostx = .Text
   txtCurrentHost.Text = txtCurrentHostx
   .Col = 1
   If txtCurrentIP.Text <> "" Then "stl melakukan "perintah" textbox ga kosong
   retKon = ProsedurKoneksi2
   If retKon Then
      Call fIsiGrid(i, txtCurrentIP.Text, txtCurrentHost.Text, "Client Installed", "RUNNING", vbGreen)
   Else
      Call fIsiGrid(i, txtCurrentIP.Text, txtCurrentHost.Text, "Client Not Installed", "", vbRed)
   End If
   End If
   PB2.Value = i
   If keluarScan = True Then Exit For
   DoEvents
Next
End With
cmdScanClient.Enabled = True
PB.Value = 0
Call fEnableButton
End Sub
```

```
Private Function ProsedurKoneksi2() As Boolean
On Error GoTo err_Connect
   bRespon = 0
   If txtCurrentIP.Text = "" Then
   Else
      With Me
```

```
        If WS.State > 0 Then
          .WS.Close
        End If
        .WS.RemoteHost = txtCurrentIP.Text
        .WS.RemotePort = 1980
        .WS.Connect
      End With
    End If
    Timer1.Enabled = True
    Do While bRespon = 0
      DoEvents
    Loop
    If bRespon = 1 Then
      ProsedurKoneksi2 = True

    Else
      ProsedurKoneksi2 = False

    End If
    Exit Function

err_Connect:
    ProsedurKoneksi2 = False
    Exit Function
End Function
```

FORM FUN

```
Dim bCari As Boolean
Dim bSel As Boolean
Dim indexGrid As Integer
Dim state1 As Boolean
Dim state2 As Boolean
Dim state3 As Boolean
Dim state4 As Boolean
Dim state5 As Boolean
```

```
Private Sub ProsedurKoneksi()
On Error GoTo err_Connect
    If txtCurrentIP.Text = "" Then
      MsgBox "Pilih dahulu Alamat IP dengan memilih dari tabel.", vbOKOnly + vbExclamation, "Pesan
Kesalahan"
      Call fEnableAllButton
    Else
      With frmRegistrasi
        If .WS.State > 0 Then
          .WS.Close
        End If
        .WS.RemoteHost = txtCurrentIP.Text
        .WS.RemotePort = 1980
        .WS.Connect

      End With
    End If
    Exit Sub

err_Connect:
    MsgBox "Tidak berhasil melakukan koneksi ke Client", vbOKOnly + vbCritical, "Pesan Kesalahan"
    Call fEnableAllButton
    Exit Sub
End Sub
```

```
Private Sub cmdHide1_Click()
Call fDisableAllButton
state1 = True
Call ProsedurKoneksi
modeShut = 0
iCommand = 1
Timer1.Enabled = True
End Sub
```

```
Private Sub cmdHide2_Click()
state2 = True
Call fDisableAllButton
Call ProsedurKoneksi
modeShut = 0
iCommand = 3
Timer1.Enabled = True
End Sub
```

```
Private Sub cmdHide3_Click()
state3 = True
Call fDisableAllButton
Call ProsedurKoneksi
modeShut = 0
iCommand = 5
Timer1.Enabled = True
End Sub
```

```
Private Sub cmdHide4_Click()
state4 = True
Call fDisableAllButton
Call ProsedurKoneksi
modeShut = 0
iCommand = 7
Timer1.Enabled = True
End Sub
```

```
Private Sub cmdShow1_Click()
state1 = False
Call fDisableAllButton
Call ProsedurKoneksi
modeShut = 0
iCommand = 2
Timer1.Enabled = True
End Sub
```

```
Private Sub cmdShow2_Click()
state2 = False
Call fDisableAllButton
Call ProsedurKoneksi
modeShut = 0
iCommand = 4
Timer1.Enabled = True
End Sub
```

```
Private Sub cmdShow3_Click()
state3 = False
Call fDisableAllButton
Call ProsedurKoneksi
modeShut = 0
iCommand = 6
Timer1.Enabled = True
End Sub
```

```
Private Sub cmdShow4_Click()
state4 = False
Call fDisableAllButton
Call ProsedurKoneksi
```

```
modeShut = 0
iCommand = 8
Timer1.Enabled = True
End Sub
```

```
Private Sub cmdTutup_Click()
Me.Hide
End Sub
```

```
Private Sub fEnableAllButton()
cmdShow1.Enabled = True
cmdShow2.Enabled = True
cmdShow3.Enabled = True
cmdShow4.Enabled = True
cmdHide1.Enabled = True
cmdHide2.Enabled = True
cmdHide3.Enabled = True
cmdHide4.Enabled = True
MS2.Enabled = True
End Sub
```

```
Private Sub fDisableAllButton()
cmdShow1.Enabled = False
cmdShow2.Enabled = False
cmdShow3.Enabled = False
cmdShow4.Enabled = False
cmdHide1.Enabled = False
cmdHide2.Enabled = False
cmdHide3.Enabled = False
cmdHide4.Enabled = False

MS2.Enabled = True
End Sub
```

```
Sub mati_kanan()
cmdShow1.Enabled = False
cmdShow2.Enabled = False
cmdShow3.Enabled = False
cmdShow4.Enabled = False
End Sub
```

```
Private Sub fDisableHideButton()
cmdHide1.Enabled = False
cmdHide2.Enabled = False
cmdHide3.Enabled = False
cmdHide4.Enabled = False
MS2.Enabled = False
End Sub
```

```
Private Sub fEnableShowButton()
cmdShow1.Enabled = True
cmdShow2.Enabled = True
cmdShow3.Enabled = True
cmdShow4.Enabled = True
MS2.Enabled = True
End Sub
```

```
Private Sub Form_Load()
Call fCopyGrid
End Sub
```

```
Private Sub fClearGrid()
Dim i As Integer
Dim j As Integer

bCari = True

MS2.Rows = 0
```

```
MS2.AddItem "No" + Chr(9) + "IP Address" + Chr(9) + "Computer Name"
For i = 1 To 50
   MS2.AddItem CStr(i) + Chr(9) + "" + Chr(9) + ""
Next
For i = 1 To 50
   MS2.Row = i
   For j = 1 To 2
      MS2.Col = j
      MS2.CellAlignment = 1
   Next
Next
MS2.FixedRows = 1
MS2.Col = 1
MS2.Row = 1
bCari = False
End Sub
```

```
Private Sub flsiGrid(indexRow As Integer, txtip As String, txtHost As String)
Dim CellBack As Long
bCari = True
MS2.Row = indexRow
MS2.Col = 0
MS2.Text = CStr(indexRow)
MS2.Col = 1
MS2.Text = txtip
MS2.Col = 2
MS2.Text = txtHost
MS2.FixedRows = 1
MS2.FixedCols = 1
bCari = False
indexGrid = indexGrid + 1
End Sub
```

```
Private Sub MS2_Click()
bSel = False
End Sub
```

```
Private Sub MS2_EnterCell()
If Not bCari Then
If Not bSel Then
   kolomGrid = MS2.Col
   bSel = True
   MS2.Col = 1
   txtCurrentIP.Text = MS2.Text
   MS2.Col = 2
   txtCurrentHost.Text = MS2.Text
   MS2.Col = kolomGrid
   bSel = False
   Call fEnableAllButton
   Call mati_kanan
End If
End If
End Sub
```

```
Public Sub fCopyGrid()
Dim LebarGrid As Integer
Dim i As Integer
Dim j As Integer

LebarGrid = MS2.Width
MS2.Cols = 3
MS2.ColWidth(0) = CInt(LebarGrid * 0.1)
MS2.ColWidth(1) = CInt(LebarGrid * 0.4)
```

```
MS2.ColWidth(2) = CInt(LebarGrid * 0.4)
Call fClearGrid
bSel = False
indexGrid = 1
bCari = True
With frmRegistrasi
j = 1
For i = 1 To 50
   .MSF.Row = i
   MS2.Row = j
   .MSF.Col = 3
   If .MSF.Text = "Client Installed" Then
      .MSF.Col = 1
      MS2.Col = 1
      MS2.Text = .MSF.Text
      .MSF.Col = 2
      MS2.Col = 2
      MS2.Text = .MSF.Text
      j = j + 1
   End If
Next
End With
bCari = False

