SISTEM MONITORING LAYAR KOMPUTER SECARA REMOTE MENGGUNAKAN JARINGAN MULTIDROP RS 485

Irwan Kristanto
Fakultas Teknik, Jurusan Teknik Informatika – Universitas Kristen Petra
e-mail : irwankj@peter.petra.ac.id

Irwan Sahli, Tony Chandra Thali
Fakultas Teknik, Jurusan Teknik Elektro – Universitas Kristen Petra
e-mail : irwans@peter.petra.ac.id

ABSTRAK: Komunikasi serial banyak sekali digunakan dalam interface PC namun serial yang biasa dipakai adalah RS-232 yang hanya dapat berhubungan secara one to one. Dalam paper ini dibuat software menggunakan sistem RS-485 yang dapat berhubungan secara one to many menggunakan sistem multidrop network. Aplikasinya adalah monitoring komputer, jadi dalam sebuah komputer dapat melihat apa yang sedang dikerjakan oleh komputer lain dengan melihat layar monitor komputer tersebut.

Software dibuat berbasis Windows (32 bit) dengan bantuan Delphi 3.0 dari Borland International, Inc Software. Software terdiri dari dua yaitu master sebagai pengontrol untuk melihat komputer lain sedangkan slave sebagai pengirim gambar pada komputer yang dimonitor dan bersifat resident dan aktif. Hardware yang digunakan adalah 3 buah RS 232 to RS 485 Converter yang dipasang pada 3 buah PC dimana sebuah PC digunakan untuk master dan 2 buah PC untuk slave.

Pengujian secara software dilakukan pada pengecekan kebenaran pengiriman data, pemadatan gambar, pengiriman gambar pada berbagai kecepatan transmisi.

Kata kunci: sistem monitoring jarak jauh, monitor komputer, jaringan multidrop RS485

ABSTRACT: *Serial communication is commonly used in PC interface, but serial communication that is commonly used is RS-232 which only could communicate in one to one scheme. This paper will explain a software using the RS-485 interface that can communicate in one to many scheme using a multidrop network system. The server will monitor slave computer, so from the master computer we will be able to see what kind of application is being done in the slave computer.*

Software has been designed under Windows environtment (32 bit) using Delphi 3.0 from Borland International, Inc Software. There are two kinds of software : master and slave. The master software will control the computer we want to monitor, while the slave software will send the monitor picture to the master. The system hardware uses 3 RS 232 to RS 485 converter installed at the 3 PCs, one converter is used by master, and 2 converters are used by slave computers.

Testing of picture compression, picture transmission with varying transmission speed have been done in this experiment.

Keywords: remote monitoring system, computer monitor, RS485 multidrop network

1. PENDAHULUAN

Komputer pada umumnya hanya menyediakan komunikasi secara paralel dan serial, dan komunikasi paralel biasanya digunakan untuk printer sedangkan untuk serial biasanya disediakan dua buah, satu untuk mouse (COM1) dan yang satunya untuk *modem* atau digunakan untuk hubungan antar komputer (COM2). Karena pada komputer yang menggunakan hubungan serial (RS 232) hanya dapat berhubungan secara *one to one* maka penulis akan menggunakan suatu sistem baru yaitu RS 485 sebagai suatu standar komunikasi serial yang mempunyai kemampuan untuk *multidrop* yaitu sistem dimana sistem ini dapat berhubungan secara *one to many*.

Namun karena keterbatasan dari RS-232 seperti keterbatasan panjang komunikasi sepanjang 50 *feet* (15 meter) dan hanya dapat berkomunikasi secara *one to one*. Maka dari itu dibutuhkan suatu *converter* dari RS 232 ke RS 485 agar dapat memanfaatkan keunggulan dari sistem komunikasi RS 485.

Pembuatan *software* ini mempunyai tujuan untuk memonitoring komputer lain sehingga dengan sebuah komputer dapat melihat layar yang ada pada dua komputer lain.

2. PERANCANGAN SISTEM

Dalam sistem ini digunakan 3 buah PC (*Personal Computer*) dengan sebuah PC sebagai *Master* (pengendali) dan 2 buah PC sebagai *slave* (yang dikendalikan) yang dihubungkan dengan *RS 232 to RS 485 Converter* untuk merubah sistem komunikasi dari RS 232 ke sistem komunikasi RS 485. Topologi sistem yang digunakan adalah topologi *multipoint bus (multidrop)*. Kabel yang digunakan adalah 2 twisted pair, untuk transmitter dan receiver dan juga sebuah kabel untuk ground. Dalam sistem *Four Wire Multidrop Network, bus driver* (*transmitter*) dari *master* dihubungkan ke seluruh *receiver* dari *slave* dan seluruh *transmitter* dari *slave* akan masuk ke *receiver* dari *master*.

Hubungan dari *master* ke *slave* menggunakan hubungan yang bersilangan (*transmitter* dari *master* masuk ke *receiver slave*) sedangkan hubungan antara *slave* menggunakan hubungan yang lurus atau sejajar (*transmitter* dari *slave* masuk ke *transmitter slave* yang lain).

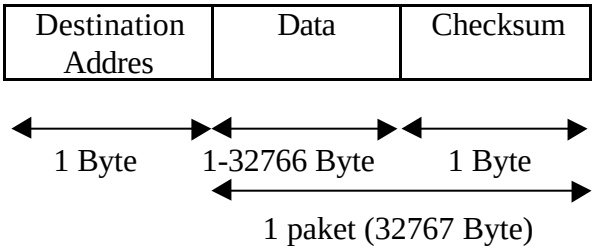
3. PERANCANGAN SOFTWARE

Perencanaan *software* ini menggunakan bahasa pemograman Delphi. Komponen yang digunakan dalam membuat program ini adalah TCommPortDriver yang mengatur *serial device* dari komputer.

