SISTEM SINTESIS DAN ANALISIS RANGKAIAN ELEKTRONIKA SECARA TERPADU DENGAN MENGGUNAKAN INTERFACE BUS IEEE.488

TUGAS AKHIR

Disusun oleh :

I NENGAH WIDIANARTA

NRP . 2292.100.064

JURUSAN TEKNIK ELEKTRO
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA

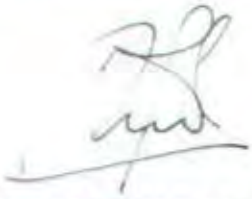
1998

SISTEM SINTESIS DAN ANALISIS RANGKAIAN ELEKTRONIKA SECARA TERPADU DENGAN MENGGUNAKAN INTERFACE BUS IEEE.488

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik Elektro
Pada
Bidang Studi Elektronika
Jurusan Teknik Elektro
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember

Mengetahui / Menyetujui
Dosen Pembimbing



Ir. MOCH. MOEFADOL ASYARI
NIP. 130 422 814

SURABAYA
JULI, 1998

ABSTRAK

Pemakaian Instrumen digital pada saat ini sudah mulai banyak, dan banyak dari instrumen tersebut dapat diprogram dan dikontrol secara terpadu melaui host dalam hal ini adalah komputer.Sistem ini mempunyai bagian pengambilan data, pengolahan data dan pengiriman data ke komputer. Pada Tugas Akhir ini akan dibuat aplikasi dari pengiriman data baik dari komputer ke instrumen untuk mengontrol kerja peralatan maupun dari instrumen ke komputer sebagai data pengukuran. Sedangkan komputer melakukan pengaturan kerja instrumen dan pengolahan datanya.

Untuk menghubungkan instrumen instrumen tersebut diperlukan cara interfacing instrumen instrumen tersebut ke komputer sehingga dapat dikontrol. Salah satu cara interface adalah dengan menggunakan standard interface bus IEEE488 yang dikenal dengan nama GPIB. Dengan menggunakan standar ini maka semua peralatan atau penambahan peralatan yang baru akan tinggal pasang dan pengaturannya sama.

Dalam Tugas Akhir ini akan dibuat interface dari instrumen instrumen ke bus GPIB sehingga instrumen tersebut dikenali dan dapat menerima data dan mengirim data. Sebagai contoh aplikasinya dibuat suatu sistem yang berfungsi untuk melakukan tes terhadap suatu rangkaian sehingga bisa diketahui respon rangkaian tersebut. Untuk merealisasikan hal tersebut maka dibuat modul modul yang dihubungkan ke bus GPIB antara lain : sinyal generator, switch dan ADC.

KATA PENGANTAR

Segala puji dan syukur penyusun panjatkan kehadirat Tuhan Yang Maha Esa berkat segala rahmat-Nya kami dapat menyelesaikan Tugas Akhir ini dengan judul:

SISTEM SINTESIS DAN ANALISIS RANGKAIAN ELEKTRONIKA SECARA TERPADU DENGAN MENGGUNAKAN INTERFACE BUS IEEE.488

Tugas Akhir ini merupakan salah satu persyaratan unutk menyelesaikan studi di Bidang Studi ElektronikaJurusan Teknik Elektro, Fakultas Teknologi Sepuluh Nopember Surabaya dengan beban 6 SKS.

Dalam pengerjaan Tugas Akhir ini penulis mengacu kepada teori yang didapat selama kuliah, literatur dan bimbingan dari para dosen dan bantuan moral dari pihak pihak yang lain. Sehingga penulis mengucapkan terima kasih yang sebesar besarnya kepada :

- Ir. Moch. Moefadol Asyari selaku dosen pembimbing
- Ir. Soetikno selaku Koordinator Bidang Studi Elektronika.
- Ir. Hanny Budhi Nugroho selaku dosen wali
- Bapak dan Ibu dosen yang telah memberikan bimbingan ilmu selama penulis menempuh kuliah.
- Teman teman Bidang Studi Elektronika antara lain Aris, Dyah, Unggul, Ajik, Pai, Gogot, Wahid, Andriawan, dan adik adik Mahasiswa yang lainnya.
- Teman teman dekat penulis yang memberi dorongan moral kepada penulis : Putu, Ketut, Benny, Doris, Khuz dan Yuni yang selalu menemani penulis dalam pengerjaan Tugas Akhir ini.
- Orang Tua penulis, adik dan kakak penulis yang selalu memberikan banyak perhatian kepada penulis selama penyusunan Tugas Akhir ini.

Penulis berharap Tugas Akhir ini dapat bermanfaat dan dikembangkan semaksimal mungkin.

Surabaya, Agustus 1998

DAFTAR ISI

DAFTAR GAMBAR

Daftar Tabel

BAB I

PENDAHULUAN

1.1. Latar Belakang

Perkembangan peralatan elektronika pada saat ini sangatlah pesat. Terutama pada elektronika digital. Saat ini banyak dijumpai peralatan digital yang dengan mudah untuk diprogram sesuai dengan kebutuhan kita. Salah satu dari peralatan digital tersebut adalah instrumentasi atau alat alat ukur baik yang dipergunakan di laboratorium maupun di industri . Dimana saat ini pengukuran pengukuran tersebut sudah mulai beralih dari pengukuran analog ke digital. Hal ini disebabkan oleh semakin pesisinya hasil yang dicapai dengan menggunakan alat ukur digital.

Semakin berkembangnya teknologi digital tersebut, menyebabkan perkembangan pula pada alat ukur tersebut. Instrument digital tersebut dapat diprogram, sesuai dengan yang kita inginkan. Parameter parameter alat ukur tersebut dapat ditentukan terlebih dahulu sebelum digunakan, atau diatur kemudian saat digunakan. Pengaturan / pemrograman tersebut dapat berupa pemrograman secara manual pada font panel dari instrument tersebut maupun dari suatu host komputer pengendali.

Jika beberapa instrument dihubungkan menjadi satu seperti sinyal generator, multimeter, frekwensi counter dan sebuah host pengendali maka dapat dipergunakan sebagai suatu sistem tes untuk menganalisa suatu rangkaian elektronika. Penggunaan suatu host pengendali, dalam hal ini adalah sebuah

komputer maka instrument instrument tersebut dapat dikendalikan untuk tujuan yang kita inginkan dalam melakukan tes atau analisa dari suatu rangkaian.

Beberapa intrumen yang berbeda tersebut dihubungkan menjadi satu ke host komputer sebagai pengendali, maka diperlukan suatu cara interface dari instrument instrument terserbut ke komputer. Sehingga semua peralatan tersebut dapat dikendalikan dan dimonitor dalam satu monitor. Salah satu cara untuk menghubungkannya adalah dengan menggunakan suatu interface standar untuk instrumentasi.

1.2. Permasalahaan

Untuk menghubungkan beberapa instrument sehingga dapat dipergunakan sebagai sistem tes maka dipergunakan interface dari tiap tiap peralatan tersebut ke host komputer. Sedangkan untuk menghubungkan alat tersebut terdapat banyak cara, sering kali tiap tiap peralatan tersebut memiliki cara yang berbeda beda sesuai dengan spesifikasi dari perusahaan pembuatnya. Sehingga kita menjadi repot jika akan menambah instrument / unit yang baru pada sistem yang telah ada.

Dari permasalahan tersebut diperlukan suatu cara / metode untuk melakukan penggabungan tiap instrument instrument tersebut sehingga dapat dihubungkan bersama sama tanpa harus banyak mengubah sistem yang telah ada. Alternatif yang ada adalah menggunakan bus yang sudah ada standarnya. Tiap peralatan yang akan dihubungkan harus memiliki terminal yang sesuai dengan standar sistem bus yang disepakati. Salah satu bus interface yang banyak digunakan untuk

menghubungkan beberapa peralatan instrumentasi adalah standar bus IEEE 488 atau sering disebut dengan General Purpose Interface Bus (GPIB).

Dimana standar bus IEEE 488 merupakan jalur jalur pasif yang mempunyai banyak saluran baik untuk data, control maupun handsakenya. Dimana format data dan control handsakenya sudah disepakati bersama. Permasalahaan yang ada adalah bagaimana kita membuat interface yang dapat dihubungkan ke host pengendali bus IEE 488 sehingga beberapa modul instrumentasi yang kita buat dapat digabungkan dengan peralatan dengan interface yang sama, sehingga dapat dipergunakan sebagai sistem analisa terhadap suatu rangkaian elektronika.

1.3. Batasan Permasalahan

Dalam Tugas Akhir ini yang menjadi pokok permasalahaan adalah bagaimana kita membuat dan menghubungkan beberapa perlalatan yang kita buat sehingga dapat dihubungkan dengan standar bus IEEE 488 sehingga membentuk suatu sistem yang dapat dipergunakan untuk menganalisa suatu rangkaian elektronika. Untuk hal tersebut maka akan dibuat interface dan modul yang dapat membentuk bus IEEE 488. Untuk keperluan pengukuran respon dari suatu rangkaian yang dalam hal ini dibatasi untuk respon DC, respon frekwensi, dan respon transient dari suatu peralatan elektronika yang akan di analisa.

Dimana peralatan yang akan dihubungkan adalah sinyal generator, sistem switching dan sistem pengmbilan data dengan menggunakan rangkaian ADC yang berfungsi sebagai voltmeter digital. Dengan tiga peralatan tersebut dapat dibentuk

sistem analisa dan sintesa rangkaian elektronika yang dikendalikan oleh host komputer dengan dasar interface bus IEEE 488.

1.4. Tujuan

Tujuan dari tugas akhir ini adalah memperkenalkan dan sekaligus merealisasikan konsep tes rangkaian secara terpadu dengan menggunakan bus IEEE 488. Dengan ini diharapkan konsep tes rangkaian tersebut dapat diperguinakan dalam menunjang jalannya praktikum praktikum yang dilaksanakan di laboratorium, sehingga proses praktikum dapat dimonitor dan hasilnya dapat disimpan dalam bentuk file.

1.5. Metodologi

Untuk mencapai tujuan yang telah direncanakan, maka dalam pengerjaan tugas akhir ini dilakukan langkah langkah sebagai berikut :

Hal pertama yang dilakukan adalah melakukan studi literatur tentang standar bus IEEE 488, dimana ini meliputi pengumpulan literatur dan informasi yang membahas tentang standar terserbut dan kemudian mempelajarinya. Dimana bus tersebut akan dipergunakan sebagai alat interface antara instrument instrument tersebut dengan host komputer.

Selanjutnya adalah studi literatur tentang tes equipment yang terdiri dari pembangkitan suatu sinyal tes dan pengambilan sinyal tes yang telah diolah oleh rangkaian yang dites. Studi ini juga meliputi proses ADC dan DAC dimana meliputi proses konversi dan pengoperasian.

Setelah studi literatur tentang GPIB dan akuisisi data maka dilanjutkan dengan studi tentang komponen pendukungnya. Dalam tugas akhir ini dilakukan proses optimasi penggunaan komponen dengan menggunakan sistem PLD dan GAL dengan bahasa pemrograman VHDL. Sehingga proses pemrograman berada pada bahasa pemrograman tingkat tinggi / high level programming language yang berorientasi pada kosa kata.

Langkah selanjutnya selanjutnya adalah mendesain rangkaian dimulai dari blok diagram yang dilanjutkan dengan mendesain rangkaian secara lengkap dimana bagian digital dari alat diusahakan dapat disatukan dengan VHDL. Serta mendesain PCB, merakitnya serta melakukan pengujian terhadap alat yang dibuat.

Proses terakhir yang dilakukan adalah mempelajari teknik pemrograman untuk mengoperasikan alat tersebut sehingga berfungsi sebagaimana yang diinginkan.

1.6. Sistematika Pembahasan

Sistematika pembahasan dalam tugas akhir ini adalah sebagai berikut :

BAB II Menjelaskan tentang konsep dasar dari standar bus GPIB.

BAB III Menjelaskan tentang konsep perencanaan sistem dan perangkat keras

BAB IV Menjelaskan tentang teknik perangkat lunak yang dipergunakan

BAB V Menjelaskan tentang hasil pengujian dan pengukuran

BAB VI Merupakan penutup yang berisikan kesimpulan dan saran tentang tugas akhir ini.

1.7.Relevansi

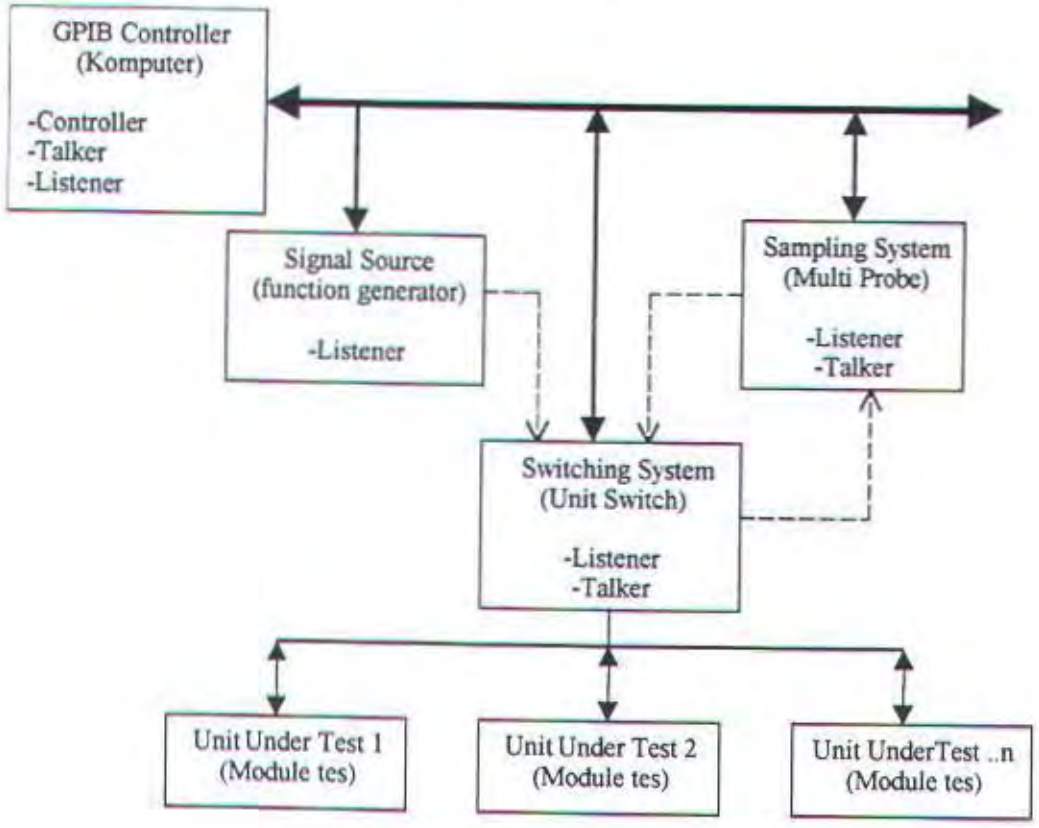
Sistem peralatan hasil dari tugas akhir ini diharapkan dapat sesuai dengan standar GPIB IEEE 488, sehingga dapat juga dihubungkan dengan instrument instrument lain yang juga memiliki fasilitas GPIB IEEE488. Dari sistem yang dibentuk ini diharapkan dapat digunakan sebagai suatu sistem untuk melakukan analisa dari suatu rangkaian elektronika, baik dipergunakan dalam laboratorium dalam praktikum maupun secara komersial diindustri.