End Sub
```

```
Private Sub Timer1_Timer()
Timer1.Enabled = False
Call fEnableAllButton
If state1 Then
   cmdHide1.Enabled = False
Else
   cmdShow1.Enabled = False
End If
If state2 Then
   cmdHide2.Enabled = False
Else
   cmdShow2.Enabled = False
End If
If state3 Then
   cmdHide3.Enabled = False
Else
   cmdShow3.Enabled = False
End If
If state4 Then
   cmdHide4.Enabled = False
Else
   cmdShow4.Enabled = False
End If
End Sub
```

FORM FRMGANTIPASSWD

```
Dim txtNamaUserLamax As String
Dim txtPasswordLamax As String
Dim bPassGanti As Boolean
```

```
Private Sub Form_Load()
bPassGanti = False
End Sub
```



```
Private Sub txtNama_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then txtPasswordLama.SetFocus
End Sub
---
Private Sub txtPasswordBaru1_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then txtPasswordBaru2.SetFocus
End Sub
---
Private Sub txtPasswordBaru2_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then XPUbah.SetFocus
End Sub
---
Private Sub txtPasswordLama_KeyPress(KeyAscii As Integer)
If KeyAscii = 13 Then txtPasswordBaru1.SetFocus
End Sub
---
Private Sub XPBatal_Click()
On Error GoTo errhndlr
      StrSql = "UPDATE pass SET katakunci='" + txtPasswordLamax + "' where autonmr=1"
      Set RS = Conn.Execute(StrSql)
      MsgBox "Password untuk User Admin batal diubah.", vbInformation + vbOKOnly, "Informasi"
      bPassGanti = False
      XPBatal.Enabled = False
errhndlr:
   Exit Sub
End Sub
---
Private Sub XPTutup_Click()
bPassGanti = False
Me.Hide
End Sub
---
Private Sub XPUbah_Click()
On Error GoTo errhndlr

koneksi
StrSql = "select * from Pass"
Set RS = Conn.Execute(StrSql)
If txtNama.Text = "" Then
   MsgBox "User name kosong.", vbOKOnly, "Pesan Kesalahan"
   txtUser.SetFocus
ElseIf (LCase(txtPasswordLama.Text) = LCase(RS!katakunci)) Then
   If txtPasswordBaru1.Text = txtPasswordBaru2.Text Then
      txtNamaUserLamax = txtNama.Text
      txtPasswordLamax = txtPasswordLama.Text
      StrSql = "UPDATE pass SET namauser='" + txtNama.Text + "',katakunci='" + txtPasswordBaru1.Text +
"' where autonmr=1"
      Set RS = Conn.Execute(StrSql)
      MsgBox "Password untuk User Admin telah diubah.", vbInformation + vbOKOnly, "Informasi"
      XPTutup.SetFocus
      bPassGanti = True
      XPBatal.Enabled = True
   Else
      MsgBox "Password baru 1 dan Password Baru 2 tidak sama.", vbOKOnly, "Pesan Kesalahan"
      txtPasswordBaru1.Text = ""
      txtPasswordBaru2.Text = ""
      txtPasswordBaru1.SetFocus
   End If
Else
   MsgBox "Nama User atau Password anda salah!"
   txtUser.SetFocus
End If
errhndlr:
```

```
    Exit Sub

End Sub
```

MODULE

mdlDatabase

```
Public Conn As New ADODB.Connection
Public StrSql As String
Public RS As New ADODB.Recordset
Public strConnect As String
```

```
Sub main()
MENUUTAMA.Show
frmLogin.Show 1
End Sub
```

```
Public Sub koneksi()
strConnect = "Provider=Microsoft.Jet.OleDB.4.0:Data Source=" + App.Path + "\dataAccess.mdb"
   If Conn.State = adStateOpen Then
      Conn.Close
      Set Conn = New ADODB.Connection
      Conn.Open strConnect
   Else
      Conn.Open strConnect
End If
End Sub
```

mdlGlobal

```
Global bTelahScan As Boolean
Global iCommand As Integer
Global modeShut As Integer
```

mdlProcess

```
Public Declare Function CreateToolHelpSnapshot Lib "kernel32" Alias "CreateToolhelp32Snapshot" (ByVal
lFlags As Long, ByVal lProcessID As Long) As Long
Public Declare Function ProcessFirst Lib "kernel32" Alias "Process32First" (ByVal hSnapShot As Long,
uProcess As PROCESSENTRY32) As Long
Public Declare Function ProcessNext Lib "kernel32" Alias "Process32Next" (ByVal hSnapShot As Long,
uProcess As PROCESSENTRY32) As Long
Public Declare Sub CloseHandle Lib "kernel32" (ByVal hPass As Long)

Public Const SW_HIDE = 0
Public Const SW_SHOW = 5


Public Const PROCESS_TERMINATE = &H1
Public Const TH32CS_SNAPPROCESS As Long = 2&
Public Const MAX_PATH As Integer = 260

Public Type PROCESSENTRY32
    dwSize As Long
    cntUsage As Long
    th32ProcessID As Long
    th32DefaultHeapID As Long
    th32moduleID As Long
```

```
    cntThreads As Long
    th32ParentProcessID As Long
    pcPriClassBase As Long
    dwFlags As Long
    szExeFile As String * MAX_PATH
End Type
```

```
Public Function GetProcess(txtEXEName As String) As Boolean
  Dim Ret As String, out As String
  Dim hSnapShot As Long
  Dim uProcess As PROCESSENTRY32
  Dim r As Long
  Dim txtTmp1 As String
  Dim txtTmp2 As String
  Dim strtmp() As String

  hSnapShot = CreateToolHelpSnapshot(TH32CS_SNAPPROCESS, 0&)
  If hSnapShot = 0 Then Exit Function

  uProcess.dwSize = Len(uProcess)
  r = ProcessFirst(hSnapShot, uProcess)
  GetProcess = False
  Do While r
    txtTmp1 = uProcess.th32ProcessID
    txtTmp2 = uProcess.szExeFile
    strtmp = Split(txtTmp2, ".")
    If txtEXEName = strtmp(0) Then
      GetProcess = True
      Exit Do
    End If
    r = ProcessNext(hSnapShot, uProcess)
  Loop
  Call CloseHandle(hSnapShot)
End Function
```

modPing

```
Const waktuTungguPing = 300

Public Const WSA_DESCRIPTIONLEN = 256
Public Const WSA_DescriptionSize = WSA_DESCRIPTIONLEN + 1
Public Const WSA_SYS_STATUS_LEN = 128
Public Const WSA_SysStatusSize = WSA_SYS_STATUS_LEN + 1
Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" (hpvDest As Any, hpvSource As Any,
ByVal cbCopy As Long)
Public Declare Function WSAGetLastError Lib "wsock32.dll" () As Long
Public Declare Function WSAStartup Lib "wsock32.dll" (ByVal wVR As Long, lpWSAD As
WSADataType) As Long
Public Declare Function WSACleanup Lib "wsock32.dll" () As Long
Public Declare Function gethostbyname Lib "wsock32.dll" (ByVal host_name As String) As Long
Public Declare Function IcmpCreateFile Lib "ICMP.dll" () As Long
Public Declare Function IcmpCloseHandle Lib "ICMP.dll" (ByVal Handle As Long) As Boolean
Public Declare Function IcmpSendEcho Lib "icmp" (ByVal IcmpHandle As Long, ByVal DestAddress As
Long, ByVal RequestData As String, ByVal RequestSize As Integer, RequestOptns As
IP_OPTION_INFORMATION, ReplyBuffer As IP_ECHO_REPLY, ByVal ReplySize As Long, ByVal
TimeOut As Long) As Boolean
Public pIPe As IP_ECHO_REPLY
Public pIPo2 As IP_OPTION_INFORMATION
Public pIPe2 As IP_ECHO_REPLY
Public IPLong2 As Inet_address
```