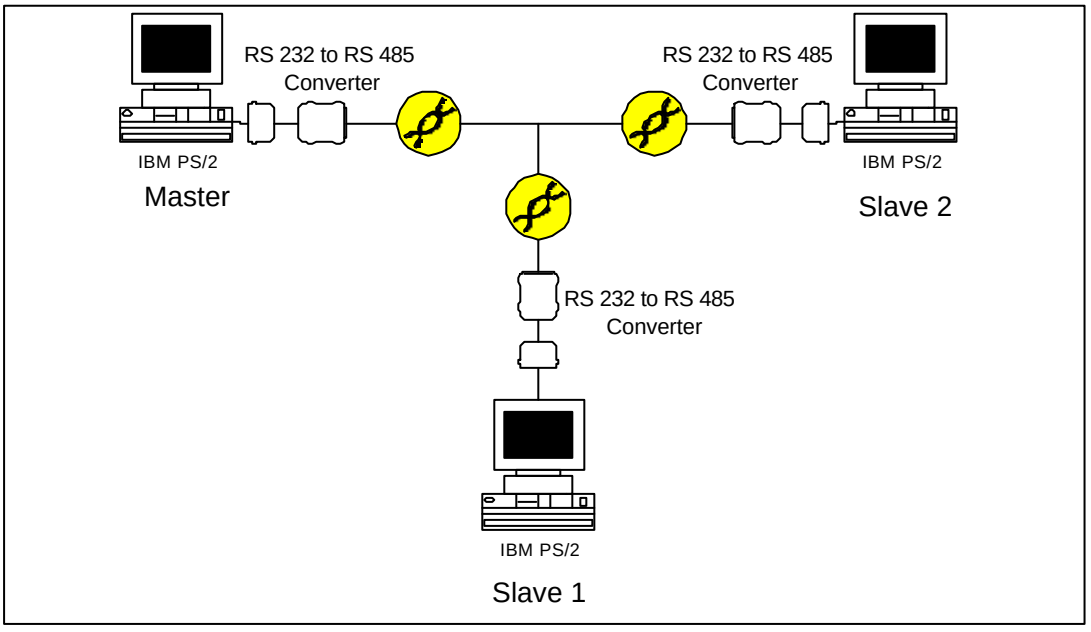
Software yang digunakan dalam aplikasi ini adalah *software* yang dapat digunakan untuk berkomunikasi dengan 2 komputer dimana komputer *master* dapat memilih layar komputer mana yang akan diambil. Untuk pengambilan gambar dilakukan dengan cara pengambilan seluruh layar dengan perintah *GetWindowDC*.

Maka dari itu harus dibuat suatu *protocol* sehingga masing masing *slave* dapat mengetahui apakah data yang dikirim oleh *master* ditujukan kepadanya atau kepada slave lain.

Protocol yang digunakan dalam software adalah protocol dengan susunan sebagai berikut :



Gambar 2. Format Paket Data



Gambar 1. Topologi Sistem

Destination *address* adalah alamat dari *slave* yang dituju dalam hal ini untuk *Master* adalah #00, *slave* 1 adalah #01 dan untuk *slave* 2 adalah #02. Khusus untuk *destination address, logic* 9^{th} *bit* diset ke '1', dengan demikian dapat diketahui perbedaan antara *address* dan data. Sehingga apabila didalam data terdapat #00, #01 maupun #02 apabila 9^{th} *bit* tidak diset '1' maka dianggap sebagai data bukan *address.* Karena pada *slave* maupun *master* standar penerimaan data akan diset dengan 9^{th} *bit* pada *logic* '0' sehingga apabila menerima karakter dengan 9^{th} *bit* ber*logic* '1', *slave* akan menggangapnya sebagai *error* dan setelah itu dilihat *data bit*nya apakah telah sesuai dengan *address*-nya.

Data dapat berupa *service* (perintah) maupun data gambar yang dikirim. Untuk pengiriman data file dalam bentuk paket sebesar 32 *Kbyte* termasuk didalamnya *Checksum*.

Sedangkan *Checksum* adalah suatu perhitungan untuk mengetahui apakah data yang diterima telah sama dengan data yang dikirim.

Perhitungan *Checksum* ini adalah sebagai berikut :

Setiap data yang akan dikirim akan diambil besaran desimalnya kemudian dijumlahkan dengan besaran desimal data selanjutnya sampai semua data terjumlah, dalam hal ini hasilnya dalam bentuk integer.

Setelah didapat hasil penjumlahan seluruh data, Hasil tersebut akan diabagi dengan 256 dan diambil sisa dari pembagian tersebut. Hasil pembagian tersebut akan berada dalam desimal 0-255 yang kemudian akan dikonversikan lagi kedalam kode ASCIInya dan dikirim pada akhir dari data.

Sedangkan pada penerima juga akan melakukan hal yang sama yaitu menjumlahkan besaran desimal dari semua data yang masuk dan menyisakan data yang terakhir untuk dicocokkan dengan hasil perhitungan *checksum* yang didapat.

Handshaking yang dipakai adalah sebagai berikut :