BAB II

TEORI PENUNJANG

Dalam tugas akhir ini akan dibuat interface untuk instrument instrument yang dapat diprogram dan komputer ke bus IEEE 488, sehingga instrument tersebut dapat dipergunakan sebagai suatu sistem untuk melakukan analisa suatu rangkaian elektronika. Hasil dari analisa tersebut dapat berupa analisa DC, analisa AC, analisa respon frekwensi atau analisa transient nya yang dapat disimpan maupun diprint.

Dalam sistem analisa dan sintesa rangkaian ini akan terdiri dari beberapa module yang berfungsi sebagai tes equipment. Module module tersebut adalah



Gambar 2.1. Diagram Sistem ATE

multi function sinyal generator yang berfungsi sebagai pembangkit sinyal tes, module switch sebagai pengendali dan penentu sinyal yang masuk ke rangkaian dan sinyal yang akan diambil sample-nya oleh ADC. Module ADC sebagai multi probe yang akan mengambil sample dari suatu titik probe pada rangkaian. Blok diagram dari sistem tes otomatis diperlihatkan pada gambar 2.1.

2.1. GPIB – General Purpose Interface Bus

2.1.1. Gambaran Umum General Purpose Interface Bus

Perkembangan dari standar interface bus telah ada beberapa tahun sebelum GPIB digunakan. Tiap perusahaan menggunakan metodenya sendiri sendiri untuk interfacing alatnya. Pada tahun 1965 Hewlett-Packard mulai membuat standar bus dan menggunakannya dengan nama HP-IB atau Hewlett Packard Interface Bus. Dengan menggunakan sistem bus ini mereka pergunakan untuk menghubungkan instrument instrument yang dapat diprogram ke sebuah pengendali. Dengan menggunakan bus ini juga sistem pengendali tersebut dapat mengatur kerja dari instrument, memerintahkan suatu instrument untuk melakukan suatu rangkaian pengukuran, menampilkan, menyimpan dan mencetak suatu hasil pengukuran tersebut.

Karena bus HPIB ini dapat mentransfer data dengan kecepatan tinggi (1Mbyte /s) dan perkembangan akan kebutuhan peralatan test yang otomatis, dan perlunya kompatibelitas antar instrument menyebabkan sistem bus ini menjadi cepat populer. National Instrument kemudian memperluas kegunaaan dari HPIB ini untuk dipergunakan oleh komputer komputer yang dibuat oleh perusahaan

perusahaan lain. Sejak itu nama GPIB lebih luas dipergunakan dari pada nama HPIB. Setelah cukup luas dipergunakan, IEEE memutuskan bahwa sistem bus ini dipergunakan sebagai dasar untuk standar nasional di USA. Bus interface ini kemudian diterima sebagai standar IEEE 488-1975 dan kemudian dikembangkan menjadi IEEE 488.1.-1987.

IEEE 488.1-1987 dibuat untuk memperkuat standar asli dengan mendefinisikan secara tepat bagaimana pengendali dan instrument dapat berkumunikasi. Tahun 990, Standard Command of Programmable Instrument (SCPI) menggunakan struktur struktur perintah yang dipergunakan dalam IEEE 488.2 dan kemudian dibuat khusus meliputi kumpulan perintah yang dipergunakan pada setiap instrument SCPI. IEEE kemudian memberi nama bus IEEE 488 sebagai ganti nama GPIB.

GPIB adalah sebuah struktur bus yang dapat mentransfer data bit paralel dan byte serial. Standar ini didesain tidak untuk jaringan komputer tetapi sebagai interface untuk penghubung sistem yang cepat antara instrument dengan sebuah komputer atau dengan instrument lain. Instrument instrument seperti komputer, voltmeter, power supply, sinyal generator, frekwensi counter dan yang lainnya dapat dilengkapi dengan bus ini.

Dengan menggunakan sistem ini banyak instrument yang bisa dihubungkan, standar IEEE mengijinkan sampai 15 peralatan dihubungkan dalam bus ini pada saat yang sama dan dengan hanya menggunakan satu interface saja. Jika tidak dengan bus ini maka bisa diperlukan 15 interface yang berbeda untuk menghubungkannya dan dikendalikan dengan software yang berbeda pula untuk

mengendalikan suatu instrument dengan fungsi yang sama. Karena pada GPIB tidak ada saluran alamat, maka alamat dilewatkan pada data bus dengan mengaktifkan saluran ATN, ketika saluran ATN ini tidak aktif maka informasi pada bus dianggap sebagai data dan jika lain akan dianggap sebagai data control yang bisa juga berupa alamat dari suatu instrument. Pada setiap peralatan biasanya dilengkapi dengan dip switch untuk menentukan alamat dari instrument tersebut.

Dengan adanya standar instrument tersebut setiap pabrik pembuat instrument yang berbeda dapat membuat instrumentnya memenuhi standar tersebut sehingga pemakai dapat membeli instrument yang berbeda dari pabrik yang lain dan kemudian menggabungkannya bersama sama dengan menggunakan bus GPIB.

Salah satu kelebihan GPIB adalah dalam penambahan instrument yang baru. Jika ingin menambah instrument yang baru maka kita tinggal membeli instrument yang dilengkapi dengan interface GPIB dan langsung dapat dihubungkan pada bus tersebut.

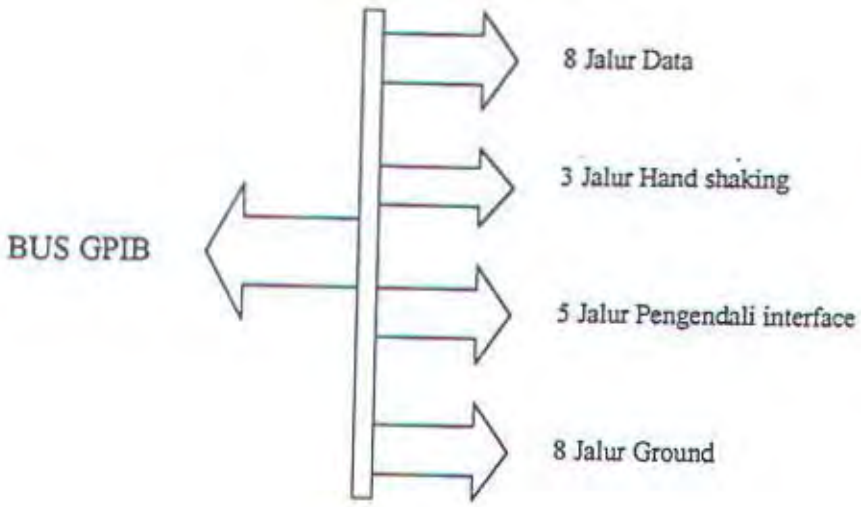
2.1.2. Struktur Bus GPIB

GPIB sebenarnya adalah sekumpulan bus atau sekumpulan dari saluran saluran yang fungsi dari masing masing saluran tersebut sudah disepakati bersama dan sudah dijadikan standar. Bus tersebut ditunjukan pada gambar 2.2.

Seperti pada gambar 2.2 bus GPIB terdiri dari 24 saluran yang dibagi dalam 4 kelompok yaitu :

- 8 Saluran data : DIO1 – DIO8
- 3 Saluran Handshaking : DAV, NRFD, NDAC

- 5 Saluran pengendali interface : IFC, ATN, SRQ, REN, EOI
- 8 Saluran Ground



Gambar 2.2. Sinyal Bus GPIB

2.1.2.1. Saluran Data

Saluran data terdiri dari 8 saluran dua arah. Selain membawa data saluran ini juga dipergunakan untuk membawa informasi lain yaitu : alamat, status Byte, dan spesial bus command. Bus data juga dipergunakan untuk memberi fungsi pengalamatan karena sistem bus GPIB tidak menyediakan bus alamat.

Semua sinyal pada bus GPIB menggunakan logika negatif yaitu level tegangan tinggi ditandai dengan logika 0 untuk kondisi false dan level tegangan rendah ditandai dengan logika 1 untuk menyatakan kondisi true.

Informasi informasi yang dibawa saluran data dibagi dalam 2 mode yaitu:

1. Mode data

Ditandai dengan ATN false (0) dimana informasi yang dibawa merupakan device dependent message atau biasa disebut data saja. Data ini dikirim

oleh talker ke satu atau lebih listener. Pada mode ini ke delapan saluran data digunakan seluruhnya.

2. Mode perintah

Ditandai dengan ATN true (1) dimana informasi yang dibawa merupakan interface message atau disebut dengan command. Semua command dikirim oleh controller kesemua peralatan yang terhubung ke pada bus baik yang aktif maupun yang tidak aktif. Pada mode ini hanya menggunakan 7 bit yaitu menggunakan kode ASCII dan bit ke delapan dipergunakan sebagai parity cek jika diperlukan.

Transfer data pada bus GPIB adalah secara Byte serial dan bit paralel. Sebuah informasi yang lengkap terdiri dari satu atau banyak byte secara serial yang dikirim secara tidak sinkron

2.1.2.2. Saluran Handshaking

Saluran handshaking ini terdiri dari 3 saluran yang masing masing mempunyai fungsi sendiri sendiri. Ketiga saluran ini dipergunakan untuk memastikan transfer data yang sah antara controller, talker, dan listener.

Informasi ditransmisikan pada saluran data dibawah urutan urutan kontrol dari ketiga saluran ini. Transfer data dapat diperoses secepat peralatan dapat meresponnya, tapi tidak dapat lebih cepat dari peralatan paling lambat yang alamatnya sedang aktif pada saat itu. Hal ini mengijinkan beberapa peralatan dengan kecepatan berbeda untuk menerima data yang sama secara bersamaan.

Masing masing saluran tersebut adalah :

1. DAV (Data Valid)

Saluran ini dikontrol oleh talker ketika mengirim data dan dikontrol oleh controller pada saat data yang dikirim adalah command. Talker membuat DAV menjadi true (0) untuk menyatakan pada listener bahwa informasi yang ada pada saluran data telah valid. Diperlukan time delay pada peralatan untuk mengeluarkan data pada bus data.

2. NRFD (Not Ready For Data)

Saluran ini dikontrol oleh listener ketika menerima suatu data dan dikontrol oleh semua peralatan ketika data yang dikirim adalah command. Dengan membuat NRFD false listener memberitahu pada talker dan controller mereka sudah siap untuk menerima data.

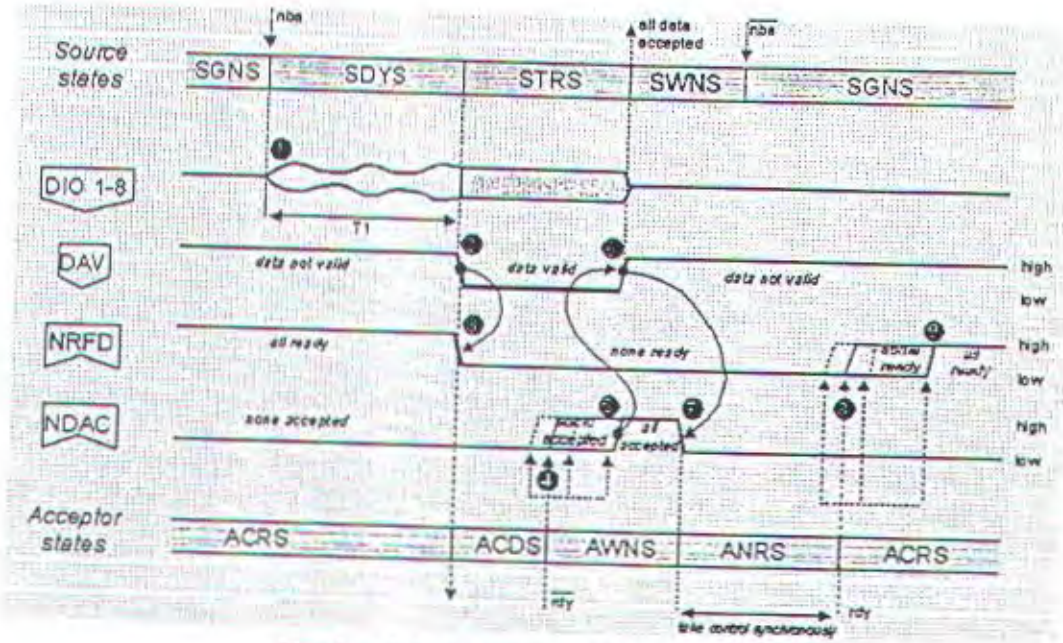
3. NDAC (Not Data Accepted)

Saluran ini dikontrol oleh listener ketika menerima sebuah data dan dikontrol oleh semua peralatan ketika menerima command. Dengan membuat NDAC false (1) listener memberitahu pada talker bahwa mereka telah menerima pesan yang dikirimkan.