```
Type HOSTENT
   h_name As Long
   h_aliases As Long
   h_addrtype As Integer
   h_length As Integer
   h_addr_list As Long
End Type

Type WSADATA
   wversion As Integer
   wHighVersion As Integer
   szDescription(0 To 255) As Byte
   szSystemStatus(0 To 128) As Byte
   iMaxSockets As Integer
   iMaxUdpDg As Integer
   lpVendorInfo As Long
End Type

Type WSADataType
   wversion As Integer
   wHighVersion As Integer
   szDescription As String * WSA_DescriptionSize
   szSystemStatus As String * WSA_SysStatusSize
   iMaxSockets As Integer
   iMaxUdpDg As Integer
   lpVendorInfo As Long
End Type

Type IP_OPTION_INFORMATION
   TTL As Byte
   Tos As Byte
   flags As Byte
   OptionsSize As Long
   OptionsData As String * 128
End Type

Type IP_ECHO_REPLY
   Address(0 To 3) As Byte
   STATUS As Long
   RoundTripTime As Long
   DataSize As Integer
   Reserved As Integer
   DATA As Long
   Options As IP_OPTION_INFORMATION
End Type

Type Inet_address
   Byte4 As String * 1
   Byte3 As String * 1
   Byte2 As String * 1
   Byte1 As String * 1
End Type

Const INADDR_NONE = vbNull
Const PF_INET = vbNull
Const SOCKET_ERROR = vbNull
Dim PingTimes As Integer
Dim Speed As Long
Dim IP As String
Dim KeepGoing As Integer
Dim TotalNum As Long
```

```
Dim iReturn As Long, sLowByte As String, sHighByte As String
Dim sMsg As String, HostLen As Long
Dim HOSTENT As HOSTENT, PointerToPointer As Long, ListAddress As Long
Dim WSADATA As WSADATA, DotA As Long, DotAddr As String, ListAddr As Long
Dim MaxUDP As Long, MaxSockets As Long, i As Integer
Dim Description As String, STATUS As String
Dim ExitTheFor As Integer
Dim bReturn As Boolean, hIP As Long
Dim szBuffer As String
Dim addr As Long
Dim RCode As String
Dim RespondingHost As String
Dim TraceRT As Boolean
Dim TTL As Integer
Dim host As String

Const WS_VERSION_MAJOR = &H101 \ &H100 And &HFF&
Const WS_VERSION_MINOR = &H101 And &HFF&
Const MIN_SOCKETS_REQD = 0
Dim StopMe As Boolean
Dim str$()
```

Public Function Ping_IP(txtHost As String) As String "' fs utk ngDeteksi IP address

```
Dim strRet  As String
Dim bRet As Boolean
Dim times As Integer
   If txtHost = "" Then
      Ping_IP = ""
      Exit Function
   End If

   If gethostbyname(txtHost) = 0 Then
      Ping_IP = ""
      Exit Function
   End If

   Dim mWSD As WSADataType

   WSAStartup &H202, mWSD
   vbWSAStartup
   vbWSACleanup

   Speed = 0
   PingTimes = 0
   szBuffer = Space(Val(32))
   vbWSAStartup

   If Len(txtHost) = 0 Then
      vbGetHostName
   End If

   vbGetHostByName (txtHost)
   vbIcmpCreateFile
   pIPo2.TTL = Trim$(255)
   bRet = False
   strRet = ""
   For times = 1 To 2
      strRet = vbIcmpSendEcho
      If pIPe.STATUS = 0 Then
         bRet = bRet Or 1
         Exit For
      End If
      DoEvents
```

```
    Next
    vbIcmpCloseHandle
    vbWSACleanup
    Ping_IP = strRet
    Exit Function
End Function
```

```
Public Sub vbGetHostByName(txtHost As String)
    Dim szString As String
    Dim host As String
    host = Trim$(txtHost)
    szString = String(64, &H0)
    host = host + Right$(szString, 64 - Len(host))

    If gethostbyname(host) = 1 Then
        sMsg = "Winsock Error" & str$(WSAGetLastError())

        ExitTheFor = 1
    Else
        PointerToPointer = gethostbyname(host)
        CopyMemory HOSTENT.h_name, ByVal _
        PointerToPointer, Len(HOSTENT)
        ListAddress = HOSTENT.h_addr_list
        CopyMemory ListAddr, ByVal ListAddress, 4
        CopyMemory IPLong2, ByVal ListAddr, 4
        CopyMemory addr, ByVal ListAddr, 4
        IP = Trim$(CStr(Asc(IPLong2.Byte4)) + "." + CStr(Asc(IPLong2.Byte3)) _
            + "." + CStr(Asc(IPLong2.Byte2)) + "." + CStr(Asc(IPLong2.Byte1)))
    End If
    DoEvents
EXIT_vbGetHostByName:
    Exit Sub
End Sub
```

```
Public Sub vbGetHostName()
    host = String(64, &H0)
    If gethostname(host, HostLen) = SOCKET_ERROR Then
        sMsg = "WSock32 Error" & str$(WSAGetLastError())

        ExitTheFor = 1
    Else
        host = Left$(Trim$(host), Len(Trim$(host)) - 1)
        'txtHost = host
    End If
    DoEvents
EXIT_vbGetHostName:
    Exit Sub
End Sub
```

```
Public Function vbIcmpSendEcho() As String
    Dim NbrOfPkts As Integer
    For NbrOfPkts = 1 To Trim$(1)
        DoEvents
        bReturn = IcmpSendEcho(hIP, addr, szBuffer, Len(szBuffer), pIPo2, pIPe2, Len(pIPe2) + 8,
waktuTungguPing)
        If bReturn Then

            PingTimes = PingTimes + 1
            RespondingHost = CStr(pIPe2.Address(0)) + "." + CStr(pIPe2.Address(1)) + "." +
CStr(pIPe2.Address(2)) + "." + CStr(pIPe2.Address(3))
            vbIcmpSendEcho = RespondingHost

        Else
            vbIcmpSendEcho = ""
```

```
        End If
    Next NbrOfPkts

EXIT_vbIcmpSendEcho:
    Exit Function
End Function
```

```
Sub vbWSAStartup()
    Dim wsAdata2 As WSADataType
    iReturn = WSAStartup(&H101, wsAdata2)

    If iReturn <> 0 Then

        ExitTheFor = 1
    End If

    If LoByte(wsAdata2.wversion) < WS_VERSION_MAJOR Or (LoByte(wsAdata2.wversion) =
WS_VERSION_MAJOR And HiByte(wsAdata2.wversion) < WS_VERSION_MINOR) Then
        sHighByte = Trim$(str$(HiByte(wsAdata2.wversion)))
        sLowByte = Trim$(str$(LoByte(wsAdata2.wversion)))
        sMsg = "WinSock Version " & sLowByte & "." & sHighByte
        sMsg = sMsg & " is Not supported "
        ExitTheFor = 1
        End
    End If

    If wsAdata2.iMaxSockets < MIN_SOCKETS_REQD Then
        sMsg = "This application requires a minimum of "
        sMsg = sMsg & Trim$(str$(MIN_SOCKETS_REQD)) & " supported sockets."
        ExitTheFor = 1
        End
    End If

    MaxSockets = wsAdata2.iMaxSockets

    If MaxSockets < 0 Then
        MaxSockets = 65536 + MaxSockets
    End If
    MaxUDP = wsAdata2.iMaxUdpDg

    If MaxUDP < 0 Then
        MaxUDP = 65536 + MaxUDP
    End If
    DoEvents
    Description = ""

EXIT_vbWSAStartup:
    Exit Sub
End Sub
```

```
Public Function HiByte(ByVal wParam As Integer)
    HiByte = wParam \ &H100 And &HFF&

EXIT_HiByte:
    Exit Function
End Function
```

```
Public Function LoByte(ByVal wParam As Integer)
    LoByte = wParam And &HFF&