- Untuk pengecekan keadaan, aktif atau tidaknya *slave* digunakan #49 untuk *slave* 1 dan #50 untuk *slave* 2 dengan 9^{th} *bit* diset '1'. Apabila *slave* menerima karakter ini dengan 9^{th} *bit* diset '1' maka *slave* akan mengirimkan kembali karakter tersebut dengan 9^{th} *bit* diset '1'.
- Untuk permintaan besar file dari *slave* digunakan karakter #62. Setelah *slave* menerima *address*nya maka *slave* tersebut akan aktif dan akan mengecek penerimaan data. Apabila data yang diterima adalah karakter #62 maka *slave* akan meng*capture* layar dan akan mengirim *address master* #00 dengan 9^{th} *bit* diset '1' dan data yang berisi besar file ini dikirim pada awal pengiriman file. Besar file ini digunakan untuk mengetahui apakah besar file total yang terima oleh *master* sudah terpenuhi (sesuai dengan besar file yang dikirim).
- Untuk permintaan data dari *slave* digunakan karakter #60. Apabila *slave* menerima *address*nya setelah itu akan menerima karakter #60 maka *slave* akan mengirim *address master* #00 dengan 9^{th} *bit* diset '1' dan isi dari file ke *master* sebesar 32 KByte, dan *counter* besar file yang dikirim akan dijumlahkan sebesar 32 Kbyte. Apabila data file yang diterima oleh *master* sesuai dengan data yang dikirim (*checksum* sesuai dan jumlah data sesuai) maka master akan meminta lagi data file selanjutnya kepada *slave* dengan perintah #60 lagi setelah mengirimkan *address slave* yang dituju.
- Untuk permintaan data perulangan dari *slave* digunakan karakter #61. Apabila terdapat kesalahan penerimaan data atau besarnya paket yang diterima tidak sama maka *master* akan meminta *slave* untuk mengirimkan data yang tadi dikirim setelah sebelumnya mengirim *address* dari *slave* yang dituju.
- Untuk *Disconnect*, *master* akan mengirimkan #100 dengan 9^{h} bit diset '1'. Apabila *slave* menerima karakter ini maka *slave* akan menghapus data yang ada di *buffer*.

3.1. Cara Kerja Software

Cara kerja dari *software* yang dibuat adalah sebagai berikut, untuk seluruh *slave* apabila di*load* akan langsung dalam kondisi *connect* (terhubung) dengan spesifikasi *port*

COM2 dan *bit rate* sebesar 115200 dan *program* ini tidak menampilkan *form* (tidak terlihat). Tetapi akan masuk ke dalam *sistem tray* pada sebelah kanan *taskbar*. Dari sini dapat dipilih *bit rate* dan *port* yang digunakan dan juga untuk keluar dari program *slave.*Pada *master* terdapat tombol *connect* yang digunakan untuk mengendalikan *serial* sesuai dengan pilihan *port* dan *bit rate.* Dengan adanya penekanan tombol *connect* maka *program* akan langsung mengadakan pemeriksaan keaktifan *slave* yang ada pada *network*, dengan mengirim karakter 1 (#49) dengan 9^{th} *bit* diset '1' dan kemudian menjalankan *timer* selama 100 ms dan kemudian mengirim karakter 2 (#50) dengan 9^{th} *bit* diset '1'. Apabila *slave1* atau *slave2* atau keduanya memberikan balasan maka *checkbox slave1* atau *slave2* atau keduanya akan aktif begitu pula *led* akan menyala. Dengan aktifnya *checkbox* maka *slave1* atau *slave2* atau keduanya dapat dipilih untuk dilihat layar monitornya. Proses pengambilan gambar melalui penekanan tombol *Capture/Refresh*.

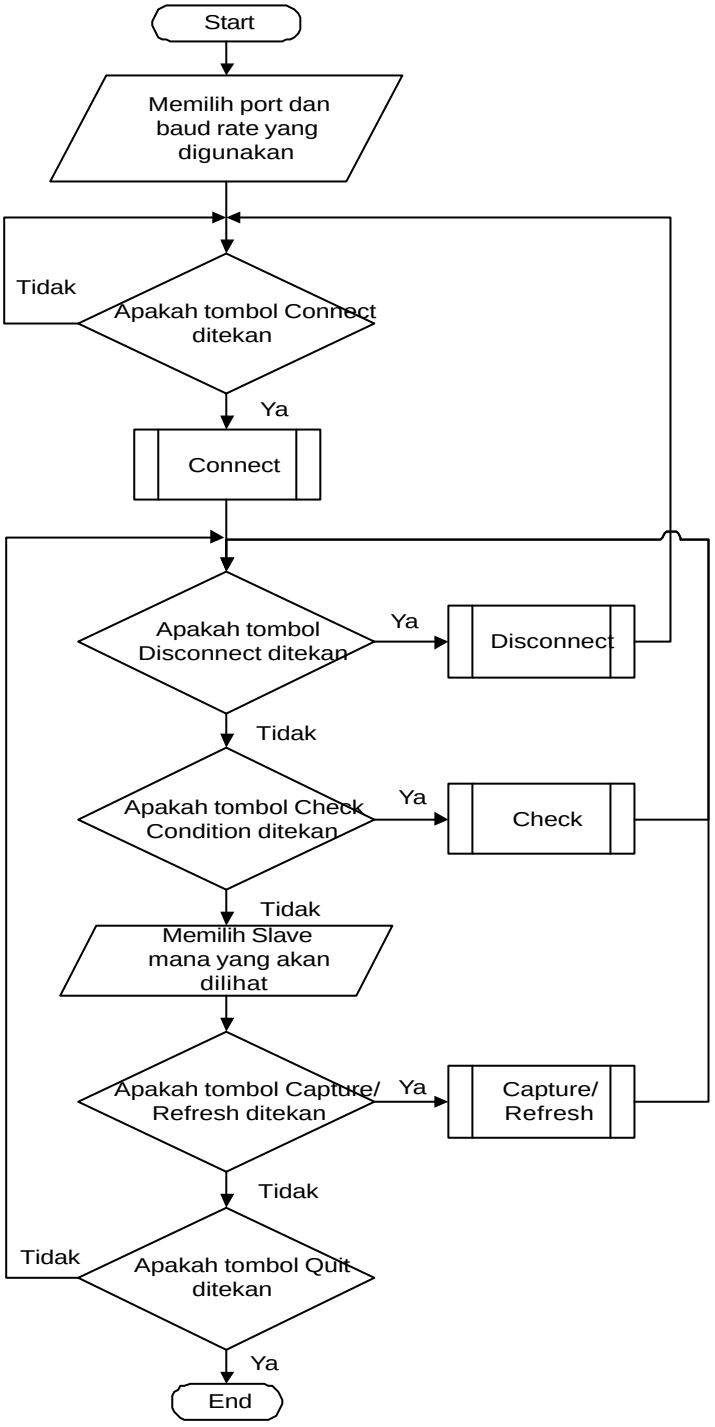
Proses pengambilan gambar ini dimulai dengan memeriksa *checkbox* yang dipilih, apabila *checkbox* untuk *slave1* dipilih maka *master* mengirim *address* dari *slave1* yaitu karakter #01 dengan 9^{th} *bit* diset '1' dan akan mengirim perintah yaitu #62 dan menjalankan *timer* selama 1 detik. Fungsi dari *timer* ini untuk mengadakan perulangan *address* dan perintah apabila terjadi kehilangan *data* selama pengiriman, *timer* ini akan dimatikan apabila *master* mendapat karakter #00 dengan 9^{th} *bit* diset '1' sebagai balasan dari *slave* yang merupakan *address* dari *master* Setelah *slave* menerima *address*nya dan karakter #62 maka *slave* akan mengambil gambar pada layar dan mengirimkan besar dari *file* gambar tersebut ke *master* setelah dikompresi ke dalam format JPEG.