Untuk saluran NRFD dan NDAC keduanya di wire-or, artinya saluran hanya dapat high jika semua instrument yang aktif mempunyai output high. Jadi saluran

NRFD akan high jika semua peralatan yang aktif dengan kecepatan yang berbeda telah siap semuanya untuk menerima data.

Dalam mode command semua instrument melakukan handshaking dengan controller, sedangkan dalam mode data hanya instrument yang aktif yang melakukan handshaking. Sedangkan dalam pembacaan status bit tidak menggunakan handshaking. Prosedur handshaking diperlihatkan dengan diagram waktu pada gambar 2.3



Gambar 2.3. timing handshaking bus GPIB

2.1.2.3. Saluran Pengendali Interface

Pengendali interface mempunyai 5 saluran yaitu: ATN, IFC, REN, EOI, dan SRQ. Semua saluran ini dibawah kendali controller kecuali SRQ yang dikendalikan oleh instrument dan digunakan untuk menginterupt controller supaya

sistem remote kontrol, controller akan mengaktifkan REN dan alamat instrumen. Instrumen akan tetap diset dalam remote kontrol selama REN diaktifkan kecuali perintah return to local diberikan. Level rendah pada REN mengijinkan semua peralatan pada bus untuk dikontrol oleh GPIB. Jika saluran ini tinggi maka semua instrumen diset ke kontrol lokal antara 100 us.

4. **EOI (end of identify)**

   Saluran EOI mempunyai 2 fungsi dalam kombinasinya dengan ATN. Dalam mode data (ATN salah) talker dapat menggunakan EOI untuk memberi isyarat kepada controller kalau ada akhir pengalihan data. Controller dapat juga mengirim sebuah EOI (benar) dalam mode perintah untuk pelaksanaan urut-urutan polling dalam paralel-poll.

5. **SRQ (service request)**

   Dengan membuat SRQ benar maka instrumen memberitahu pada controller kalau ada yang memerlukan perhatian dari controller agar diijinkan menggunakan bus data, ketika bus data sedang dipakai oleh instrumen yang lain. SRQ digunakan antara lain untuk mengisyaratkan kalau tugas yang diberikan telah diselesaikan, terjadi error dalam suatu operasi, atau ada data yang ingin ditransmisikan, karena kebanyakan operasi didalam instrumen bisa membutuhkan waktu yang lama antara puluhan detik (misal pengukuran frekuensi yang amat rendah). Maka tidak bijaksana kalau sistem diblok sampai instrumen menyelesaikan pengukuran. Lebih baik waktu yang terbuang digunakan untuk melanjutkan aktifitas lain dan akan menginterupt dengan cara mengirim sinyal SRQ jika pekerjaan telah selesai.

Ketika controller menerima sinyal SRQ dari salah satu instrumen maka controller akan menghentikan semua aktifitas yang lain dan menanggapi permintaan SRQ. Langkah pertama yang dilakukan controller adalah mencari instrumen mana yang mengirim sinyal SRQ. Ada dua cara berbeda yaitu parallel-poll dan serial-poll

Parallel Polling

Parallel polling mengijinkan controller mengecek 8 instrumen sekaligus dengan menentukan status tertentu diantara peralatan. Dalam mode parallel-poll, setiap instrumen mempunyai saluran data khusus sebagai identifikasi. Controller dapat mengenali instrumen yang meminta pelayanan dengan melihat saluran data mana yang membawa sinyal. Controller membuat ATN dan EOI benar, hal ini menyebabkan peralatan yang dikonfigurasi untuk parallel-poll (dengan PPC command) menempatkan status bit pada salah satu saluran data bus. Controller kemudian membaca respon yang diberikan oleh instrumen dan menentukan peralatan yang mengirim SRQ. Ketikan EOI atau ATN false, maka controller akan menghentikan polling. Instrumen harus merespon antara 200 ns setelah EOI dan ATN low.

Serial Polling

Dalam mode serial poll, semua instrumen ditanya sendiri-sendiri apakah mereka meminta pelayanan. Pertama controller mengirim bus command UNL kemudian mengaktifkan serial port dengan mengirim SPE. Controller kemudian mengirim alamat instrumen sat persatu, instrumen merespon dengan

mengirim single status byte (STB) pada controller. Status byte terdiri 8 bit yang masing-masing mempunyai fungsi sendiri-sendiri. Bit 1 sampai bit 6 dan bit 8 digunakan untuk menspesifikasikan status device dependent yang ditentukan sendiri oleh perancang peralatan. Bit 7 digunakan untuk pesan RQS (request service), jika bit 7 membawa informasi 1 berarti instrumen tersebut yang meminta pelayanan. Controller membaca dan mencocokan apakah peralatan tersebut yang mengirim SRQ. Setelah ditemukan, controller mengubah lagi interface untuk keluar dari mode serial poll dengan mengirimkan command SPD kemudian memberi pelayanan yang dibutuhkan.

2.1.2.4. Sistem Data GPIB

Device Dependent Message

Informasi pada bus data disebut device dependent message ketika interface dalam mode data. Device dependent message dibagi dalam 4 kelompok.

1. **Data pengukuran**

   Data pengukuran adalah merupakan output dari sebuah instrumen pengukur (misal DVM, pencacah frekuensi, dll). Instrumen mengirim data ini melalui bus ketika dia dialamati sebagai talker, kesatu atau lebih listener yang telah dialamati.

2. **Data display**

   Adalah informasi yang diterima oleh instrumen pada saat listener. Misalnya adalah data yang akan dicetak.

3. **Data status**

Adalah informasi yang dikirim oleh instrumen, informasi ini memberitahukan kondisi internal dari instrumen. Contoh data status adalah byte status yang dikirim pada saat rutin serial poll.

4. **Data programming**

Disebut juga instruksi programming, data ini digunakan untuk mengontrol satu atau lebih fungsi-fungsi dari instrumen (misal: range, mode, dll). Data ini dikirim oleh controller ke satu listener tertentu. Instruksi ini berbeda antara instrumen yang satu dengan yang lainnya dan ditentukan oleh perancang instrumen.

Interface Message

Interface message dikirim oleh controller pada bus data selama mode command (ATN benar). Interface message digunakan untuk memprogram sistem interface dari peralatan yang terhubung pada bus. Instruksi-instruksi ini dikodekan dalam 7 bit dan telah didefinisikan sehingga mempunyai kesamaan dalam semua sistem GPIB.

Karena setiap instruksi hanya menggunakan 7 bit maka dapat dipresentasikan dengan karakter ASCII seperti tampak pada tabel 2.1.

Command dapat berupa Uniline atau multiline. Uniline command mengaktifkan satu kadang lebih dari 5 saluran pengendali interface. Multiline command adalah yang juga disebut dengan interface message. Uniline tidak memerlukan bus data, sedangkan multiline menggunakan bus data.

Tabel 2.2 Uniline Command

	Command	DAV	NRFD	NDAC	ATN	EOI	SRQ	IFC	REN
ATN	Attention				1				
DAB	Data byte				0				
DAC	Data accept			1					
DAV	Data valid	1							
END	Akhir data					1			
IFC	Interface clear							1	
PPR	Paralel poll resp				1	1			
REN	Remote enable								1
RFD	Ready for data		1						
SRQ	Service request						1		

Interface message yang berupa multiline command dibagi dalam 4 kelompok yaitu: address command, universal command, addressed command, dan secondary command

Type dari command ditentukan oleh bit 7, 6, dan 5. Sebuah peralatan dapat memberikan respon terhadap command-command tersebut jika pada bagian decoding message pada interface driver mempunyai fasilitas deteksi untuk kode yang bersangkutan. Tidak semua instrumen harus mampu mendeteksi seluruh command-command tersebut tapi hanya jika diperlukan saja baru digunakan.

1. Address command

Perintah ini digunakan untuk mengalamati suatu instrumen sebagai talker atau listener. Jika controller ingin mengalamati instrumen, dia menempatkan kode alamat instrumen yang diinginkan pada bus data, dan pada saat yang sama ATN

dibuat rendah. Kemudian semua peralatan pada bus GPIB membandingkan kode alamat tersebut dengan alamat mereka sendiri, jika alamatnya sesuai maka instrumen tersebut akan aktif pada bus.

Sebuah kode alamat terdiri dari 7 bit sedang bit MSB tidak dipakai. Bit ke 1-5 adalah kombinasi bilangan biner untuk alamat instrumen. Ada 2 special pengalamatan yaitu UNL dan UNT dimana bit ke 1 sampai bit ke 5 semua berharga 1, pengalamatan ini digunakan untuk menghapus pengalamatan pada instrumen. Sedang bit ke 6 dan 7 digunakan untuk menentukan apakah alamat pada bit ke 1 sampai 5 sebagai talker atau listener.

- Talker Address (TAD)

  Untuk memilih satu instrumen sebagai sumber data (talker). Pengalamatan ini secara otomatis mematikan talker yang aktif sebelumnya.

- Listener Address (LAD)

  Digunakan untuk memilih sebuah instrumen sebagai penerima data *listener), pengalamatan ini tidak akan mempengaruhi listener yang sedang aktif.

- Unlisten (UNL)

  Semua listener direset untuk tidak aktif atau dalam keadaan tak dialamati. perintah ini dapat digunakan sebelum listener baru diaktifkan.

- Untalk (UNT)

  Perintah ini digunakan untuk menghapus pengalamatan dari talker. Talker dapat juga direset secara otomatis pada saat mengirimkan alamat talker yang baru.

2. Universal Command

Bus command dalam group ini akan mempengaruhi semua peralatan, apakah mereka dialamati atau tidak, asalkan mereka mempunyai fasilitas dekoding untuk perintah-perintah yang bersangkutan didalam bagian dekoding message.

- **Device Clear (DCL)**

  Semua peralatan akan direset ke keadaan yang telah didefinisikan. DCL hanya mereset bagian device functionnya saja, sedangkan bagian interface function tidak direset.

- **Local Lock Out (LLO)**

  Bus command ini bersama-sama dengan aktifitas dari saluran REN akan mematikan tombol kontrol lokal panel depan sederhana. Dengan mematikan panel kontrol lokal akan dapat dihindari konflik antara instruksi yang dikirim melalui bus dan instruksi yang dikirim dari panel kontrol lokal.

- **Serial POLL Enable (SPE)**

  Perintah ini mengijinkan mode serial poll pada bus, sehingga urut-urutan serial polling dapat dimulai. Hal ini dikerjakan setelah permintaan pelayanan dari peralatan dengan mengaktifkan saluran SRQ.

- **Serial Pool Disable (SPD)**

  Bus command ini akan mematikan mode serial poll pada bus GPIB dan digunakan untuk mengakhiri urut-urutan serial poll.

- **Paralel Poll Unconfigure (PPU)**

Bus command ini akan mereset respon parallel-poll semua peralatan pada bus GPIB. Hal ini mengijinkan respon parallel-poll baru dispesifikasikan pada peralatan.

3. Addressed Command

Hanya peralatan yang telah dialamati yang dapat merespon addressed command.

- **Selective Device Clear (SDC)**

  Peralatan yang telah dialamati sebagai listener saja yang akan dikembalikan ke kondisi yang diketahui atau dengan kata lain direset.

- **Go to Local (GTL)**

  Bus command ini akan mengembalikan respon semua instrumen yang dialamati listener untuk dikontrol oleh kontrol lokal. Perintah ini membatalkan LLO dan mengijinkan peralatan untuk dikontrol manual.

- **Group Execute Trigger (GET)**

  Digunakan untuk mengsinkronkan operasi pada sejumlah peralatan. Peralatan yang akan ditrigger pertama-tama harus diprogram untuk membentuk beberapa tugas khusus ketika mereka menerima trigger. Begitu peralatan menerima GET, mereka memulai tugas mereka.

- **Take Control (TCT)**

  Perintah ini diberikan ketika aktifitas controller dipindahkan untuk mengontrol peralatan lain. Peralatan yang akan menjadi controller baru akan mengirimkan

sebuah listen command (LAD) diikuti oleh TCT command. Setelah menerima TCT command, peralatan bisa meletakkan kontrol pada bus dan memulai mengirim command ke peralatan yang lain.

- **Parallel Poll Configure (PPC)**

  Bus command ini digunakan untuk menyiapkan peralatan-peralatan yang akan ikut serta dalam parallel-poll. Command ini akan diikuti oleh secondary command.

4. Secondary Command

Setelah PPC dikirim, peralatan menunggu untuk menerima secondary command (SC). Salah satu SC yaitu PPE digunakan untuk menspesifikasikan bit ke delapan pada register parallel-poll yang akan digunakan oleh peralatan tersebut untuk menampilkan status, dan menentukan logika status menggunakan logika 0 atau logika 1.

Tiga bit rendah pada byte SC menentukan bit keberapa dari register parallel-poll yang akan dipakai oleh peralatan tersebut. Bit ke-4 digunakan untuk menentukan logika yang digunakan untuk status bit tersebut. Bit ke-5 digunakan untuk menentukan apakah byte command ini sebagai PPE atau sebagai PPD, jika bit ke-5 adalah 0 maka command adalah PPE, jika bit ke-5 berharga 1 maka command merupakan PPD. Bit ke-7 dan bit ke-6 menunjukkan bahwa byte tersebut adalah secondary command.

contoh:

b8	b7	b6	b5	b4	b3	B2	b1
X	1	1	0	0	0	1	0

b3,b2,b1 = 010 = 2 berarti bit yang digunakan bit ke 3, dan logika yang dipakai adalah logika 0, secondary command yang dikirim adalah PPE.

Untuk mendisable peralatan dari respon terhadap parallel-poll digunakan PPD (parallel-poll disable). Tidak semua peralatan memerlukan semua bus command, untuk sebuah contoh misalnya sebuah printer (listener) tidak memerlukan kemampuan decoding bus command TCT, PPC, dan PPU, sehingga bagian decoding message pada interface GPIB dari instrumen dapat dibuat lebih sederhana.