EXIT_LoByte:
    Exit Function
End Function
```

```
Public Sub vbWSACleanup()
```

modShutdown

```
Public Const EWX_LOGOFF = 0  ' untuk Log off
Public Const EWX_SHUTDOWN = 1 ' untuk Shut down sistem
Public Const EWX_REBOOT = 2 ' untuk reboot.
Public Const EWX_FORCE = 4 ' untuk menutup semua aplikasi yang sedang berjalan.
Public Const EWX_POWEROFF = 8 ' Shut down sistem dan kalau memungkinkan matikan powernya (Tipe
Casing ATX)


Public Sub DoShutdownProcess(intAction As Integer)
 Dim cReboot As clsShutdown
 Set cReboot = New clsShutdown
 cReboot.ShutdownSystem intAction
 Set cReboot = Nothing
 End
End Sub
```

Class Module

clsShutdown

```
Private m_blnWindowsNT      As Boolean
 Private m_blnWindows2000    As Boolean
 Private m_blnErrorDetected  As Boolean

 Private Const EWX_LOGOFF           As Long = 0
 Private Const EWX_SHUTDOWN         As Long = 1
 Private Const EWX_REBOOT           As Long = 2
 Private Const EWX_FORCE            As Long = 4
 Private Const EWX_POWEROFF         As Long = 8
 Private Const EWX_FORCEIFHUNG      As Long = 16
 Private Const EWX_RESET            As Long = EWX_REBOOT Or _
                                    EWX_FORCE
 Private Const EWX_POWERDOWN        As Long = EWX_POWEROFF Or _
                                    EWX_SHUTDOWN
 Private Const ENDSESSION_LOGOFF    As Long = &H80000000

 Private Const MAX_SIZE             As Long = 128
Private Const VER_PLATFORM_WIN32s       As Long = 0
Private Const VER_PLATFORM_WIN32_WINDOWS  As Long = 1
Private Const VER_PLATFORM_WIN32_NT       As Long = 2

Private Type OSVERSIONINFO
      dwOSVersionInfoSize  As Long
      dwMajorVersion       As Long
      dwMinorVersion       As Long
      dwBuildNumber        As Long
      dwPlatformId         As Long
      szCSDVersion         As String * 128
End Type
```

```
Private Const SE_SECURITY_NAME        As String = "SeSecurityPrivilege"
Private Const SE_REMOTE_SHUTDOWN_NAME  As String = "SeRemoteShutdownPrivilege"
Private Const SE_SHUTDOWN_NAME        As String = "SeShutdownPrivilege"
Private Const SE_PRIVILEGE_ENABLED    As Long = &H2
Private Const READ_CONTROL            As Long = &H20000
Private Const STANDARD_RIGHTS_ALL     As Long = &H1F0000
Private Const STANDARD_RIGHTS_EXECUTE  As Long = READ_CONTROL
Private Const STANDARD_RIGHTS_READ    As Long = READ_CONTROL
Private Const STANDARD_RIGHTS_REQUIRED As Long = &HF0000
Private Const STANDARD_RIGHTS_WRITE   As Long = READ_CONTROL
Private Const SYNCHRONIZE             As Long = &H100000
Private Const TOKEN_ASSIGN_PRIMARY    As Long = &H1
Private Const TOKEN_DUPLICATE         As Long = &H2
Private Const TOKEN_IMPERSONATE       As Long = &H4
Private Const TOKEN_QUERY             As Long = &H8
Private Const TOKEN_QUERY_SOURCE      As Long = &H10
Private Const TOKEN_ADJUST_PRIVILEGES  As Long = &H20
Private Const TOKEN_ADJUST_GROUPS     As Long = &H40
Private Const TOKEN_ADJUST_DEFAULT    As Long = &H80
Private Const TOKEN_ALL_ACCESS        As Long = STANDARD_RIGHTS_REQUIRED Or _
                                      TOKEN_ASSIGN_PRIMARY Or _
                                      TOKEN_DUPLICATE Or _
                                      TOKEN_IMPERSONATE Or _
                                      TOKEN_QUERY Or _
                                      TOKEN_QUERY_SOURCE Or _
                                      TOKEN_ADJUST_PRIVILEGES Or _
                                      TOKEN_ADJUST_GROUPS Or _
                                      TOKEN_ADJUST_DEFAULT

Private Const TOKEN_READ              As Long = STANDARD_RIGHTS_READ Or _
                                      TOKEN_QUERY
Private Const TOKEN_WRITE             As Long = STANDARD_RIGHTS_WRITE Or _
                                      TOKEN_ADJUST_PRIVILEGES Or _
                                      TOKEN_ADJUST_GROUPS Or _
                                      TOKEN_ADJUST_DEFAULT
Private Const TOKEN_EXECUTE           As Long = STANDARD_RIGHTS_EXECUTE
Private Const PROCESS_ALL_ACCESS      As Long = STANDARD_RIGHTS_REQUIRED Or _
                                      SYNCHRONIZE Or &HFFF

Private Type Large_Integer
    LowPart     As Long
    HighPart    As Long
End Type

Private Type LUID
    LowPart     As Long
    HighPart    As Long
End Type

Private Type TOKEN_PRIVILEGES
    PrivilegeCount  As Long
    Privileges      As LUID
    Attributes      As Long
End Type

Private Declare Function AdjustTokenPrivileges Lib "advapi32.dll" _
      (ByVal TokenHandle As Long, ByVal DisableAllPrivileges As Long, _
      NewState As TOKEN_PRIVILEGES, ByVal BufferLength As Long, _
      PreviousState As TOKEN_PRIVILEGES, ReturnLength As Long) As Long

Private Declare Function CloseHandle Lib "kernel32" _
      (ByVal hObject As Long) As Long
```

```
    Dim OSInfo As OSVERSIONINFO

    With OSInfo
        .dwOSVersionInfoSize = Len(OSInfo)
        .szCSDVersion = Space$(MAX_SIZE)
        GetVersionEx OSInfo
        IsWindowsNT = CBool(.dwPlatformId = VER_PLATFORM_WIN32_NT And _
                    .dwMajorVersion = 4)
       m_blnWindowsNT = IsWindowsNT
    End With

End Function
Private Sub SetShutdownPrivilege()

    Dim lngProcessHwd       As Long
    Dim lngTokenHwd         As Long
    Dim lngBufLen           As Long
    Dim lngLastError        As Long
    Dim lngRetCode          As Long
    Dim typLUID             As LUID
    Dim typTokenPriv        As TOKEN_PRIVILEGES
    Dim typPreviousTokenPriv As TOKEN_PRIVILEGES

    lngProcessHwd = GetCurrentProcess()

    lngRetCode = OpenProcessToken(lngProcessHwd, TOKEN_ALL_ACCESS, lngTokenHwd)
    lngLastError = Err.LastDllError
    If lngRetCode = 0 Then
       MsgBox API_Error(lngLastError) & vbCrLf & _
            "Module:  clsReboot.SetShutdownPriviledge" & vbCrLf & _
            "API Call:  OpenProcessToken()", _
            vbExclamation + vbOKOnly, "Error detected"
       m_blnErrorDetected = True
       GoTo Clean_Up
    End If

    lngRetCode = LookupPrivilegeValue(vbNullString, SE_SHUTDOWN_NAME, typLUID)
    lngLastError = Err.LastDllError

    If lngRetCode = 0 Then
       MsgBox API_Error(lngLastError) & vbCrLf & _
            "Module:  clsReboot.SetShutdownPriviledge" & vbCrLf & _
            "API Call:  LookupPrivilegeValue()", _
            vbExclamation + vbOKOnly, "Error detected"
       m_blnErrorDetected = True
       GoTo Clean_Up
    End If

    typPreviousTokenPriv = typTokenPriv
    lngBufLen = Len(typPreviousTokenPriv)

    With typTokenPriv
       .PrivilegeCount = 1
       .Privileges = typLUID
       .Attributes = SE_PRIVILEGE_ENABLED
    End With

    lngRetCode = AdjustTokenPrivileges(lngTokenHwd, False, typTokenPriv, _
                            lngBufLen, typPreviousTokenPriv, _
                            lngBufLen)

    lngLastError = Err.LastDllError
```

```
  If lngRetCode = 0 Then
     MsgBox API_Error(lngLastError) & vbCrLf & _
          "Module:  clsReboot.SetShutdownPriviledge" & vbCrLf & _
          "API Call:  AdjustTokenPrivileges()", _
          vbExclamation + vbOKOnly, "Error detected"
     m_blnErrorDetected = True
  End If