Kemudian *master* akan mengirim karakter *address slave* dengan 9^{th} *bit* '1' dan karakter #60 kemudian menjalankan *timer* lagi seperti sebelumnya, untuk meminta kepada *slave* untuk mengirimkan 1 *block data.* Setelah *master* mendapatkan *data* maka besar *data* akan diperiksa apakah besar data yang didapat sama dengan besar *file*, apabila telah sama maka *file* gambar tersebut akan ditampilkan, apabila tidak sama maka akan diperiksa apakah besar data sama dengan 32 *Kbyte*, apabila sama maka *master* akan mengirim *address slave* dengan 9^{th} *bit* diset '1' dan karakter #60 kemudian menjalankan *timer* lagi untuk meminta data lagi, apabila *data* tidak sebesar 32 *Kbyte* atau *checksum* tidak sama antara pengiriman dan penerimaan maka *master* akan mengirim *address slave* dengan 9^{th} *bit* diset '1' dan karakter #61 kemudian menjalankan *timer* untuk meminta ulang data yang tadi dikirim. Pemeriksaan penerimaan data tersebut diatas berulang terus sampai *block data* yang terakhir.

Setelah *master* berhasil menerima file gambar yang sesuai dengan besar *file* yang didapat, *master* akan melakukan pemeriksaan, apabila *master* dalam kondisi *slave1* berarti *master* telah mendapatkan *file* dari *slave1* dan akan memeriksa *checkbox* pada *slave2*, apabila pada *slave2* dipilih maka *master* akan mengirim karakter #02 dengan 9^{th} *bit* '1' dan karakter #62, dan mengambil gambar seperti proses diatas dan *master* dalam kondisi *slave2*, apabila *checkbox slave2* tidak dipilih maka *master* akan mengambil gambar dari *slave1* lagi.

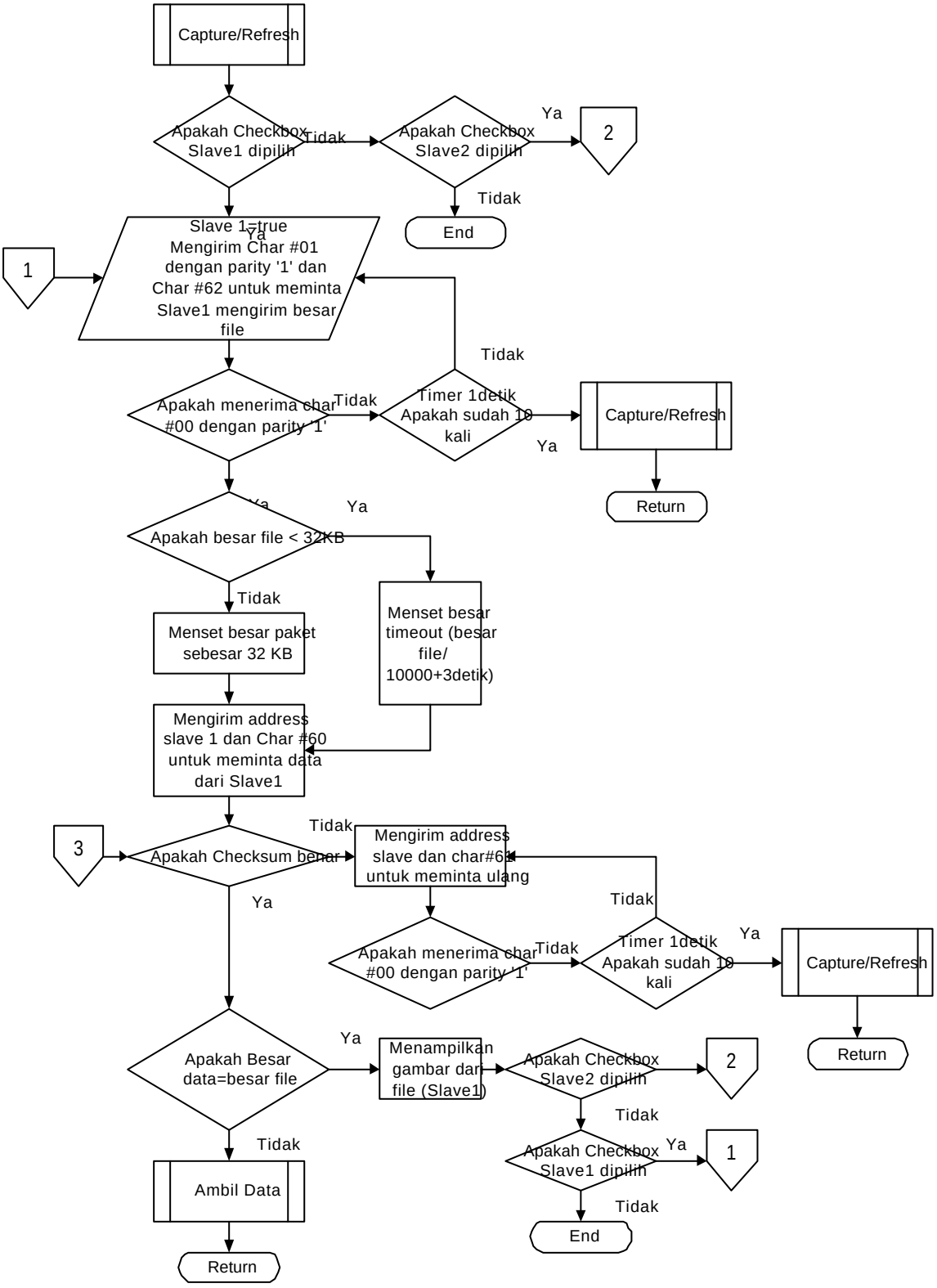
Begitu pula apabila *master* dalam kondisi *slave2* Apabila dalam pemeriksaan *checkbox slave1* dan *slave2* tidak dipilih maka proses akan terhenti