2.1.2.5. Karakteristik GPIB

Standar GPIB menspesifikasikan secara mekanik, elektrik, dan fungsional untuk interface. Dengan informasi ini perancang alat dapat merancang sebuah peralatan dengan sebuah interface yang secara mekanik, elektrik, dan fungsi kompatibel dengan peralatan lainnya pada bus.

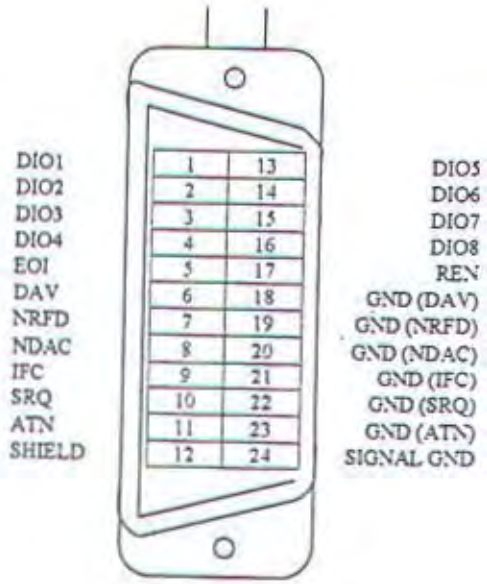
Karakteristik elektrik

Level-level tegangan yang dipakai didalam standar bus GPIB didasarkan pada teknologi TTL. Parameter-parameter yang didefinisikan antara lain: kecepatan transfer data, jumlah peralatan dan panjang kabel, tegangan dan level arus.

seluruh kabel. 7 kabel lagi merupakan grounding yang membungkus masing-masing saluran pengendali.

2. Konektor GPIB

Untuk menghubungkan kabel GPIB pada instrumen digunakan sebuah konektor GPIB. Gambar konektor yang dipakai tampak pada gambar 2.3. Posisi letak sinyal GPIB pada konektor tersebut sudah ditentukan seperti pada gambar. Shield (pin 12) dihubungkan ke bumi atau chasis.

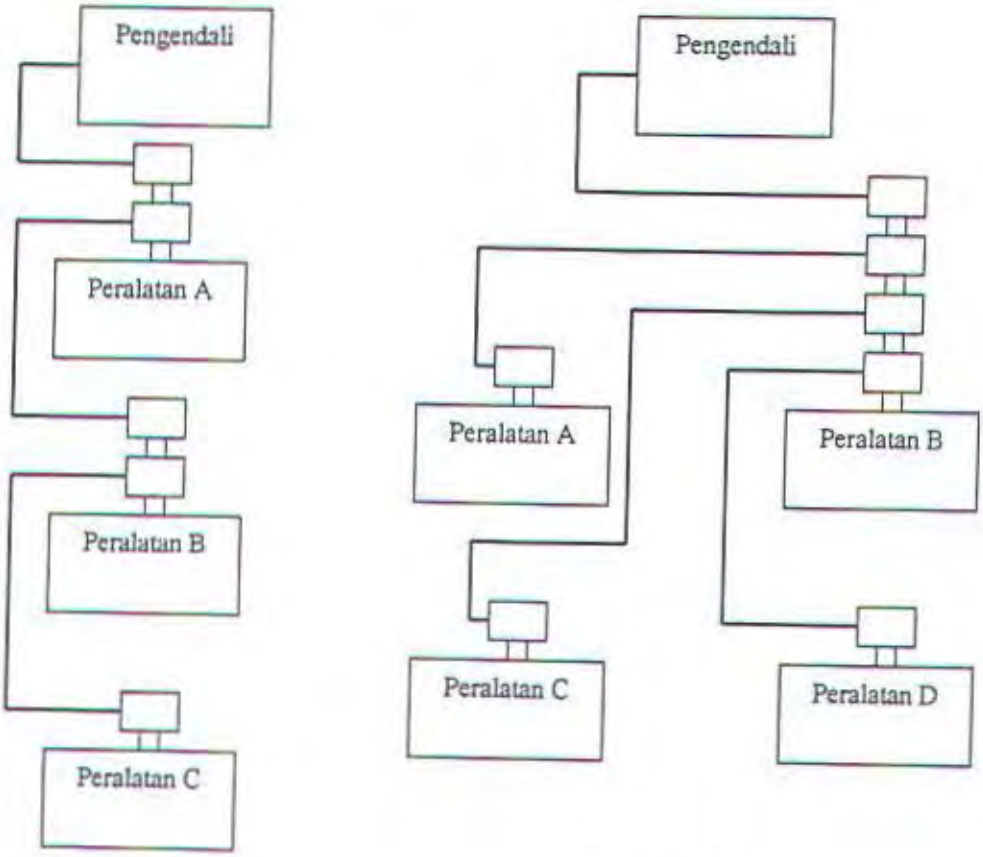


Gambar 2.4. Konektor GPIB

3. Konfigurasi sambungan

Instrumen dapat dihubungkan dalam konfigurasi linier, konfigurasi star atau gabungan dari kedua konfigurasi tersebut. Cara penyusunan peralatan ini

diperlukan jika ingin menyusun beberapa peralatan dengan posisi tempat yang sudah tertentu. Dengan demikian kita dapat memilih konfigurasi dengan menyesuaikan jumlah peralatan yang terpasang dan panjang kabel yang sudah ditentukan.



a. konfigurasi linier

b. konfigurasi star

Gambar 2.5. Konfigurasi Sambungan GPIB

Karakteristik Fungsional

Instrumen-instrumen yang akan dihubungkan pada bus GPIB dapat dibedakan berdasarkan fungsi kerja instrumen tersebut. Ada 3 tipe instrumen berdasarkan fungsinya yaitu:

- Hanya sebagai talker, misalnya pencacah frekuensi, instrumen ini hanya mengirimkan data hasil pengukuran.
- Hanya sebagai listener, misalnya sinyal generator, instrumen ini hanya menerima data berupa instruksi untuk mengeset besar amplitude atau frekuensi.
- Sebagai talker dan listener, misalnya digital voltmeter, instrumen ini dapat bertindak sebagai talker maupun listener. Sebagai talker jika instrumen sedang mengirimkan data hasil pengukuran dan sebagai listener pada saat menerima instruksi untuk mengatur range maupun fungsinya (AC, DC, Volt, mA, R, dll).
- Sebagai Controller, talker, dan listener, misalnya komputer yang mempunyai fasilitas GPIB.

Instrumen-instrumen tersebut dihubungkan secara paralel dengan menggunakan bus GPIB. Bus GPIB sendiri terdiri dari 16 saluran yang dibagi dalam 3 kelompok yaitu 8 jalur bus data, 5 saluran pengendali interface, dan 3 saluran handshaking. Masing-masing instrumen memerlukan interface driver supaya dapat berhubungan dengan bus GPIB. Dimana sistem interface drivernya sendiri terdiri dari 4 bagian utama yaitu:

1. Interface function

   Fungsi-fungsi yang ada pada interface function sudah didefinisikan dalam standar IEEE 488, jadi kalau ingin membuat interface untuk instrumen pada bus GPIB harus diketahui terlebih dahulu karakteristik fungsi-fungsi tersebut.

2. Device function

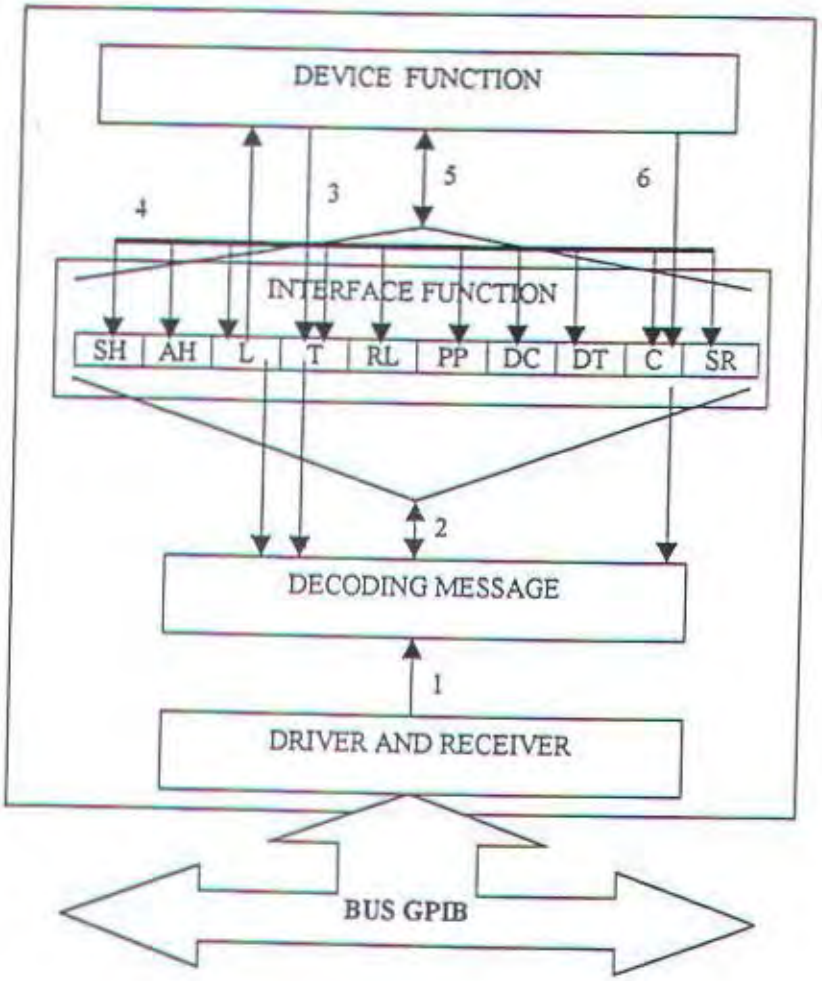
Ruang lingkup, kegunaan, ukuran, isi dan fungsi-fungsi yang ada di dalamnya misalnya kemampuan untuk mengukur sinyal analog, mode operasi, dll adalah diluar standar IEEE 488 ini, dengan demikian desain dari bagian ini tergantung dari masing-masing perancangnya.

3. Driver dan receiver

   Bagian ini berguna untuk menyesuaikan karakteristik elektrik peralatan dengan karakteristik GPIB.

4. Decoding message

   Setiap informasi yang masuk ke interface driver akan melewati bagian ini terlebih dahulu. Pada bagian ini informasi akan dibedakan antara device dependent message dan interface message, jika informasi yang diterima adalah berupa device dependent message maka informasi akan langsung diteruskan pada bagian device function . Tapi jika informasi yang masuk merupakan interface message maka informasi ini akan didecoding terlebih dahulu untuk menterjemahkan command yang dikirim, sinyal-sinyal dari hasil decoding ini akan mempengaruhi state-state pada bagian interface function.



Gambar 2.6. Interface Driver pada GPIB

- Saluran 1 adalah interface bus signal line, sinyal-sinyal informasi pada jalur ini masih utuh, belum dikodekan ataupun diinterpretasikan untuk mempengaruhi state-state dalam interface.
- Saluran 2 adalah jalur interface message dari ataupun yang menuju interface function. Sinyal-sinyal dari saluran 1 akan didecoding dan dibedakan antara device dependent message dan interface message, jika informasi yang diterima pada saluran 1 adalah device dependent message maka akan langsung

2. AH - acceptor handshaking

   Fungsi ini memungkinkan untuk menerima data dari talker dengan menggunakan 3 saluran handshaking.

3. T - talker

   Fungsi ini memungkinkan untuk mengirim data dari device function ketika dialamati sebagai talker.

4. L - listener

   Fungsi ini memungkinkan untuk meneruskan data ke device function ketika dialamati sebagai listener.

5. SR - service request

   Fungsi ini memungkinkan untuk minta interrupt dari controller.

6. RL - remote local

   Fungsi ini memungkinkan untuk beroperasi dalam mode remote lewat GPIB dan dalam mode lokal lewat kontrol panel depan instrumen.

7. PP - parallel-poll

   Memungkinkan untuk merepresentasikan satu bit status kepada controller tanpa dialamati untuk talk dan tanpa handshaking.

8. DC - device clear

   Memungkinkan bagian dari device function pada instrumen untuk direset oleh controller.

9. DT - device trigger

   Memungkinkan start secara individu atau sebagai bagian dari sebuah group.

10. C - controller

Memungkinkan untuk mengirim universal atau address command ke peralatan.

Tidak semua peralatan mempunyai semua fungsi dari interface function misalnya untuk interface listener hanya terdiri dari fungsi: AH, L, RL, DC, dan DT saja bahkan untuk sebagian besar instrumen yang sederhana bisa saja hanya memiliki fungsi AH, dan L saja.

Untuk interface talker hanya terdiri dari fungsi: SH, AH, T, RL, DC, DT, dan PP saja, dan untuk instrumen yang sederhana interface talker bisa hanya terdiri dari SH, AH, dan T saja, sedangkan fungsi controller hanya diperlukan untuk peralatan yang mempunyai sistem kontrol.

Untuk merealisasikan interface function tersebut diperlukan command-command yang sudah distandarkan seperti dijelaskan pada bagian sistem data.

Untuk menjelaskan sistem kerja dari tiap-tiap interface function digunakan diagram state, dimana diagram state akan menjelaskan segala keadaan yang mungkin dengan hubungannya dengan saluran handshaking dan saluran pengendali interface.

Diagram state untuk setiap fungsi sudah tertentu tapi implementasinya pada rangkaian bisa bermacam-macam tergantung pada perancangnya dan instrumentnya sendiri, dimana implementasi dari tiap rangkain tidak boleh menyimpang dari diagram state tersebut.