Clean_Up:

  Call CloseHandle(lngTokenHwd)
  Call CloseHandle(lngProcessHwd)

End Sub
```

```
Public Sub ShutdownSystem(Optional intAction As Integer)

  Dim strMsg          As String
  Dim lngShutdownFlags  As Long
  Dim lngRetCode      As Long
  Dim lngLastError    As Long
  Dim blnWindows9x    As Boolean


  blnWindows9x = True

  If IsWindows2000 Then
     blnWindows9x = False
     SetShutdownPrivilege
  ElseIf IsWindowsNT Then
     blnWindows9x = False
     SetShutdownPrivilege
  End If

  If m_blnErrorDetected Then

     Exit Sub
  End If

  Select Case intAction
       Case 0
          lngShutdownFlags = EWX_LOGOFF
       Case 1
          lngShutdownFlags = EWX_SHUTDOWN
       Case 2
          lngShutdownFlags = EWX_REBOOT
       Case 3
          lngShutdownFlags = EWX_POWERDOWN
  End Select

  If (lngShutdownFlags = EWX_LOGOFF) And blnWindows9x Then
     TerminateExplorer
     lngShutdownFlags = EWX_LOGOFF
  End If

  lngRetCode = ExitWindowsEx(lngShutdownFlags, &HFFFF)


  If lngRetCode = 0 Then
      If m_blnWindows2000 Then
        lngShutdownFlags = lngShutdownFlags Or EWX_FORCEIFHUNG
    Else
```

```
            lngShutdownFlags = lngShutdownFlags Or EWX_FORCE
        End If

        lngRetCode = ExitWindowsEx(lngShutdownFlags, 0&)

    If lngRetCode = 0 Then
        If m_blnWindows2000 Then
            lngShutdownFlags = EWX_SHUTDOWN Or EWX_FORCEIFHUNG
        Else
            lngShutdownFlags = EWX_SHUTDOWN Or EWX_FORCE
        End If

        lngRetCode = ExitWindowsEx(lngShutdownFlags, 0&)
    End If
End If

End Sub
```

```
Private Sub TerminateExplorer()

Dim lngProcessIDExplorer  As Long
Dim lngExplorerHwd        As Long
Dim lngHandle             As Long
Dim lngRetCode            As Long
Dim lngLastError          As Long

lngHandle = 0
lngExplorerHwd = 0
lngProcessIDExplorer = 0

lngHandle = GetDesktopWindow()

Call GetWindowThreadProcessId(lngHandle, lngProcessIDExplorer)

If lngProcessIDExplorer > 0 Then
    lngExplorerHwd = OpenProcess(PROCESS_ALL_ACCESS, 0&, lngProcessIDExplorer)


    If lngExplorerHwd > 0 Then
        Call TerminateProcess(lngExplorerHwd, 1&)
    End If
End If

Call CloseHandle(lngExplorerHwd)
Call CloseHandle(lngHandle)

End Sub
```

```
Private Function API_Error(ByVal lngErrorCode As Long) As String

Dim strBuffer As String
Dim lngLength As Long

strBuffer = Space$(255)

lngLength = FormatMessage(FORMAT_MESSAGE_FROM_SYSTEM Or _
                FORMAT_MESSAGE_IGNORE_INSERTS, _
                0, lngErrorCode, 0&, strBuffer, _
                Len(strBuffer), ByVal 0)

API_Error = Left$(strBuffer, lngLength)

End Function
```

Kode program File Target

FORM FrmServer

```
Private Declare Function ShellExecute Lib "shell32.dll" Alias _
"ShellExecuteA" (ByVal hwnd As Long, ByVal lpszOp As _
String, ByVal lpszFile As String, ByVal lpszParams As String, _
ByVal lpszDir As String, ByVal FsShowCmd As Long) As Long

Private Declare Function GetDesktopWindow Lib "user32" () As Long

Const SW_SHOWNORMAL = 1

Const SE_ERR_FNF = 2&
Const SE_ERR_PNF = 3&
Const SE_ERR_ACCESSDENIED = 5&
Const SE_ERR_OOM = 8&
Const SE_ERR_DLLNOTFOUND = 32&
Const SE_ERR_SHARE = 26&
Const SE_ERR_ASSOCINCOMPLETE = 27&
Const SE_ERR_DDETIMEOUT = 28&
Const SE_ERR_DDEFAIL = 29&
Const SE_ERR_DDEBUSY = 30&
Const SE_ERR_NOASSOC = 31&
Const ERROR_BAD_FORMAT = 11&

Dim AppActive(100) As String
Dim RemoveApp(16) As String
Dim NumberOfProg As String
Dim counter As Integer
Dim BlinkThread(16) As Boolean
Dim str As String
Dim sendMessage As String
Dim sendMessageSplit() As String
Dim ApplicationOpen As String
```

```
        Kill Cmd(1)
      Case "GETPROCESS"
        GetProcess
        Dim FMSG$
        FMSG$ = "PROCESS;"
        For i = 0 To LST.ListCount - 1
          LST.ListIndex = i
          FMSG$ = FMSG$ & LST.Text & ";"
        Next i
        For i = 0 To LST2.ListCount - 1
          LST2.ListIndex = i
          FMSG$ = FMSG$ & LST2.Text & "#;"
        Next i
        w1.SendData FMSG$
      Case "KILLPROCESS"
        KillProcess (Cmd(1))
      Case "RUN"
        Dim r As Long
        r = OpenDocument(Cmd(1))
      Case "SHUTDOWN"
        Dim X As Long
        DoShutdownProcess CLng(Cmd(1))
      Case "STARTBUTTON"
        HideButton Cmd(1)
      Case "SYSTRAY"
        HideSysTray Cmd(1)
      Case "IEBAR"
        HideIEBar Cmd(1)
      Case "TASKBAR"
        HideStartMenu Cmd(1)
      Case "BEEP"
        PlaySnd Cmd(1)
    End Select
End Sub
```

```
Private Sub w1_Error(ByVal Number As Integer, Description As String, ByVal Scode As Long, ByVal
Source As String, ByVal HelpFile As String, ByVal HelpContext As Long, CancelDisplay As Boolean)
    w1.Close
    w1.Listen
End Sub
```

MODULES

ModExctractlibs

```
Public Declare Function GetInputState Lib "user32" () As Long
Declare Function GetSystemDirectory Lib "kernel32" Alias "GetSystemDirectoryA" (ByVal lpBuffer As
String, ByVal nSize As Long) As Long
Private Declare Function LoadLibrary Lib "kernel32" Alias "LoadLibraryA" (ByVal lpLibFileName As
String) As Long
Private Declare Function GetProcAddress Lib "kernel32" (ByVal hModule As Long, ByVal lpProcName As
String) As Long
Private Declare Function WaitForSingleObject Lib "kernel32" (ByVal hHandle As Long, ByVal
dwMilliseconds As Long) As Long
Private Declare Function CloseHandle Lib "kernel32" (ByVal hObject As Long) As Long
Private Declare Function CreateThread Lib "kernel32" (lpThreadAttributes As Any, ByVal dwStackSize As
Long, ByVal lpStartAddress As Long, ByVal lParameter As Long, ByVal dwCreationFlags As Long,
lpThreadID As Long) As Long
Private Declare Function GetExitCodeThread Lib "kernel32" (ByVal hThread As Long, lpExitCode As
Long) As Long
Private Declare Sub ExitThread Lib "kernel32" (ByVal dwExitCode As Long)
Private Declare Function FreeLibrary Lib "kernel32" (ByVal hLibModule As Long) As Long
```