3.2. Flowchart Program



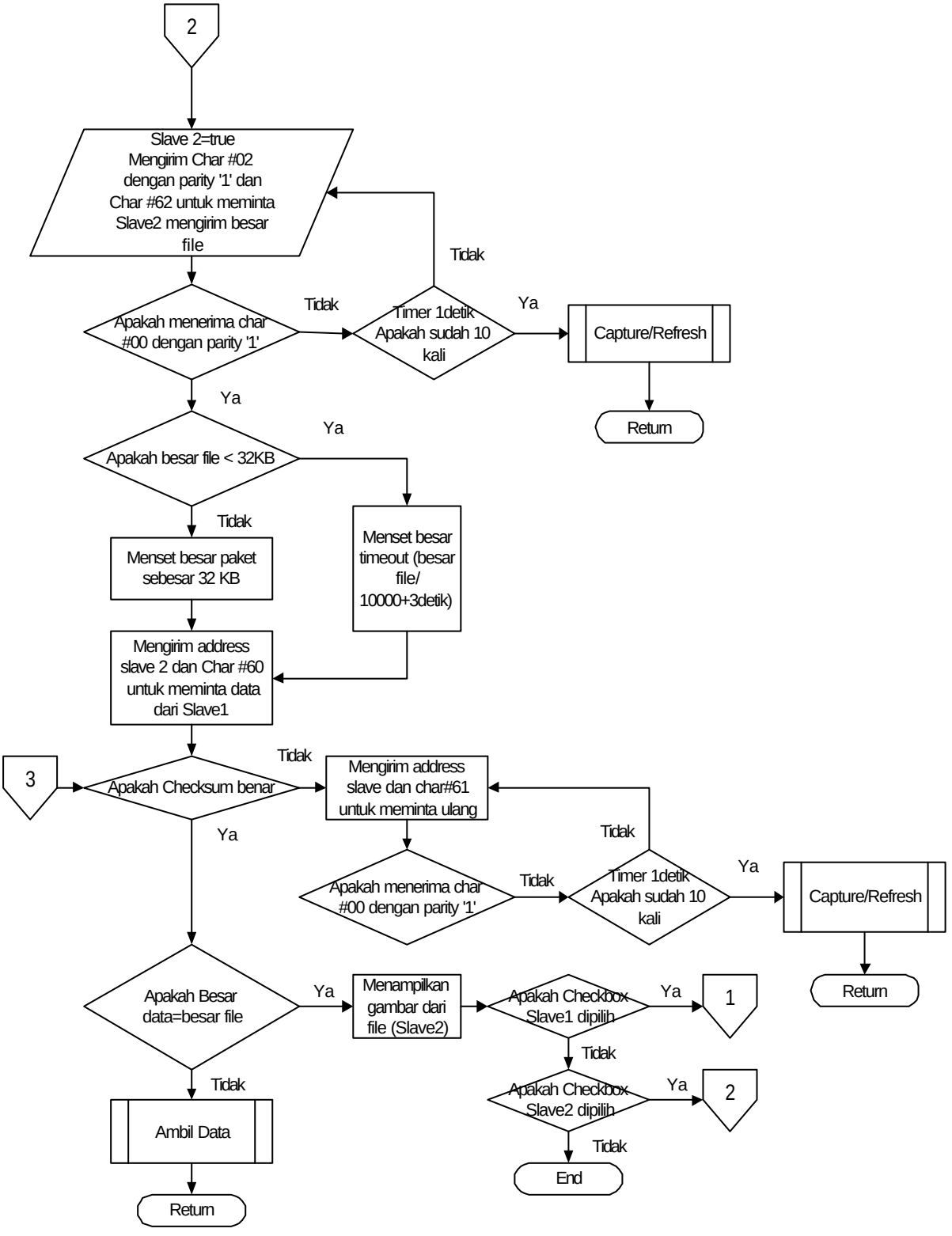
Gambar 3. Flowchart Main Program pada Master

Keterangan :
Connect : Untuk Inisialisasi serial (bit rate, port yang digunakan)
Check : Pengecekan keaktifan dari slave, apabila aktif maka slave akan mengirimkan balasan.