2.2. Progammable Logic Device dan HDL

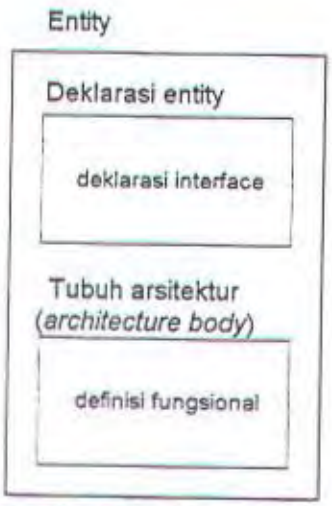
PLD atau Programmable Logic Device adalah peralatan yang dapat diprogram sesuai dengan keinginan kita, baik yang OTP maupun yang erasable. Proses pemrogramannya pun ada beberapa cara, baik secara low level programming maupun dengan high level programing. Secara umum PLD yang OTP hanya dapat diprogram sekali seperti contohnya: PAL16L8, PAL16R8, QL8X12B dan sebagainya. Sedangkan yang eraseble pada umumnya dapat di reprogram lebih dari 10 kali, tergantung kepada garansi dari perusahaan yang memproduksi.

Cara pemrograman secara high level programming ini sering disebut dengan Hardware Discription Language (HDL) dimana programmer berorientasi kepada sintak dan algoritmanya. Salah satunya adalah dengan VHDL, dimana programmer berorientasi ke sintak, sedangkan compiler akan menterjemahkannya menjadi persamaan persamaan yang equivalen.

2.2.1 Hardware Discription Language VHDL

Dalam pembuatan desain VHDL, bergantung pada alat bantu (*tools*) yang digunakan, yang berbeda-beda antara perusahaan satu dengan yang lain. Ada alat bantu yang dapat menghasilkan kode VHDL dari skema rangkaian, ada juga yang tidak. Tetapi pada dasarnya, kode VHDL merupakan kode berbasis teks, yang bisa dibuat menggunakan berbagai macam program pengolah kata (*word processor*) seperti Notepad, Edit, Copy Con, WS, MS Word, dan lain-lain.

Kode VHDL selalu terdiri entity desain (*design entity*) yang memuat minimal sebuah deklarasi entity dan sebuah arsitektur. Dimungkinkan untuk menggunakan banyak arsitektur, tetapi tidak mungkin menggunakan banyak deklarasi entity. Pada bagian deklarasi entity akan didefinisikan kegunaan pin-pin IC yang hendak diprogram. Pada bagian arsitektur akan dideskripsikan cara kerja IC secara fungsional.



Gambar 2.7. Hubungan antara desain entity dan tubuh arsitektur

2.2.1.1. Deklarasi Entity

Penulisan deklarasi entity memiliki syntax:

```
ENTITY entity_name IS PORT (
        [signal_name] : [direction] type ;
        [signal_name] : [direction] type ;
        ..............
        ..............);
Attribute pin _number of entity name:Entity
        " net:pin_number ";
END entity_name;
```

Nama signal merupakan nama yang akan digunakan pada tiap-tiap pin IC. Penggunaan nama lebih baik menurut fungsi dari pin tersebut, misalnya pin digunakan sebagai masuknya sinyal clock, maka diberi nama clock.

Direction mempunyai 4 macam mode, yaitu : IN, OUT, INOUT, BUFFER. Mode IN hanya dapat dipakai sebagai input, OUT hanya sebagai output, dan INOUT sebagai input maupun OUTPUT. Mode BUFFER sebenarnya merupakan output, tetapi secara internal, sinyal outputnya dapat digunakan sebagai *feedback*. Pemilihan mode ini sangat menentukan pin yang dipakai apakah suatu pin io digunakan hanya sebagai output dari suatu persamaan, hasil dari suatu persamaan dapat diambil lagi sebagai input dari suatu persamaan atau pin tersebut digunakan sebagai input saja dan fungsi output .

Selain mendefinisikan fungsi pin yang dipakai juga mendefinisikan nomor pin yang dipakai, hal ini sangat berguna untuk mempermudah penyusunan PCB karena posisi pin pin dapat diatur seperti yang dikehendaki.

2.2.1.2 Arsitektur

Tubuh arsitektur dalam suatu entity berfungsi untuk mendeskripsikan apa yang akan dilakukan atau proses apa yang akan dikerjakan oleh perangkat keras yang didesain. Syntax penyusunannya sebagai berikut :

Architecture name **of** entity_name **is**
[type declarations]
[signal declarations]
[constant declarations]
begin

[architecture definition]
end name;

Penyusunan arsitektur dapat menggunakan 3 buah cara, yaitu :

- Deskripsi tingkah laku (*behavioral description*)

  Dengan deskripsi ini, maka tidak diperlukan struktur dari rangkaian atau skema dari rangkaian. Deskripsi ini lebih menekankan pada output dari IC bila input tertentu masuk. Dengan model ini, maka desain dapat difokuskan ke arah fungsi dari IC tersebut. Ciri utama deskripsi ini adalah adanya perintah "process", yang kadang-kadang diikuti oleh *sentivity list* di belakangnya. Perintah-perintah yang tertulis di dalam bagian arsitektur ini dijalankan secara sekuensial dari awal bagian sampai akhir, sehingga hasil dari persamaan yang dihasilkan adalah suatu proses sekuensial dan bukan suatu persamaan kombinatorial, sehingga sering diakhiri dengan sebuah Flip –Flop dan delaynya diperhitungkan..

- Deskripsi aliran data (*Dataflow description*)

  Deskripsi ini menyatakan bagaimana data akan ditransfer dari sinyal ke sinyal dan dari input ke output tanpa menggunakan pernyataan yang sekuensial. Beberapa ahli menganggap bahwa teknik ini sebenarnya sama saja dengan deskripsi tingkah laku. Perbedaan yang utama dari deskripsi ini adalah tidak digunakannya perintah "process". Dengan diskripsi ini proses yang terjadi pada persamaan adalah berupa persamaan kombinatorial yang hanya terdiri dari gate gate logic dan tanpa diakhiri dengan sebuah Flip –Flop kecuali pada akhir persamaan hasil dari suatu

pin diambil lagi sebagai input ke persamaan tersebut. Seperti contoh A <= '1' when B=0 else A.

- Deskripsi struktural (*structural description*)

  Deskripsi ini bercirikan penggunaan paket-paket komponen (*packages*) yang digunakan sebagai pustaka (*library*). Penggunaan pustaka komponen ini menggunakan perintah "use". Pada alat bantu yang dapat melakukan desain VHDL dari skema rangkaian, biasanya menggunakan deskripsi seperti ini. Di dalam alat bantu tersebut telah terdapat beberapa pustaka gerbang yang sering digunakan, sehingga dari skema rangkaian logika dapat dibuat kode VHDL berdasarkan hubungan komponen dalam rangkaian dan pustaka yang ada. Seandainya diperlukan, pustaka dapat juga dibuat sendiri untuk melengkapi pustaka yang telah ada. Cara desain ini sangat banyak dipergunakan dalam menterjemahkan suatu desain dengan skema rangkaian mejadi suatu bentuk VHDL yang dapat dikompile dengan program desain ke bentuk JEDEC yang akan diisikan.

2.2.2. Pemrograman PLD Dengan VHDL

Kode VHDL yang telah dibuat, diubah (*compile*) menjadi file JEDEC. File ini yang nantinya akan dituliskan pada IC-IC yang akan diprogram. File JEDEC ini dapat diprogramkan pada IC berjenis PAL, GAL, FPGA, cPLD, pASIC, dan lain-lain, contohnya : 16L8, 16R8, 22V10, CY371, CY342, QL5000.

Pada tugas akhir ini digunakan IC bertipe GAL22V10 dan PALCE16V8 National Semiconductor yang merupakan salah satu jenis IC cPLD. Dimana kedua IC tersebut dapat direprogram dengan menggunakan tool dari Cypress dan HiLo system. Untuk proses desain menggunakan software Galaxy dan NOVA untuk mendapatkan file JEDEC yang diisikan pada IC.

Proses desain dengan PLD dan VHDL dapat dibagi menjadi 2 kelompok yaitu dengan menggunakan proses kombinatorial untuk implementasi rangkaian logika dengan gate gate kombinatorial dan dengan menggunakan proses kekuensial yang berurutan dan kadang kala nilai proses terdahulu mempengaruhi proses yang sedang terjadi dan sebagainya sehingga memerlukan sebuah atau lebih flip-flop. Disini menggunakan flip-flop untuk menyimpan suatu nilai karena sebuah ouput dari suatu GAL /PLD selalu terdiri dari makrosel yang terdiri dari array persamaan dan masuk ke sebuah flip –flop. Penggunaan flip flop ini disesuaikan dengan cara pendesainan yang dilakukan.

Hal yang perlu diperhatikan dalam desain adalah jumlah makrosel yang ada pada IC tersebut dan kemampuan makrosel tersebut menangani persamaan inputnya. Karena tiap pin pada suatu IC PLD mempunyai maksimum persamaan yang dapat dimasukkan secara SOP.

2.3. Automatic Test Equipment

Saat ini Otomatisasi telah menggeser industri elektronika. Komputer sekarang dapat menangani disain dan pembuatan PCB sampai pada perakitannya. Langkah kemudian adalah mencoba untuk melakukan tes secara otomatis kepada peralatan yang dirakit tadi dengan menggunakan tes otomatis atau sering disebut Automatic Test Equipment.

Penggunaan sitem tes otomatis ini akan meningkatkan kwalitas kerja dari personal, mengeliminasi beberapa pengulangan step dari suatu tes dan memperkecil campur tangan manusia untuk mengontrol rutin dari suatu tes. Suatu sistem tes otomatis sangatlah tergantung kepada peralatan yang akan ditesnya, namun secara garis besar dalam suatu sistem tes otomatis akan terdapat :

- Controller, pada umumnya adalah mini komputer, mikrokomputer, kalkulator atau sebuah pengendali bus tertentu yang akan mengatur urutan dari tes. Mengontrol urutan perintah, urutan data dan proses peritungannya. Sehingga controller sangat memerlukakn suatu software pengendali yang mengontrol tiap step dari urutan tes.
- Stimuli, pada umumnya adalah sumber sinyal, yang akan memberikan input tes kepada UUT. Sumber sinyal ini dapat berupa power supply, function generator frekwensi sinteser atau yang sejenisnya.
- Instrument pengukur, yang akan melakukan pengukuran pada titik titik uji pada UUT, dimana dapat berupa ADC, frekwensi counter, digital multimeter, atau jenis pengukur yang lainnya.

- System Switching, yang akan menentukan jalan sinyal antara UUT dan bagian lain dalam sistem ATE.
- Operator machine Interface, untuk melakukan komunikasi controller dengan operator. Mungkin saja bagian ini berada pada bagian controller, yang akan memperlihatkan keadaan dari sistem tes yang dilakukan seperti keadaan switch, hasil output yang ditampilkan pada printer dan lain sebagainya.
- UUT interface , Unit Under Test yang akan diuji pada sistem ATE.

Jika dilihat alur tes yang dipergunakan dari suatu tes otomatis ada berapa peralatan pada tugas akhir ini yang akan dipergunakan, dimana ditekankan pada proses interfacing dari suatu peralatan penguji ke sebuah bus yang dalam hal ini adalah GPIB, sehingga peralatan tersebut dapat dipergunakan sebagai sistem ATE. Adapun peralatan yang akan dipakai atau di interface-kan kedalam bus GPIB adalah : Sebuah mikrokomputer sebagai controller dan operator kontrolnya, sebuah unit function generator yang diinterfacekan dengan GPIB, unit switching yang diinterfacekan dengan GPIB dan sebuah ADC yang diinterfacekan dengan GPIB.

Proses ATE dilakukan pada mikrokomputer yang dilengkapi dengan sebuah card GPIB sehingga dapat mengontrol semua kegiatan transfer data. Proses perhitungan data akan dipegang oleh komputer sebagai operator console yang mendisplaykan hasil hasil dan kondisi sistemnya.

BAB III

PERENCANAAN HARDWARE

Seperti telah dijelaskan pada bab terdahulu, suatu sistem tes otomatis yang dikendalikan oleh GPIB dapat terdiri dari :

- GPIB Controller sebagai pengendali.
- Panel Connector untuk mengubah konektor GPIB ke paralel DB 25
- Unit Switching dengan interface GPIB untuk menghubungkan UUT dengan modul tester
- Sinyal Generator dengan interface GPIB sebagai pembangkit gelombang
- ADC / multi probe dengan interface GPIB sebagai pengambil sinyal yang diukur.
- Multimeter dengan interface GPIB
- Frekwensi Counter dengan interface GPIB

Pada tugas akhir ini hanya dibatasi untuk proses pengiriman data dengan menggunakan bus GPIB untuk mengontrol modul Sinyal Generator, modul ADC sebagai multi probe dan modul Switching.

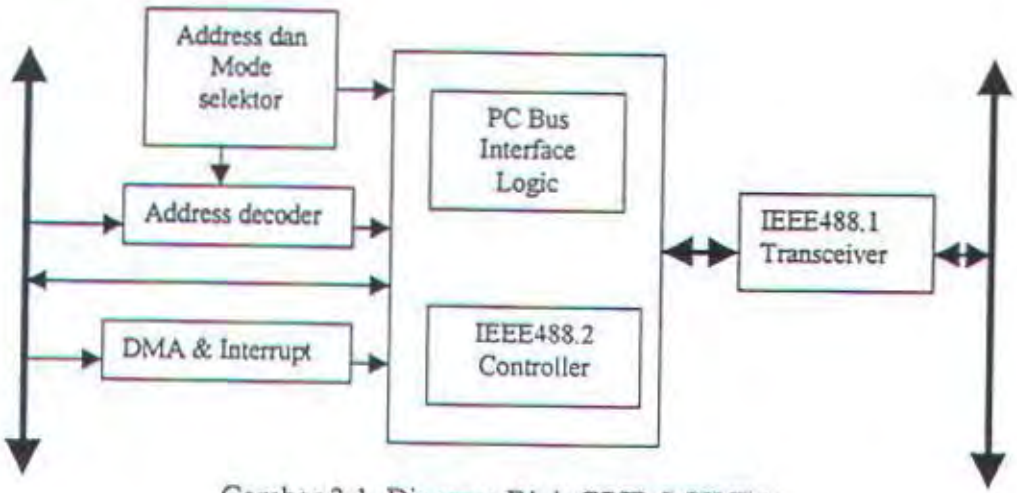
Komunikasi yang ada pada sistem ini adalah satu arah untuk komputer sebagai controller dengan Sinyal Generator dan Unit Switching. Sedangkan untuk komunikasi antara ADC dengan komputer adalah dua arah dimana dalam proses inisialisasi dan kontrol fungsi dari ADC, ADC berfungsi sebagai Listener sedangkan untuk transfer data hasil pengukuran dari ADC ke komputer ADC berfungsi sebagai Talker.