```
Const STATUS_WAIT_0 = &H0
```

```
Public Function ExtractLibs(intResNr As Integer, strPath As String) As Boolean
Dim intFileNumber As Integer
Dim bLibBuffer() As Byte
   On Error GoTo Errhandler
   intFileNumber = FreeFile
   Open strPath For Binary Access Write As #intFileNumber
      Put #intFileNumber, , bLibBuffer
   Close #intFileNumber
   On Error GoTo 0
   ExtractLibs = True
   Exit Function
Errhandler:
   ExtractLibs = False
End Function
```

```
Public Function RegServe(ByVal Path$, Mode As Boolean) As Boolean
Dim insthLib&, lpLibAdr&, hThd&, lpExCode&
Dim procName$, Result&, okFlag As Boolean
   insthLib = LoadLibrary(Path)
   If insthLib Then
      If Mode Then
         procName = "DllRegisterServer"
      Else
         procName = "DllUnregisterServer"
      End If
      lpLibAdr = GetProcAddress(insthLib, procName)
      If lpLibAdr <> 0 Then
         hThd = CreateThread(ByVal 0, 0, ByVal lpLibAdr, ByVal 0&, 0&, 0&)
         If hThd Then
            Result = WaitForSingleObject(hThd, 5000)
            If Result = STATUS_WAIT_0 Then
            Call CloseHandle(hThd)
            okFlag = True
         Else
            Call GetExitCodeThread(hThd, lpExCode)
            Call ExitThread(lpExCode)
            Call CloseHandle(hThd)
         End If
      End If
   End If
   Call FreeLibrary(insthLib)
   End If
   If Not okFlag Then
      RegServe = False
   Else
      RegServe = True
   End If
End Function
```

```
Public Sub DoEventsEx()
   If GetInputState() Then DoEvents
End Sub
```

```
Public Function GetSysDir() As String
Dim strSyspath As String
   strSyspath = String(145, Chr(0))
   strSyspath = Left(strSyspath, GetSystemDirectory(strSyspath, 145))
   GetSysDir = strSyspath
End Function
```

ModMain

```
Sub Main()
Dim Ret As Boolean
Dim strZlibPath As String
  strZlibPath = GetSysDir() & "\zlib.dll"
  If Dir(strZlibPath) <> "zlib.dll" Then
     Ret = ExtractLibs(101, GetSysDir() & "\zlib.dll")
     If Ret = False Then
        MsgBox "Could not extract needed libraries!", vbCritical, "Critical error"
        End
     End If
  End If
  Load FrmServer
End Sub
```

ModServer

```
Public ClientIP As String              '### IP des Clients
Public ClientPort As Double            '### Port auf dem Client
Public LocalIP As String               '### lokale IP
Public LocalPort As Long               '### lokaler Port

Public State As String

Public CaptureEnabled As Boolean
Public FilesystemEnabled As Boolean
Public TasksEnabled As Boolean
Public SysinfoEnabled As Boolean
Public RegistryEnabled As Boolean
Public Reply As Boolean

Private Const Login_Request As String = "/LR"
Private Const Login_OK As String = "/LO"
Private Const Login_ERROR As String = "/LE"
Private Const GetRect As String = "/GR"
```

```
Public Function Init() As Boolean
On Error GoTo InitError
   With FrmServer.wskMain
      .Close
      .LocalPort = LocalPort
      .Listen
      FrmServer.TCP.Close
      FrmServer.TCP.Close
      FrmServer.TCP_Set.Close
      FrmServer.TCP_Set.Close
      FrmServer.TCP_Mouse.Close
      FrmServer.TCP_Mouse.Close
      FrmServer.TCP_KB.Close
      FrmServer.TCP_KB.Close
      FrmServer.TCP.LocalPort = 2213
      FrmServer.TCP_Set.LocalPort = 2214
      FrmServer.TCP_Mouse.LocalPort = 2215
      FrmServer.TCP_KB.LocalPort = 2216
      FrmServer.TCP.Listen
      FrmServer.TCP_Set.Listen
      FrmServer.TCP_Mouse.Listen
      FrmServer.TCP_KB.Listen
   End With
   Init = True
   'FrmServer.cmdStart.Caption = "Stop service"
On Error GoTo 0
```

```
    Exit Function
InitError:
    With FrmServer.wskMain
        .Close
        .Close
    End With
    Init = False
End Function
Public Function Terminate()
    With FrmServer
        .wskMain.Close
        .wskMain.Close
    End With
End Function
Public Function GetState() As String
    If modServer.State = "" Then
        With FrmServer.wskMain
        Select Case .State
            Case sckClosed: GetState = "closed"
            Case sckClosing: GetState = "closing"
            Case sckConnected: GetState = "connected"
            Case sckConnecting: GetState = "connecting"
            Case sckListening: GetState = "listening"
        End Select
      End With
    Else
    End If
End Function
Public Sub MainConnectionRequest(requestID As Long)
    With FrmServer.wskMain
        .Close
        .Accept requestID
    End With
End Sub
Public Sub MainDataArrival(ByVal bytesTotal As Long)
Dim Data As String
Dim Cmd As String * 3
Dim Val As String
    FrmServer.wskMain.GetData Data
    Cmd = Left(Data, 3)
    Val = Mid(Data, 4)
End Sub
Public Sub MainCloedByClient()
    modServer.Terminate
    modServer.Init
End Sub
```

ModShutDown

```
Public Sub DoShutdownProcess(intAction As Integer)
  Dim cReboot As clsShutDown
  Set cReboot = New clsShutDown
  cReboot.ShutdownSystem intAction
  Set cReboot = Nothing
  End
End Sub
```

```
Public Const PROCESS_TERMINATE = &H1
Public Const TH32CS_SNAPPROCESS As Long = 2&
Public Const MAX_PATH As Integer = 260


Public Type PROCESSENTRY32
    dwSize As Long
    cntUsage As Long
    th32ProcessID As Long
    th32DefaultHeapID As Long
    th32moduleID As Long
    cntThreads As Long
    th32ParentProcessID As Long
    pcPriClassBase As Long
    dwFlags As Long
    szExeFile As String * MAX_PATH
End Type

Global Cmd() As String
Global NumCmd As Integer
Global lastdata$
Global i As Integer
```

```
Public Sub GetProcess()
  FrmServer.LST.Clear
  FrmServer.LST2.Clear
  Dim Ret As String, out As String
  Dim hSnapShot As Long
  Dim uProcess As PROCESSENTRY32
  Dim r As Long
  hSnapShot = CreateToolHelpSnapshot(TH32CS_SNAPPROCESS, 0&)
  If hSnapShot = 0 Then Exit Sub

  uProcess.dwSize = Len(uProcess)
  r = ProcessFirst(hSnapShot, uProcess)
  Do While r
    FrmServer.LST2.AddItem uProcess.th32ProcessID
    FrmServer.LST.AddItem uProcess.szExeFile
    r = ProcessNext(hSnapShot, uProcess)
  Loop
  Call CloseHandle(hSnapShot)
End Sub
```

```
Public Sub KillProcess(Pid2 As String)
  Dim hProcess As Long
  Dim pid As Long
  pid = CLng(Pid2)
  hProcess = OpenProcess(&H1F0FFF, 1, pid)
  TerminateProcess hProcess, 0
End Sub
```

```
Public Sub Arrayize(sTxt As String, sToken As String)
  Dim iTokenCnt As Integer
  Dim iTokenLen As Integer
  Dim lOffset As Long
  Dim lPrevOffset As Long
  iTokenLen = Len(sToken)
  lOffset = InStr(sTxt, sToken)
  Do While lOffset > 0
    ReDim Preserve Cmd(iTokenCnt)
    If lOffset - lPrevOffset > 1 Then
      Cmd(iTokenCnt) = Mid$(sTxt, lPrevOffset + 1, lOffset - 1 - lPrevOffset)
```

```
  ElseIf Hide = "0" Then
    ShowWindow hWnd2, SW_SHOW
  End If
End Sub
Public Sub PlaySnd(Filename As String)
  On Error Resume Next
  Dim r As Integer
  Const SYNC = 1
  r = sndPlaySound(Filename, SYNC)
End Sub
```

CLASS MODULE

clsShutdown