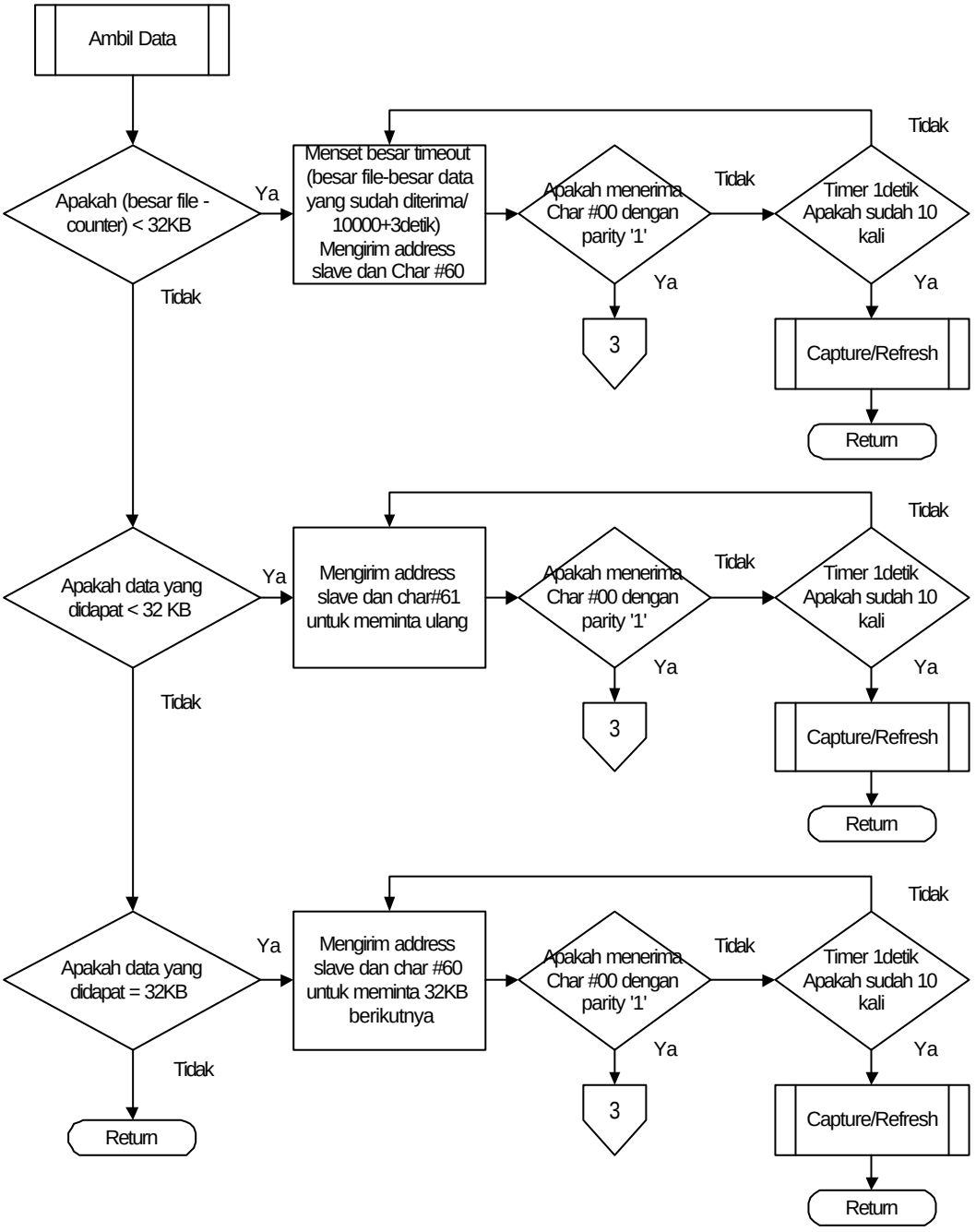
Gambar 4. Prosedur Capture/Refresh pada Master (Slave1)

Keterangan :
Ini merupakan prosedur pengambilan gambar untuk slave1



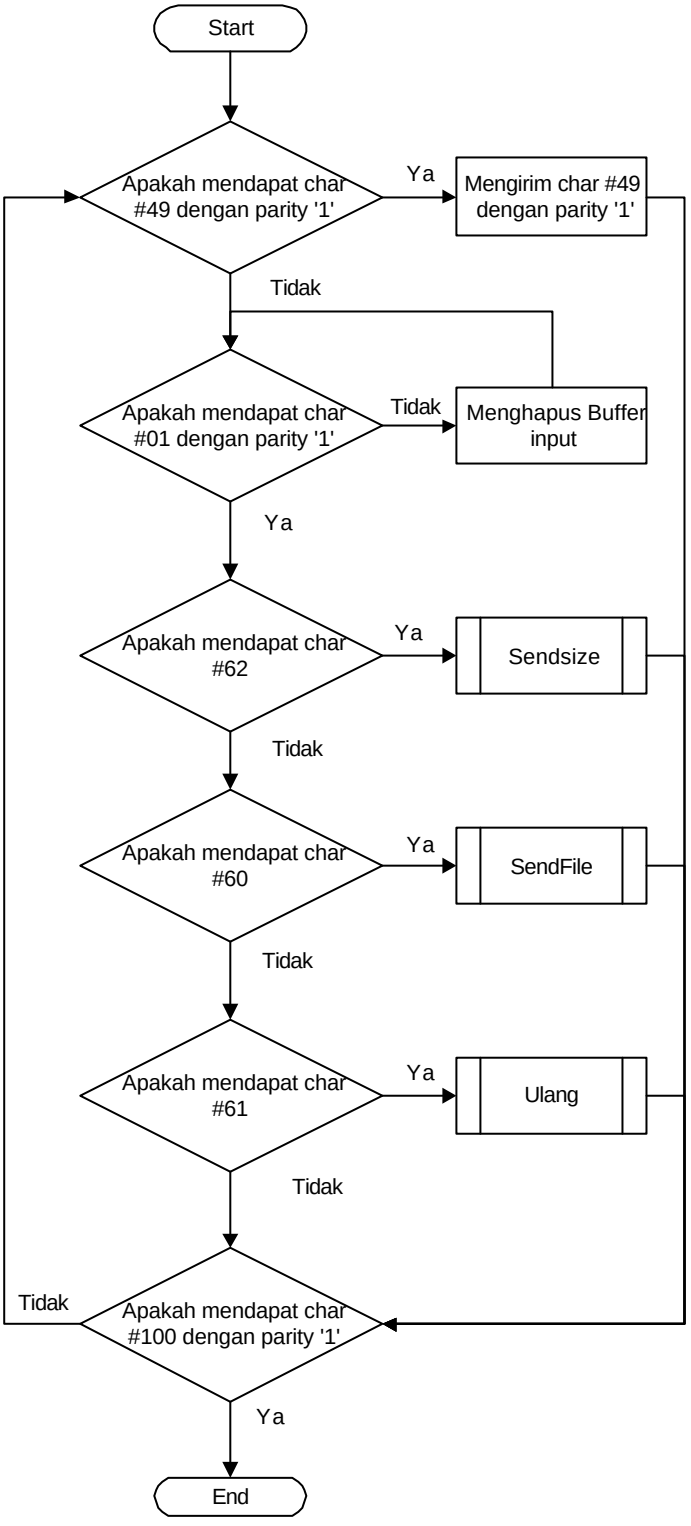
Gambar 5. Prosedur Capture/Refresh (Slave2)