Untuk contoh implementasinya dipergunakan untuk melakukan pengukuran respon suatu rangkaian sehingga dapat dilihat karakteristik dari rangkaian tersebut. Tapi implementasinya masih menggunakan komputer sebagai controller dan display dari hasil pengukuran.

3.1. GPIB Controller Dengan GPIB PC II/IIA

Untuk melakukan interface antara bus standar IEEE488 dengan komputer maka dipergunakan sebuah card interface board GPIB. Dimana interface ini menggunakan chip controller TNT 488-2C atau NAT 4882, dimana chip ini memegang banyak fungsi sebagai controller dan sebagai interface lain yaitu Listener pada saat menerima data dari bus dan sebagai Talker pada saat memberikan data byte kepada bus.

Blok Diagram dari card interface GPIB-PCII/IIA nampak seperti pada gambar dibawah ini

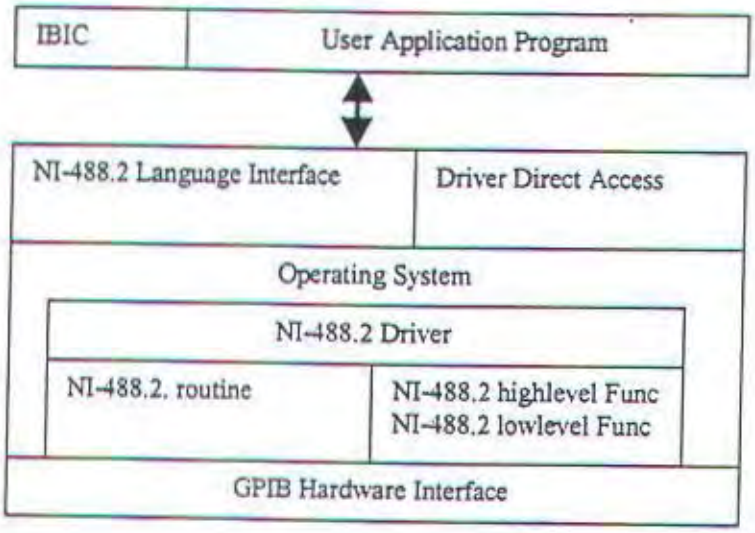


Gambar 3.1. Diagram Blok GPIB-PCII/IIA

Interface board ini dapat dikonfigurasi sebagai GPIB-PCII atau sebagai GPIB-PCIIA. Tergantung kepada setting jamper pada cardnya.

3.1.2. Konfigurasi Perangkat Lunak

Perangkat lunak yang dipergunakan untuk mengontrol card GPIB-PCII/IIA menggunakan software dari National Instrument NI-488.2. Adapun struktur dari software ini adalah sebagai berikut.



Gambar 3.2 Struktur NI-488.2

Seperti terlihat pada gambar diatas maka program yang dipergunakan untuk mengendalikan suatu peralatan GPIB yang terhubung pada card ini dengan software NI-488.2 diperlukan program pendukung yang sudah disediakan yang berupa prosedur prosedur dan driver yang telah ada.

Program dibuat dengan User Application Program yaitu bahasa pemrograman tingkat tinggi seperti C, Pascal, Basic dan sebagainya. Dalam hal ini dipergunakan bahasa pemrograman dengan Borland Delphi dengan alasan visualisasinya lebih cepat dan masih kompatibel dengan bahasa pemrograman Pascal yang telah dikuasai. Selain itu pada Delphi sudah disediakan routine routine yang

dipergunakan untuk mengakses card GPIB secara langsung dengan menggunakan routine dan function yang sudah disediakan pada driver cardnya.

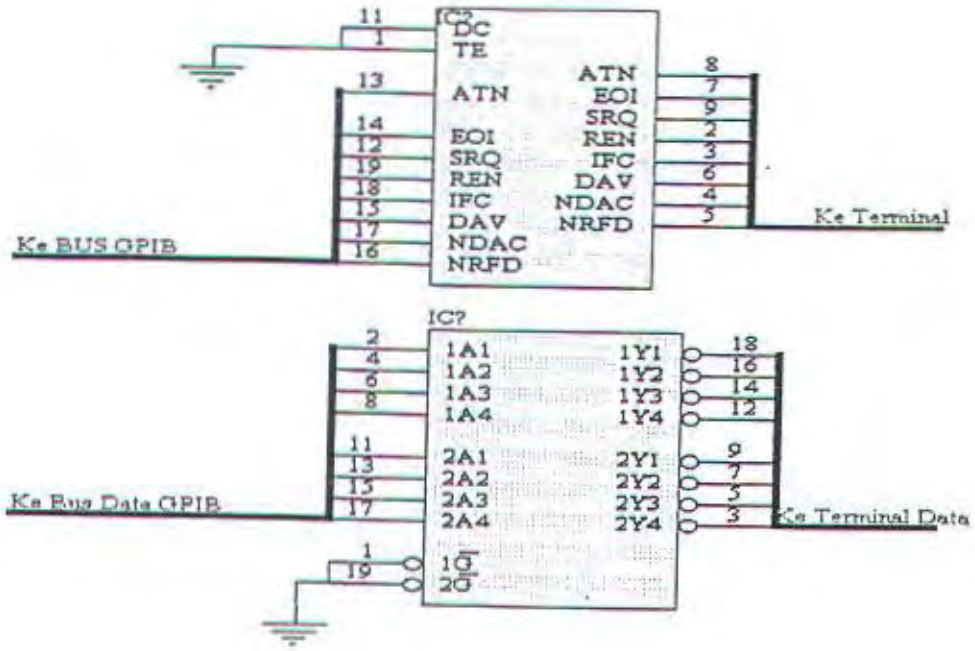
3.2. Unit Switching Dengan interface GPIB

Seperti yang telah dijelaskan Unit Switching mempunyai 2 bagian pokok yaitu rangkaian interfacenya dan bagian rangkaian switchnya sendiri. Dimana pada rangkaian switch nya menggunakan rangkaian switch analog dengan IC CMOS. Sedangkan untuk rangkaian interfacenya sendiri terdiri dari beberapa bagian yaitu rangkaian buffer untuk interface bus GPIB, rangkaian kontrol interface GPIB, dan rangkaian buffer switchnya. Rangkaian switching ini jika dilihat pada blok ATE maka disini akan difungsikan hanya sebagai Listener saja.

3.2.1. Rangkaian Buffer Bus GPIB

Rangkaian buffer interface GPIB ini berfungsi sebagai buffer dari bus ke peralatan. Buffer ini diperlukan untuk menyesuaikan level logika pada bus GPIB dengan level logika TTL yang dipakai pada rangkaian berikutnya. Rangkaian ini direalisasikan dengan menggunakan 2 buah IC yaitu IC 74LS240 sebagai buffer jalur data, dimana selain sebagai buffer juga membalik level logikanya sehingga menjadi logika normal untuk dipergunakan pada rangkaian berikutnya, IC DS76161 yang merupakan IC transceiver GPIB yang berfungsi untuk menyesuaikan taraf logika TTL menjadi taraf logika GPIB sehingga menjadi sesuai dengan taraf logika TTL dan sebaliknya. Namun tidak dilakukan

pembalikan taraf logika, karena pada rangkaian berikutnya bisa dimodifikasikan pada rangkaian kontrol.



Gambar 3.3 Rangkaian Buffer GPIB

3.2.2. Rangkaian Kontrol interface GPIB

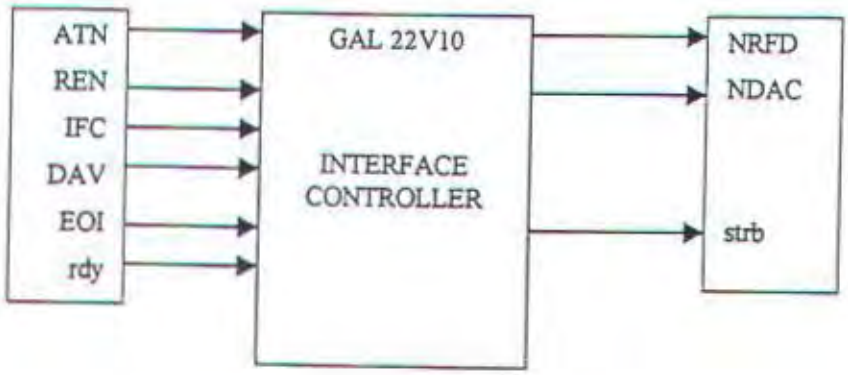
Rangkaian kontrol interface GPIB merupakan rangkaian yang akan mengontrol kerja dari peralatan dibawahnya. Rangkaian mempunyai beberapa bagian yaitu massage decoder, interface function dan device function. Dimana ketiga bagian tersebut diwujudkan dengan menggunakan teknologi PLD dengan menggunakan IC 22V10 yang bisa mengatasi fungsi fungsi tersebut.

Ketiga bagian tersebut kemudian diimplementasikan menjadi 2 buah bagian yaitu: function control dan Data control.

3.2.2.1. Function Control.

Function control terdiri dari 2 fungsi yaitu menterjemahkan fungsi fungsi kontrol yang dikirim oleh kontrol GPIB dan mengatur proses Handshaking dari peralatan dengan bus GPIB sehingga dapat berkomunikasi. Function control hanya memerlukan sebuah IC 22V10, dan dapat melakukan tugas tugasnya.

Signal signal bus yang mempengaruhi kerja dari kontrol interface ini adalah ATN,REN,IFC,DAV,EOI dan local message rdy. Dan untuk menterjemahkan fungsi yang harus dilakukan / message decoder maka 7 bit signal Data yang mempengaruhinya. Jika dibuat blok dari rangkaian kontrol ini adalah sebagai berikut :



Gambar 3.4 Blok kontrol GPIB

Sedangkan untuk sinyal hasil dari kontrol ini adalah NRFD dan NDAC yang digunakan untuk mengatur proses handshaking dengan talker. Sehingga yang sangat mempengaruhi sinyal ini adalah DAV. Selain sinyal diatas juga dihasilkan sinyal lain untuk mengontrol local control yaitu Strb dan GTL.

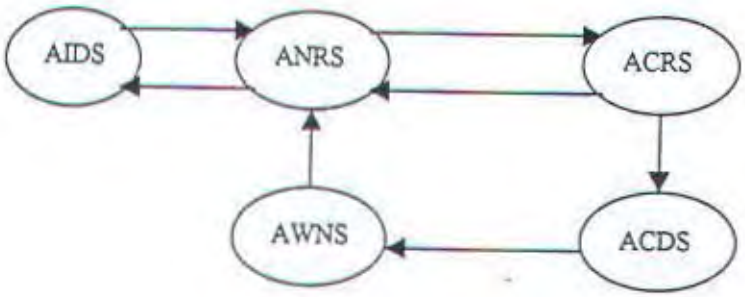
Alur dari proses kontrol ini sangat dipengaruhi oleh State dari Listener yaitu AH State, dan L state. Dalam hal ini kedua state tersebut dijadikan satu.

Dilihat dari alur handshaking dan kedua state tersebut maka dapat dibuat sebuah state machine untuk mengontrol proses interface-nya dengan bus dan dengan data control dibawahnya.

Fungsi fungsi yang didukung dalam state machine ini adalah:

- MLA dengan kode 01XXXXX dimana X adalah alamat interface
- UNL dengan kode 011111111
- AH State untuk NRFD dan NDAC
- L State untuk control AH state nya.

Adapun implementasi state machine-nya adalah sebagai berikut:



Gambar 3.5. AH-State Machine

Dilihat dari state tersebut maka dapat dibuat kode PLD untuk GAL22V10. Dalam hal ini yang dipergunakan adalah bahasa VHDL. Listing dari kode VHDL yang dipergunakan untuk kontrol ini secara lengkap dilampirkan pada lampiran.

3.2.2.2 Data Control

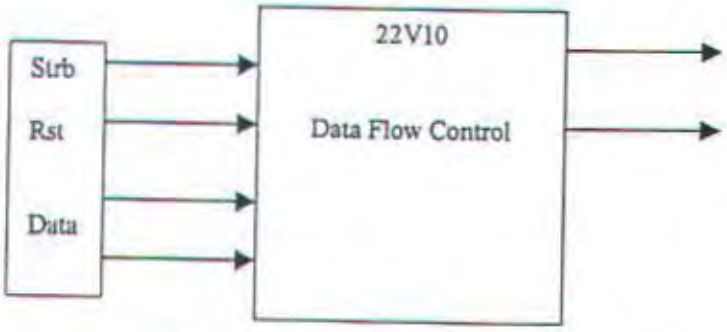
Fungsi dari data control ini adalah sebagai pengontrol proses pengontrolan dari switch yang dipergunakan dimana yang dikontrol adalah proses switching

input terpilih ke output terpilih. Sinyal yang mempengaruhi adalah sinyal dari interface control yaitu Strb dan sinyal dari bus yaitu Data dan IFC untuk melakukan proses reset. Pengontrolan yang dilakukan adalah pengontrolan terhadap data kanal yang dipilih dan data group yang dipilih.

Secara garis besar Data control ini mempunyai bagian bagian sebagai berikut:

- Latch decoder group untuk mendekodekan byte yang merupakan data untuk memilih kanal yang akan dikontrol, dimana pada bagian ini dengan kontrol ini data pada bus merupakan data kanal yang diinginkan.
- Timing latch control untuk mengatur waktu latch data dari data bus kedalam latch switch untuk menggerakkan switch digital yang dipergunakan.