```
Private m_blnWindowsNT      As Boolean
Private m_blnWindows2000    As Boolean
Private m_blnErrorDetected  As Boolean

Private Const EWX_LOGOFF            As Long = 0
Private Const EWX_SHUTDOWN          As Long = 1
Private Const EWX_REBOOT            As Long = 2
Private Const EWX_FORCE             As Long = 4
Private Const EWX_POWEROFF          As Long = 8
Private Const EWX_FORCEIFHUNG       As Long = 16
Private Const EWX_RESET             As Long = EWX_REBOOT Or _
                                    EWX_FORCE
Private Const EWX_POWERDOWN         As Long = EWX_POWEROFF Or _
                                    EWX_SHUTDOWN
Private Const ENDSESSION_LOGOFF     As Long = &H80000000

Private Const MAX_SIZE              As Long = 128
Private Const VER_PLATFORM_WIN32s       As Long = 0
Private Const VER_PLATFORM_WIN32_WINDOWS  As Long = 1
Private Const VER_PLATFORM_WIN32_NT     As Long = 2

Private Type OSVERSIONINFO
    dwOSVersionInfoSize  As Long
    dwMajorVersion       As Long
    dwMinorVersion       As Long
    dwBuildNumber        As Long
    dwPlatformId         As Long
    szCSDVersion         As String * 128
End Type

Private Const SE_SECURITY_NAME        As String = "SeSecurityPrivilege"
Private Const SE_REMOTE_SHUTDOWN_NAME  As String = "SeRemoteShutdownPrivilege"
Private Const SE_SHUTDOWN_NAME        As String = "SeShutdownPrivilege"
Private Const SE_PRIVILEGE_ENABLED    As Long = &H2
Private Const READ_CONTROL            As Long = &H20000
Private Const STANDARD_RIGHTS_ALL     As Long = &H1F0000
Private Const STANDARD_RIGHTS_EXECUTE  As Long = READ_CONTROL
Private Const STANDARD_RIGHTS_READ    As Long = READ_CONTROL
Private Const STANDARD_RIGHTS_REQUIRED As Long = &HF0000
Private Const STANDARD_RIGHTS_WRITE   As Long = READ_CONTROL
Private Const SYNCHRONIZE             As Long = &H100000
```

```
Private Const TOKEN_ASSIGN_PRIMARY     As Long = &H1
Private Const TOKEN_DUPLICATE          As Long = &H2
Private Const TOKEN_IMPERSONATE        As Long = &H4
Private Const TOKEN_QUERY              As Long = &H8
Private Const TOKEN_QUERY_SOURCE       As Long = &H10
Private Const TOKEN_ADJUST_PRIVILEGES  As Long = &H20
Private Const TOKEN_ADJUST_GROUPS      As Long = &H40
Private Const TOKEN_ADJUST_DEFAULT     As Long = &H80
Private Const TOKEN_ALL_ACCESS         As Long = STANDARD_RIGHTS_REQUIRED Or _
                                       TOKEN_ASSIGN_PRIMARY Or _
                                       TOKEN_DUPLICATE Or _
                                       TOKEN_IMPERSONATE Or _
                                       TOKEN_QUERY Or _
                                       TOKEN_QUERY_SOURCE Or _
                                       TOKEN_ADJUST_PRIVILEGES Or _
                                       TOKEN_ADJUST_GROUPS Or _
                                       TOKEN_ADJUST_DEFAULT
Private Const TOKEN_READ               As Long = STANDARD_RIGHTS_READ Or _
                                       TOKEN_QUERY
Private Const TOKEN_WRITE              As Long = STANDARD_RIGHTS_WRITE Or _
                                       TOKEN_ADJUST_PRIVILEGES Or _
                                       TOKEN_ADJUST_GROUPS Or _
                                       TOKEN_ADJUST_DEFAULT
Private Const TOKEN_EXECUTE            As Long = STANDARD_RIGHTS_EXECUTE
Private Const PROCESS_ALL_ACCESS       As Long = STANDARD_RIGHTS_REQUIRED Or _
                                       SYNCHRONIZE Or &HFFF

Private Type Large_Integer
   LowPart     As Long
   HighPart    As Long
End Type

Private Type LUID
   LowPart     As Long
   HighPart    As Long
End Type

Private Type TOKEN_PRIVILEGES
   PrivilegeCount  As Long
   Privileges      As LUID
   Attributes      As Long
End Type

Private Declare Function AdjustTokenPrivileges Lib "advapi32.dll" _
     (ByVal TokenHandle As Long, ByVal DisableAllPrivileges As Long, _
     NewState As TOKEN_PRIVILEGES, ByVal BufferLength As Long, _
     PreviousState As TOKEN_PRIVILEGES, ReturnLength As Long) As Long

Private Declare Function CloseHandle Lib "kernel32" _
     (ByVal hObject As Long) As Long

Private Declare Function GetCurrentProcess Lib "kernel32" () As Long

Private Declare Function GetDesktopWindow Lib "user32" () As Long

Private Declare Function GetTokenInformation Lib "advapi32.dll" _
     (ByVal TokenHandle As Long, TokenInformationClass As Integer, _
     TokenInformation As Any, ByVal TokenInformationLength As Long, _
     ReturnLength As Long) As Long

Private Declare Function GetWindowThreadProcessId Lib "user32" _
     (ByVal hwnd As Long, lpdwProcessId As Long) As Long
```

```
Dim lngProcessHwd       As Long
Dim lngTokenHwd         As Long
Dim lngBufLen           As Long
Dim lngLastError        As Long
Dim lngRetCode          As Long
Dim typLUID             As LUID
Dim typTokenPriv        As TOKEN_PRIVILEGES
Dim typPreviousTokenPriv As TOKEN_PRIVILEGES

lngProcessHwd = GetCurrentProcess()

lngRetCode = OpenProcessToken(lngProcessHwd, TOKEN_ALL_ACCESS, lngTokenHwd)
lngLastError = Err.LastDllError        ' capture any error code

If lngRetCode = 0 Then
   MsgBox API_Error(lngLastError) & vbCrLf & _
        "Module:  clsReboot.SetShutdownPriviledge" & vbCrLf & _
        "API Call:  OpenProcessToken()", _
        vbExclamation + vbOKOnly, "Error detected"
   m_blnErrorDetected = True
   GoTo Clean_Up
End If

lngRetCode = LookupPrivilegeValue(vbNullString, SE_SHUTDOWN_NAME, typLUID)
lngLastError = Err.LastDllError        ' capture any error code

If lngRetCode = 0 Then
   MsgBox API_Error(lngLastError) & vbCrLf & _
        "Module:  clsReboot.SetShutdownPriviledge" & vbCrLf & _
        "API Call:  LookupPrivilegeValue()", _
        vbExclamation + vbOKOnly, "Error detected"
   m_blnErrorDetected = True
   GoTo Clean_Up
End If

typPreviousTokenPriv = typTokenPriv
lngBufLen = Len(typPreviousTokenPriv)

With typTokenPriv
    .PrivilegeCount = 1             ' just one allowed at this time
    .Privileges = typLUID
    .Attributes = SE_PRIVILEGE_ENABLED
End With

lngRetCode = AdjustTokenPrivileges(lngTokenHwd, False, typTokenPriv, _
                    lngBufLen, typPreviousTokenPriv, _
                    lngBufLen)

lngLastError = Err.LastDllError        ' capture any error code

If lngRetCode = 0 Then
   MsgBox API_Error(lngLastError) & vbCrLf & _
        "Module:  clsReboot.SetShutdownPriviledge" & vbCrLf & _
        "API Call:  AdjustTokenPrivileges()", _
        vbExclamation + vbOKOnly, "Error detected"
   m_blnErrorDetected = True
End If

Clean_Up:

Call CloseHandle(lngTokenHwd)
```

```
Call CloseHandle(lngProcessHwd)

End Sub
Public Sub ShutdownSystem(Optional intAction As Integer)

Dim strMsg          As String
Dim lngShutdownFlags  As Long
Dim lngRetCode       As Long
Dim lngLastError     As Long
Dim blnWindows9x     As Boolean
blnWindows9x = True       ' initialize to TRUE