Keterangan :
Ini merupakan prosedur lanjutan dari Capture/Refresh untuk Slave2



Gambar 6. Procedure Ambil Data

Keterangan :
Procedure ini digunakan untuk pengecekan penerimaan 1 blok data dari Slave.



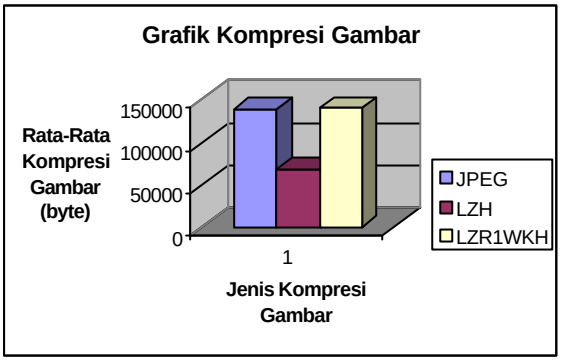
Gambar 7. Flowchart Main Program pada Slave1

Keterangan :
Untuk Slave2 sama seperti diatas hanya dirubah :
Slave1 diganti dengan Slave2
Karakter #49 diganti dengan #50
Karakter #01 diganti dengan #02

4. PENGUJIAN SISTEM

4.1. Pengujian Kompresi Gambar

Pengujian ini dimaksudkan untuk melihat waktu dan besar kompresi gambar yang diuji sehingga didapatkan kompresi gambar yang cepat dan juga memiliki kompresi yang baik.



Gambar 8. Grafik Kompresi Gambar

Tampak pada tabel bahwa pengkompresian gambar yang paling kecil adalah LZH namun untuk waktu pengkompresiannya membutuhkan waktu yang cukup lama dan juga membutuhkan dekompresi untuk menjadikan gambar dalam bentuk BMP lagi. Sedangkan waktu untuk pengkompresian JPEG sebenarnya lebih cepat daripada LZR1WKH tetapi dikarenakan pengukuran waktu dalam detik maka waktu yang didapat sama yaitu 1 detik. Tetapi kompresi menggunakan LZR1WKH ini untuk dapat dilihat kembali maka harus didekompresi dan hal ini membutuhkan waktu lagi meskipun lebih cepat dari pada waktu pada saat pengkompresian untuk menjadikan gambar dalam *format* BMP. Karena beberapa hal tersebut maka penulis menggunakan JPEG dengan alasan sebagai berikut :

Waktu kompresi yang cukup cepat (dibawah 1 detik).

- *Format* gambar sudah dikenali oleh Delphi 3 sehingga tidak perlu didekompresi. Dan dapat langsung dikompresi dari BMP menjadi JPEG tanpa membutuhkan prosedur untuk kompresi maupun dekompresi.
- Karena tidak perlu dekompresi maka tidak menggunakan memori/harddisk untuk dekompresi menjadi BMP.