Blok diagram dari Data control ini adalah sebagai berikut



Gambar 3.6. Blok diagram Data Control

Format data yang didukung oleh data control ini adalah sebagai berikut:

Dataselect	Group	Data

Keterangan :

- Data Select : Sistem pengaturan yang diinginkan yaitu secara berurutan dari group pertama sampai group terakhir atau salah satu dari group yang ada saja.
- Group : Nomer group yang akan diatur.
- Data : Data dari group yang diatur dimana D6-D7 untuk data saluran ke instrument, D0-D2 untuk data saluran ke UUT pada Group Input dan Output, sedangkan D0-D7 berupa data saluran Group saluran Reserve.

Dilihat dari format data tersebut maka dapat dilakukan pengontrolan secara bersama dan terpisah antar tiap group sehingga tiap group tidak saling mempengaruhi dan kontrol dapat dilakukan secara acak untuk tiap kanalnya.

3.2.2 Rangkaian Digital Switch

Rangkaian ini terdiri dari 2 bagian yaitu rangkaian latch yang berfungsi untuk menyimpan data switch yang aktif dan rangkaian digital switch yang terdiri dari array switch analog yang dikontrol secara digital. Adapun blok diagram dari rangkaian digital switch ini lengkapnya seperti pada lampiran

3.3. Sinyal Generator Dengan interface GPIB

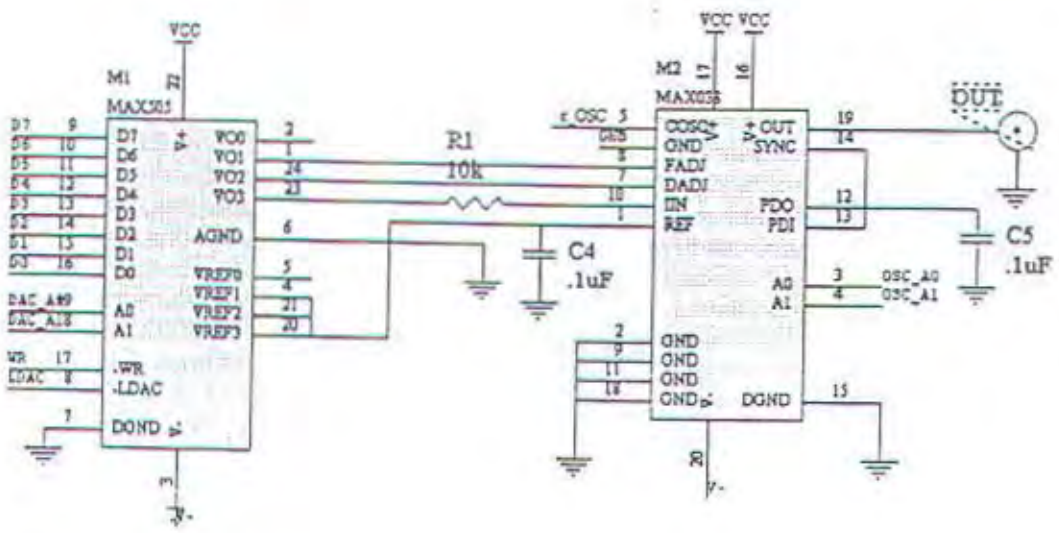
Dalam disain ini akan dibuat suatu sistem interface yang dipergunakan untuk mengontrol sebuah function generator. Untuk bagian sinyal generator disini dipergunakan suatu rangkaian yang sudah ada dan dalam hal ini bukan bagian yang diutamakan, tapi lebih mengutamakan kepada cara pengontrolan dari

function generator tersebut dengan menggunakan GPIB sehingga dapat dipergunakan pada system tes. Seperti halnya Unit Switching akan Sinyal generator juga terdiri dari dua bagian utama yaitu bagian sinyal generator yang akan dikontrol dan bagian interface nya.

3.3.1. Sinyal Generator

Untuk rangkaian sinyal generator dipergunakan komponen IC MAX038 dimana kontrol fungsinya dibuat digital. Dimana yang dapat diatur adalah bentuk gelombangnya, range frekwensi, deviasi frekwensi terhadap pusatnya dan duty cycle serta attenuator amplitude output. Untuk keperluan tersebut maka diperlukan komponen pendukung seperti analog multiplexer dan DAC.

Rangkaian dari sinyal generator secara lengkap terdapat pada lampiran, sedangkan secara garis besar rangkaian sinyal generator MAX038 ditunjukkan dibawah ini :



Gambar 3.7. Rangkaian Generator MAX038

Pengaturan dari fungsi sinyal generator dilakukan sebagai berikut:

- Pengaturan frekwensi tengah dilakukan dengan memberikan sumber arus pada kaki IIN MAX038 dimana arus ini didapat dari sebuah DAC voltage output yang diseri dengan R sehingga menghasilkan arus yang besarnya dapat diatur secara digital. Dimana persamaan untuk besarnya frekwensi ini adalah sebagai berikut: $Fo = Vin / (R * Cf)$

  Dimana:

  Fo = Frekwensi fundamental yang dihasilkan.

  Vin = tegangan input

  R = Resistor seri untuk membatasi arus yang masuk.

  Cf = capasitor yang terhubung pada rangkaian resonansi internal

  Dengan batasan nilai IIN yang masuk : $2uA < IIN < 750 uA$.

- Pengaturan deviasi frekwensi dilakukan dengan memberikan tegangan pada kaki Fadj, dimana tegangan ini didapatkan dengan sebuah DAC voltage output. Perubahan / pergeseran frekwensi output yang dapat diatur secara digital dan megikuti persamaan : $Fx = Fo * (1-,,,)$

  Dimana :

  Fx = frekwensi output

  Fo = frekwensi fundamentalnya

  Vin = tegangan input.

  Dengan batasan nilai Vin : $-2.3 V < Vin < 2.3 V$

- Pengaturan Duty Cycle dilakukan dengan memberikan tegangan pada kaki Dadj yang didapatkan dari sebuah DAC voltage output. Pengaturan Duty Cycle secara penuh dapat diatur dari 15 % - 85 % dengan keadaan normal pada saat Dadj = 0V menghasilkan Duty Cycle 50%. Perubahan Duty Cycle diatur dengan memperhatikan persamaan : DC = 50% -

  Dimana :

- Pengaturan range frekwensi dan Attenuator dilakukan dengan memberikan switch digital pada kontrolnya masing masing sehingga dapat diatur digital. Dimana range frekwensi diatur 3 range yaitu :

  Range Low : 10 Hz – 50 KHz

  Range Mid : 5 KHz – 5 MHz

  Range High: 2 MHz – 15 MHz

Pengaturan dengan DAC dilakukan dengan menggunakan IC MAX505 yang merupakan DAC dengan 4 channel output voltage, internal multiplexer dan internal buffer, Sehingga tidak lagi diperlukan latch untuk menyimpan data yang diatur.

3.3.2. Rangkaian Interface

Untuk mengatur kerja dari sebuah function generator seperti diatas maka diperlukan sebuah kontrol yang dapat diprogram untuk pengontrolan sinyal generator. Untuk ini bagian Interface dibagi 2 bagian yaitu bagian interface ke bus

dan bagian data control dimana kedua kontrol tersebut seperti halnya unit switch menggunakan teknologi PLD dengan pemrograman VHDL.

3.3.2.1 Rangkaian Interface GPIB

Rangkaian interface GPIB ini berfungsi untuk mengatur proses handshaking dengan bus sehingga dapat berkomunikasi dengan yang lain. Dalam hal ini modul function generator difungsikan sebagai Listener saja yaitu hanya sebagai penerima kontrol saja.

Alur dari proses kontrol ini sangat dipengaruhi oleh State dari Listener yaitu AH State, dan L state. Dalam hal ini kedua state tersebut dijadikan satu. Dilihat dari alur handshaking dan kedua state tersebut maka dapat dibuat sebuah state machine untuk mengontrol proses interface-nya dengan bus dan dengan data control dibawahnya.

Fungsi fungsi yang didukung dalam state machine ini adalah:

- MLA dengan kode 01XXXXX dimana X adalah alamat interface
- UNL dengan kode 011111111
- AH State untuk NRFD dan NDAC
- L State untuk control AH state dan transfer datanya
- GTL dengan kode 0000001
- LLO dengan kode 0010001

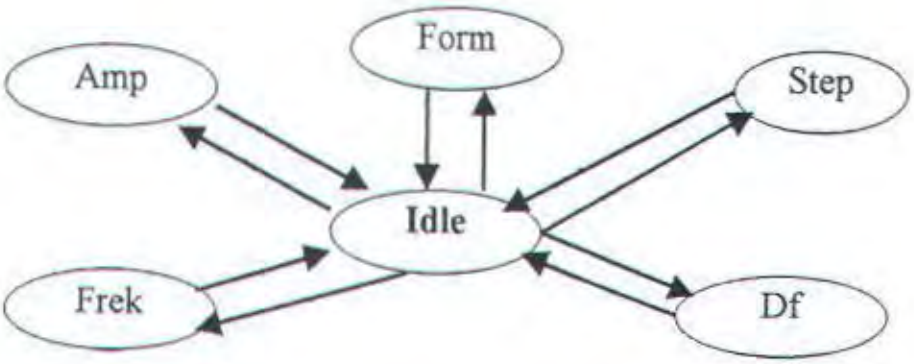
Adapun implementasi state machine-nya adalah sama seperti pada control interface switch unit karena mempunyai fungsi yang sama sebagai listener.

Dilihat dari state tersebut maka dapat dibuat kode PLD untuk GAL22V10. Dalam hal ini yang dipergunakan adalah bahasa VHDL. Listing dari program VHDL yang dipergunakan untuk kontrol ini secara lengkap dilampirkan pada lampiran.

3.3.2.2 Rangkaian Data Control

Fungsi dari bagian ini adalah mengatur alur data dan latch pada modul function generator. Sinyal yang mempengaruhi pada bagian ini adalah Strb dan GTL dari interface control, IFC dan Data dari bus. Pengontrolan yang dilakukan adalah bentuk gelombang, besarnya frekwensi, range frekwensi, deviasi frekwensi, Duty cycle dan attenuator. Dimana pengaturan tiap tiap parameter tersebut dapat dilakukan secara bersama sekaligus atau secara terpisah satu persatu secara acak. Penentuan pengaturan tersebut terjadi pada saat state Idle, pada state ini data bus akan diterjemahkan sebagai kode pengatur yaitu :

Jika data yang ada adalah "000" maka pengaturan yang dilakukan adalah berurutan, dan jika yang lain maka pengaturan hanya satu kali sesuai kodenya.



Gambar 3.8. State machine Data Control

Untuk mewujudkan proses kontrol tersebut maka dapat dibuat alur control yang diimplementasikan dengan sebuah state machine. Dimana untuk state machine nya akan mendukung data format seperti dibawah , dan state machine dari kontrolnya adalah seperti pada gambar 3.8 .

Tabel 3.1. Format Data Function Generator

CW	Form	Frek	Df	DutyC	Range

Listing dari program VHDL nya secara lengkap terdapat pada lampiran.

3.4. Rangkaian ADC Dengan interface GPIB

Untuk melakukan pengukuran / probe pada titik titik uji maka diperlukan suatu rangkaian untuk hal tersebut. Pada disain ini akan dibuat suatu rangakaian ADC 2 channel dimana 1 channel bisa berfungsi untuk probe digital 8 bit data serial. Dengan tujuan sebagai fungsi tambahan jika ingin dipergunakan sebagai logic analyzer.

Rangkaian ADC ini dirancang sehingga frekwensi sample-nya dapat diprogram, dengan menggunakan sebuah pembagi frekwensi terprogram PIT8254. Selain itu mempunyai sebuah signal output yang dapat diprogram pula besar frekwensinya. ADC yang diinterfacekan ke bus GPIB ini mempunyai 3 bagian yaitu pada rangkaian attenuator digital, rangkaian ADC, dan rangkaian interface GPIBnya.

3.4.1. Rangkaian Attenuator

Rangkaian Attenuator disini didisain sehingga pada output-nya mempunyai tegangan antara 0-1024 mV. Hal ini disebabkan karena pada input ADCnya input analog yang diijinkan adalah 1024 mV. Pelemahan yang dilakukan terdapat 4 tingkat dengan menggunakan 4 buah array resistor yang dihubungkan dengan sebuah multiplexer analog 4051, dimana IC 4051 ini mempunyai 3 bit selektor sehingga hanya memerlukan 3 bit kontrol untuk melakukan pengaturan attenuator. Sedangkan untuk penguatnya dipergunakan sebuah Op-Amp dengan band witdh 100 Mhz dengan harapan untuk sinyal frekwensi tidak terjadi pelemahan yang nggak diinginkan akibat band witdh Op-Amp.

Pengaturan penguatan yang dipakai adalah :

- Konfigurasi penguatan adalah menggunakan inverting amplifier dengan harapan penguatan bisa diatur dibawah -1.
- Range yang dipakai adalah 8 step yaitu :

  Untuk input max 20 V dikuatkan - 1/20 kali

  Untuk input max 10 V dikuatkan - 1/10 kali

  Untuk input max 5 V dikuatkan - 1/5 kali

  Untuk input max 2 V dikuatkan - 1/2 kali

  Untuk input max 1 V dikuatkan -1 kali.