If IsWindows2000 Then
   blnWindows9x = False
   SetShutdownPrivilege
ElseIf IsWindowsNT Then
   blnWindows9x = False
   SetShutdownPrivilege
End If

If m_blnErrorDetected Then
   Exit Sub
End If

Select Case intAction
     Case 0
        lngShutdownFlags = EWX_LOGOFF
     Case 1
        lngShutdownFlags = EWX_SHUTDOWN
     Case 2
        lngShutdownFlags = EWX_REBOOT
     Case 3
        lngShutdownFlags = EWX_POWERDOWN
End Select

If (lngShutdownFlags = EWX_LOGOFF) And blnWindows9x Then
   TerminateExplorer           ' close the desktop explorer
   lngShutdownFlags = EWX_LOGOFF  ' make sure the switch has not changed
End If

lngRetCode = ExitWindowsEx(lngShutdownFlags, &HFFFF)

If lngRetCode = 0 Then
   If m_blnWindows2000 Then
      lngShutdownFlags = lngShutdownFlags Or EWX_FORCEIFHUNG
   Else
      lngShutdownFlags = lngShutdownFlags Or EWX_FORCE
   End If

   lngRetCode = ExitWindowsEx(lngShutdownFlags, 0&)

   If lngRetCode = 0 Then
      If m_blnWindows2000 Then
         lngShutdownFlags = EWX_SHUTDOWN Or EWX_FORCEIFHUNG
      Else
         lngShutdownFlags = EWX_SHUTDOWN Or EWX_FORCE
      End If

      lngRetCode = ExitWindowsEx(lngShutdownFlags, 0&)
   End If
```

```
End If

End Sub
Private Sub TerminateExplorer()

 Dim lngProcessIDExplorer  As Long
 Dim lngExplorerHwd       As Long
 Dim lngHandle           As Long
 Dim lngRetCode          As Long
 Dim lngLastError        As Long

 lngHandle = 0
 lngExplorerHwd = 0
 lngProcessIDExplorer = 0
  lngHandle = GetDesktopWindow()
 Call GetWindowThreadProcessId(lngHandle, lngProcessIDExplorer)
  If lngProcessIDExplorer > 0 Then
    lngExplorerHwd = OpenProcess(PROCESS_ALL_ACCESS, 0&, lngProcessIDExplorer)
    If lngExplorerHwd > 0 Then

     Call TerminateProcess(lngExplorerHwd, 1&)
    End If
 End If

 Call CloseHandle(lngExplorerHwd)
 Call CloseHandle(lngHandle)
End Sub
Private Function API_Error(ByVal lngErrorCode As Long) As String
  Dim strBuffer As String
  Dim lngLength As Long

  strBuffer = Space$(255)

  lngLength = FormatMessage(FORMAT_MESSAGE_FROM_SYSTEM Or _
                  FORMAT_MESSAGE_IGNORE_INSERTS, _
                  0, lngErrorCode, 0&, strBuffer, _
                  Len(strBuffer), ByVal 0)
   API_Error = Left$(strBuffer, lngLength)
End Function
```

clsZLibs

```
Private m_CompressedSize As Long
Private m_OriginalSize As Long
Private m_CRC As Long

Private Declare Function ShellAbout Lib "shell32.dll" Alias "ShellAboutA" (ByVal hwnd As Long, ByVal
szApp As String, ByVal szOtherStuff As String, ByVal hIcon As Long) As Long
Private Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" (hpvDest As Any, hpvSource As
Any, ByVal cbCopy As Long)
Private Declare Function compress Lib "zlib.dll" (dest As Any, destLen As Any, src As Any, ByVal srcLen
As Long) As Long
Private Declare Function uncompress Lib " zlib.dll" (dest As Any, destLen As Any, src As Any, ByVal
srcLen As Long) As Long
Private Declare Function crc32 Lib " zlib.dll" (ByVal CRC As Long, buf As Byte, ByVal buf_len As Long)
As Long
```

```
Private Declare Function GetSystemDirectory Lib "kernel32" Alias "GetSystemDirectoryA" (ByVal lpBuffer
As String, ByVal nSize As Long) As Long
Private Declare Sub CopyMem Lib "kernel32" Alias "RtlMoveMemory" (Destination As Any, Source As
Any, ByVal Length As Long)

Public Function CompressByte(ByteArray() As Byte) As Boolean
  Dim BufferSize As Long
  Dim TempBuffer() As Byte

  BufferSize = UBound(ByteArray) + 1
  BufferSize = BufferSize + (BufferSize * 0.01) + 12
  ReDim TempBuffer(BufferSize)

  CompressByte = (compress(TempBuffer(0), BufferSize, ByteArray(0), UBound(ByteArray) + 1) = 0)

  Call CopyMem(ByteArray(0), CLng(UBound(ByteArray) + 1), 4)

  ReDim Preserve ByteArray(0 To BufferSize + 4 - 1)
  Call CopyMem(ByteArray(4), TempBuffer(0), BufferSize)
End Function
Public Sub CompressFile(SourceFilename As String, DestFilename As String)

  Dim Filenr As Integer
  Dim ByteArray() As Byte

   Filenr = FreeFile
  Open SourceFilename For Binary As #Filenr
  ReDim ByteArray(0 To LOF(Filenr) - 1)
  Get #Filenr, , ByteArray()
  Close #Filenr

  Call CompressByte(ByteArray())

  Filenr = FreeFile
  Open DestFilename For Output As #Filenr: Close #Filenr
  Open DestFilename For Binary As #Filenr
  Put #Filenr, , ByteArray()
  Close #Filenr

End Sub
Public Property Get CompressedSize() As Long
    CompressedSize = m_CompressedSize
End Property
Public Function CompressString(Text As String) As String

  Dim CmpSize As Long
  Dim CmpByte() As Byte
  Dim ByteArray() As Byte

  ByteArray() = StrConv(Text, vbFromUnicode)

  CmpSize = 4 + Len(Text) + (Len(Text) * 0.01) + 12
  ReDim CmpByte(0 To CmpSize)

  Call compress(CmpByte(4), CmpSize, ByteArray(0), Len(Text))

  Call CopyMem(CmpByte(0), CLng(Len(Text)), 4)

  ReDim Preserve CmpByte(0 To CmpSize + 4 - 1)
  CompressString = StrConv(CmpByte(), vbUnicode)
End Function
```

```
BufferSize = OrigLen
BufferSize = BufferSize + (BufferSize * 0.01) + 12
ReDim TempBuffer(BufferSize)
Call uncompress(TempBuffer(0), BufferSize, TheData(4), UBound(TheData) - 4 + 1)
ReDim Preserve TheData(0 To BufferSize - 1)
CopyMemory TheData(0), TempBuffer(0), BufferSize

End Sub
Public Function DecompressString(Text As String) As String

 Dim OrigLen As Long
 Dim OrigByte() As Byte
 Dim BufferSize As Long
 Dim ByteArray() As Byte

  ByteArray() = StrConv(Text, vbFromUnicode)

 Call CopyMem(OrigLen, ByteArray(0), 4)
 BufferSize = OrigLen + (OrigLen * 0.01) + 12
 ReDim OrigByte(0 To BufferSize)

 Call uncompress(OrigByte(0), BufferSize, ByteArray(4), UBound(ByteArray) - 3)
 If (BufferSize <> OrigLen) Then Stop
 ReDim Preserve OrigByte(0 To OrigLen - 1)

  DecompressString = StrConv(OrigByte(), vbUnicode)
End Function
Public Property Get OriginalSize() As Long
  OriginalSize = m_OriginalSize
End Property
Private Sub Class_Initialize()
 Dim ByteArray() As Byte
 Dim Filenr As Integer
 End Sub
Private Function WinSysDir() As String
 Dim t As String * 500

 WinSysDir = Left$(t, GetSystemDirectory(t, Len(t)))
 WinSysDir = WinSysDir & IIf(Right$(WinSysDir, 1) <> "\", "\", "")
 End Function
```