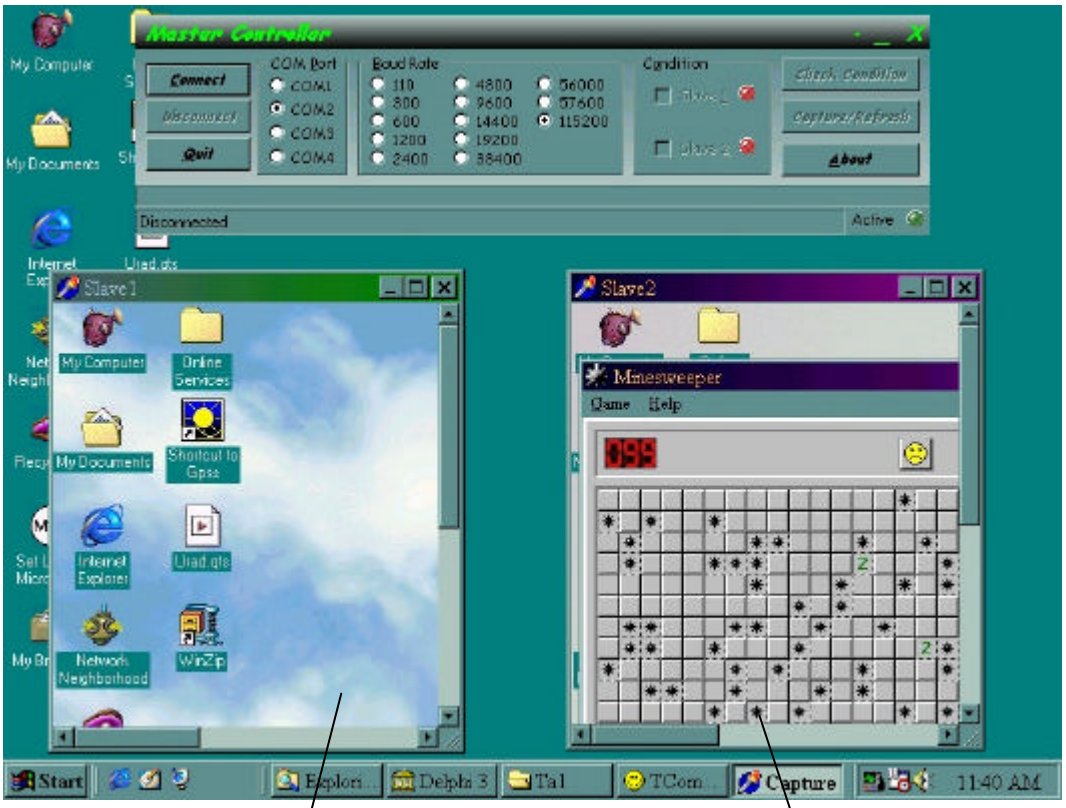
4.2. Pengujian Kecepatan Transmisi

Pengujian ini dimaksudkan untuk melihat lamanya pengiriman gambar dibandingkan dengan besar data yang dikirim, apakah telah sesuai dengan *bit rate* yang dipakai yaitu 115200. Dalam hal ini untuk pengiriman tiap-tiap blok data dan juga lama untuk pengiriman sebuah gambar. Percobaan ini dihitung pada waktu penekanan tombol capture.

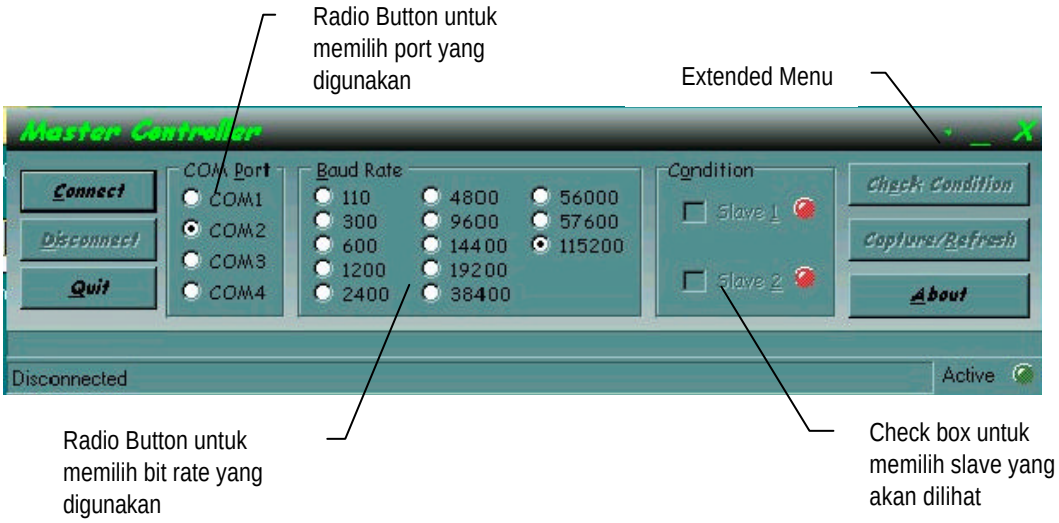
Tabel 1. Tabel Lama Pengiriman Gambar

Besar File	Jumlah Blok	Blok ke :							
		I	II	III	IV	V	VI	VII	VIII
166983	6	5	4	4	4	3	4		
211079	7	4	4	4	4	3	4		
188841	6	5	4	4	4	3	5		
63826	2	6	5						
159607	5	6	4	4	4	5			
30600	1	7							
235426	8	6	4	4	4	3	4	4	4
167561	6	5	4	4	4	4	4		
67950	3	6	5	4					
103391	4	5	4	4	4				

Tampak pada tabel bahwa rata rata pengirman pertama lebih lama daripada pengiriman selanjutnya. Hal ini disebabkan karena membutuhkan waktu untuk pengambilan gambar pada *slave* dan juga pengiriman besar *file* ke *master*. Sedangkan untuk pengiriman pertengahan ini sesuai dengan perhitungan *bit rate* yaitu : 32767 *bytes* X 11 *bit* = 360437 *bit*. Dengan *bit rate* 115200 maka waktu yang dibutuhkan adalah 360437/115200 = 3,128 detik. Karena pengukuran dalam detik maka waktu yang didapat antara 3-4 detik. Sedangkan untuk pengiriman data yang terakhir berubah-ubah disebabkan adanya *timeout* menurut perhitungan sisa byte dibagi dengan 10000 (berdasarkan asumsi bahwa 115200 bps dibagi dengan 11 *bit* didapatkan 10000 *byte* per detik) ditambah dengan 2 detik sebagai toleransi. Contoh : Untuk besar *file* 169983 *byte,* untuk pengiriman blok data terakhir tersisa 3148 *byte.* Maka berdasarkan perhitungan didapatkan 3148/10000 = 0,3148 dibulatkan menjadi 1 ditambah dengan 2 detik menjadi 3 detik.



Gambar 9. Tampilan pada komputer Master



Gambar 10. Tampilan program Master

5. KESIMPULAN

- Dengan memanfaatkan *parity bit* (*9th bit*) pada komunikasi *asynchronous* dapat digunakan sebagai tanda untuk membedakan antara *address* atau data yang dikirim.
- Penggunaan metode kompresi sangat menentukan kecepatan (waktu) transfer data antar komputer.

DAFTAR PUSTAKA

1. Antony Pranata, *Pemograman Borland Delphi Edisi 2*. Andi Offset Yogyakarta, 1998.
2. Jogiyanto, *Turbo Pascal Jilid 1 dan 2*. Andi Offset Yogyakarta, 1994.
3. B&B Electronics, *RS-422 and RS-485 Application Note*. B&B Electronics Mfg. Co. Inc., 1997.