  Untuk input max 0.5 V dikuatkan –2 kali

  Untuk input max 0.1 V dikuatkan -10 kali

  Untuk input max 20 mV dikuatkan - 50 kali

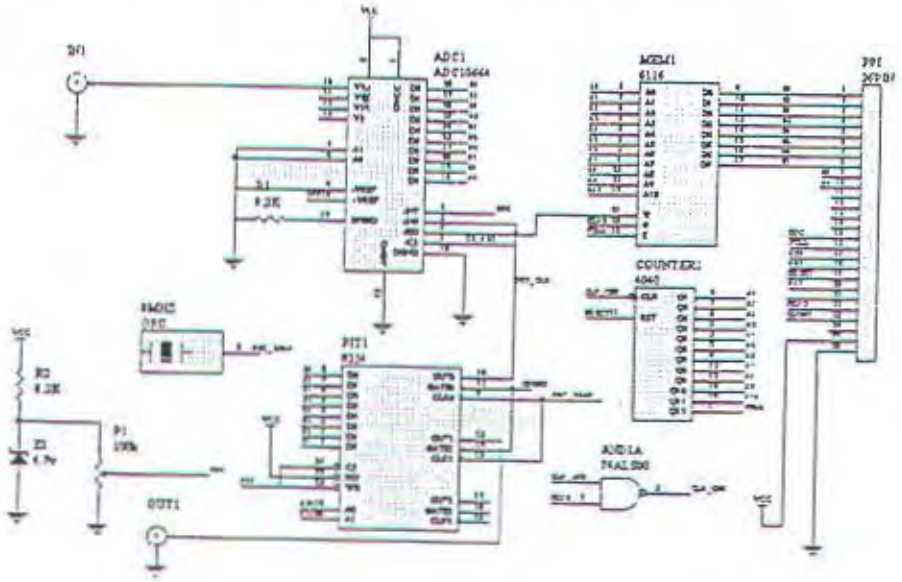
3.4.2. Rangkaian ADC

Rangkaian ADC yang digunakan adalah dengan menggunakan ADC semi flash dengan resolusi 10 bit. Namun bit yang dipakai adalah 8 bit saja, dengan pertimbangan pemilihan ADC ini adalah dari segi kecepatan yang cukup yaitu 360ns. Namun untuk transfer data dari ADC ke bus adalah kira kira 1 Mhz , sehingga dipergunakan sebuah RAM yang dipergunakan untuk menyimpan sementara data hasil konversi sebelum ditransfer ke bus.

Konfigurasi yang dipakai dalam disain ini adalah ADC yang berjalan stand alone dan akan selesai melakukan konversi jika RAMnya sudah penuh dan akan

`memberikan sinyal ke Bus. Untuk keperluan tersebut maka diperlukan beberapa komponen yaitu :

- Sebuah PIT8254 yang berfungsi memberikan clock konversi ke ADC, dimana PIT akan berjalan jika pada input gatenya diberikan signal high.
- Sebuah RAM Statis yang menyimpan data konversi selama ADC melakukan konversi, sehingga kecepatan konversi bisa dimaksimalkan.
- Sebuah penghitung 12bit untuk pengalamatan data di RAM.
- Sebuah oscillator 4 MHZ sebagai clock referensi untuk PIT.



Gambar 4.10 Blok skema ADC

Besarnya clock referensi dipilih 4 MHZ karena ADC memiliki timing start konversi selebar minimal 150 ns sedangkan untuk 4 MHZ memiliki lebar pulsa 250 ns sehingga tidak akan mempengaruhi start konversi dan kecepatan konversi sebesar 360 ns.

Waktu pengambilan data dari ADC ke RAM dilakukan dengan prosedur : data ADC n-1 akan masuk ke alamat n pada RAM karena sinyal WR dijadikan satu dengan Start konversi.

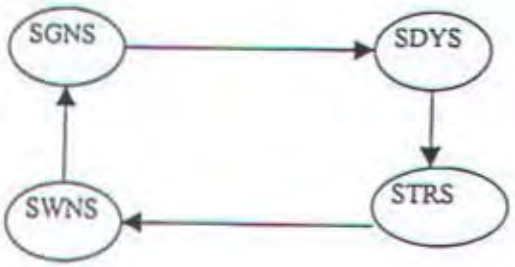
3.4.3 Rangkaian Interface GPIB

Pengaturan Kerja dari ADC dilakukan dengan menggunakan kontrol dengan disain VHDL, dimana ADC jika dilihat fungsinya akan berfungsi sebagai Talker. Namun untuk inisialisasi awal dan pengontrolan sampling rate nya maka akan berfungsi sebagai listener. Sehingga disain control harus dapat mendukung dua fungsi tersebut kedalam sebuah interface.

Untuk keperluan interface ini maka diperlukan 2 bagian yang berfungsi sebagai kontrol handshaking dan kontrol datanya.

3.4.3.1. Interface function

Interface kontrol harus mendukung 2 fungsi sekaligus sehingga 2 buah state interface harus digabungkan yaitu AH state dan L state sebagai Listener dengan SH state dan T state pada saat sebagai Talker. Sedangkan untuk mengdukung interupt hanya menggunakan sistem polling secara tetap karena address dari talker hanya satu yaitu pada ADC. Adapun State machine untuk pengontrolan handshaking adalam sebagai berikut :



Gambar 3.10 State Machine SH State

Untuk kode VHDLnya secara lengkap ada pada lampuiran.

3.4.3.2. Data control function

Data control function berfungsi untuk mengatur proses inisialisasi yang akan dilakuakan oleh PIT yang berfungsi sebagai pengatur frekwensi sampling dari ADC. Selain itu data control berfungsi untuk mengatur fungsi kerja dari ADC apakan bekerja sebagai logic analizer atau dengan input analog. Untuk proses pengaturan ini maka dibuat state machine sebagai berikut :

dengan kode VHDL yang lengkap terdapat pada lampiran.

BAB IV

PERENCANAAN SOFTWARE

Software yang dirancang disini dibatasi untuk memanajemen proses pengiriman data baik dari talker ke komputer sebagai listener, komputer ke listener dan dari controller ke listener. Kemudian menampilkan proses yang terjadi sebagai respon dari instrumen instrumen yang diatur. Sebagai contoh aplikasinya adalah digunakan untuk mengatur alur pengujian suatu peralatan elektronika sehingga dapat diketahui respon dari peralatan tersebut. Baik respon DC nya, AC dan transiennya.

Dalam perencanaan ini menggunakan bahasa pemrograman Borland Delphi dengan alasan Visualisasinya dan penguasaan penulis terhadap bahasa pemrograman. Sedangkan untuk rutin rutin dan fungsi IO tingkat rendah menggunakan library yang ada pada driver card GPIB.

4.1. Fungsi dan Rutin Pengendali GPIB

Seperti yang ditulis di atas pengendalian tingkat rendah ke bus GPIB menggunakan library pada driver cardnya. Fungsi tingkat rendah tersebut merupakan Board Level dan tidak semua fungsi yang ada pada lib tersebut dapat diaplikasikan ke sistem yang dirancang, karena kemampuan dari suatu instrumen untuk meresponnya. Adapun fungsi dan rutin pengendali yang dipakai adalah :

- ☑ **DLLSendIFC (bd)**: Suatu fungsi yang digunakan untuk menginisialisasi bus GPIB ke kondisi yang diketahui, seperti halnya reset pada suatu sistem.
- ☑ **DLLFindLstn(bd,addr[],reslt[],limit,ibstat,iberr,ibcntl)**: Suatu fungsi yang digunakan untuk mengenali peralatan yang terhubung pada bus dan hasil deteksinya didapatkan pada variable reslt.
- ☑ **DLLDevClear(bd,addr,ibsta,iberr,ibcntl)** : Fungsi yang digunakan untuk mereset suatu peralatan yang terhubung ke bus sehingga statenya diketahui.
- ☑ **DLLReceive(bd,addr,data[],count,eot,ibsta,iberr,ibcntl)** : Rutin yang digunakan untuk menerima data dari suatu instrument yang terhubung dengan bus dan komputer bertindak sebagai listener.
- ☑ **DLLSend(bd,add,data[],count,eot,ibsta,iberr,ibcntl)** : Rutin untuk mengirim data dari komputer sebagai talker ke instrumen sebagai listener.
- ☑ **DLLTrigger(bd,addr,ibsta,iberr,ibcntl)**: Fungsi yang dipergunakan untuk memerintahkan suatu instrument untuk segera melakukan suatu tugas yang telah difungsikannya.
- ☑ **DLLFindRQS(bd,addr[],reslt[],ibsta,iberr,ibcntl)**: Rutin yang dipergunakan untuk mengetahui peralatan yang membutuhkan pelayanan interupt.

4.2. Procedure Pengendali Modul

Low level programming untuk mengendalikan modul modul tersebut terbagi menjadi 3 procedure sesuai dengan modul yang dipakai. Perbedaaan yang mendasar adalah alamat dan format data yang dikirimkan tapi menggunakan proses / langkah

langkah yang sama. Semua fungsi dan rutin tersebut telah didefinisikan dalam sebuah file dengan nama gpib.dll, yang menangani semua tugas.

4.2.1 Procedure Pengendali Unit Switching

Procedure pengendali untuk unit switching terdiri dari pengalamatan instrumen, mencabut alamat instrumen dan pengiriman data dari komputer ke switching. Procedure tersebut langsung mengambil dari librari. Dimana unit ini akan merespon perintah : FindLstn, SendIFC, Send.

Perintah perintah tersebut akan direspon. Sedangkan untuk pengiriman data ke unit switch harus memperhatikan format data yang direspon yang terdapat 2 mode pengiriman data yaitu :

Mode data 2 byte : byte pertama adalah data group yang dipilih, data byte kedua adalah data channel yang aktif.

Mode data 4 byte : byte pertama adalah data awal pengiriman data dimana bit 0 dan bit 1 selalu berharga "00". Sedangkan data berikutnya adalah berurut urutan: data channel group input, channel group output dan channel group reserve.

Langkah langkah pengirimannya adalah :

- Menginisialisasi sistem interface keperalatan yang terhubung ke bus dengan mengirim sinyal IFC yaitu : DLLSendIFC(0) ; dimana (0) adalah board GPIB
- Mengirim data channel dan group dengan menggunakan mode 2 byte atau mode 4 byte dengan perintah : DLLSend(0,data,mode,ibstat,iberr,ibcntl).

- Mengirim perintah Unlisten agar unit tidak ikut menerima data lagi dengan menggunakan perintah DLLSendCmds(0,Cmds,count,ibstat,iberr,ibcntl).

4.2.2. Procedure Pengendali Unit Function Generator

Format data untuk mengendalikan Unit function Generator juga mengijinkan untuk menggunakan 2 mode format pengiriman data yaitu :

Mode data 2 byte : byte pertama adalah data yang akan diatur dan byte kedua adalah data programming.

Mode data 6 byte : byte pertama sebagai tanda mode yang dipakai yang selalu mempunyai kondisi bit 0,1 bernilai "00".

Langkah langkah pemrogramannya adalah sebagai berikut:

- Mengirim perintah inisialisasi /reset dengan mengirim IFC aktif dengan menggunakan perintah: DLLSendIFC(0).
- Mengirim data mode dan selanjutnya megirim data programming dengan menggunakan perintah :DLLSend(0,data,mode,ibstat,iberr,ibcntl).
- Mengirim perintah Unlisten agar unit tidak ikut menerima data lagi dengan menggunakan perintah DLLSendCmds(0,Cmds,count,ibstat,iberr,ibcntl).

4.2.3. Procedure Pengendalian Unit ADC / Multiprobe

Procedure untuk mengendalikan ADC terdiri dari 2 bagian yaitu untuk inisialisasi PIT untuk memberikan data programming untuk frekwensi samplingnya

dan fungsi general dari ADC. Jika telah terinisialisasi ADC menunggu untuk melakukan pengukuran dengan menunggu perintah trigger dan devclear.

Langkah pengiriman data dan perintah untuk ADC adalah sebagai berikut:

- Menginisialisasi bus dengan memberikan sinyal IFC dengan perintah sendIfc yaitu: DLLSendIFC(0).
- Mengirim data programming untuk data frekwensi sampling dan data programming fungsi dari ADC dengan memberikan perintah seperti unit yang lain tapi dengan format data yang lain DLLSend(0,data,count,ibstat,iberr,ibcntl).
- Memberikan perintah unlisten ke ADC atau langsung memberikan perintah talk untuk mengaktifkan fungsi talknya.
- Memberikan perintah trigger untuk memerintahkan ADC start melakukan pengukuran: DLLTrigger(0,addr,ibstat,iberr,ibcntl).
- Menunggu ADC menyelesaikan pengukuran dan memerintahkan mengirim data pengukuran ke komputer sebagai listener: DLLReceive(0,addr,data[],count,eot,ibstat,iberr,ibcntl).
- Memberikan perintah DevClear untuk mengembalikan kondisi ADC pada state data awal dan pada channel 2: DLLDevClear(0,addr,ibstat,iberr,ibcntl).
- Memberikan perintah mengirim data lagi dengan menggunakan perintah DLLReceive(0,addr,data[],count,eot,ibstat,iberr,ibcntl).
- Memberikan perintah Untalk sebagai perrintah akhir pengiriman.

4.3. Software Managemen ATE

Proses pengaturan alur data untuk melakukan tes equipment adalah dengan mengatur besarnya sinyal yang masuk ke switch, memilih input sinyal yang diinginkan, memilih node yang akan diukur memberikan sinyal sample ke peralatan pengambil data dan mengirim data hasil pengukuran ke komputer sebagai listener.

4.3.1 Procedure Pengambilan Data Respon DC

Program untuk mengambil respon DC adalah dengan memberikan input UUT ke GND dan memilih node yang diukur ke ADC. Proses yang dilakukan adalah:

Mengatur posisi switch sehingga input ada pada GND.

Mengatur posisi Switch sehingga output berada pada node yang diinginkan dan mengambil sample dan mengirim ke komputer.

4.3.2 Procedure Pengambilan Data Respon AC

Program untuk mengambil respon AC adalah dengan memberikan input UUT dengan output sinyal dari Function generator, node UUT yang akan diukur diberikan pada outgroup switch dan diberikan ke ADC untuk diambil datanya. Proses yang dilakukan adalah :

- Sinyal Function generator dinaikkan perstep dan tiap kenaikan tersebut ADC melakukan sampling pada node yang diinginkan dan input sinyalnya.
- Mengambil amplitude dari tiap step kenaikan frekwensi.

- Membandingkan fase sinyal input dengan sinyal node yang diukur.

4.3.3 Procedure Pengambilan Data Transien

Program untuk mengambil respon transien adalah dengan memberikan pulsa dengan periode tetap dan ADC mengambil data secara periodik pula dengan asumsi periode pengambilan data dilakukan hanya pada saat pulsa ada pada posisi high. Frekwensi sampling dari ADC harus lebih ganjil dan pulsa input harus Genap sehingga waktu sampling seolah olah diperkecil dan dilakukan secara periodik dan sinkron sehingga sinyal yang disampling kelihatan diam.

BAB V

TES DAN PENGUJIAN MODUL

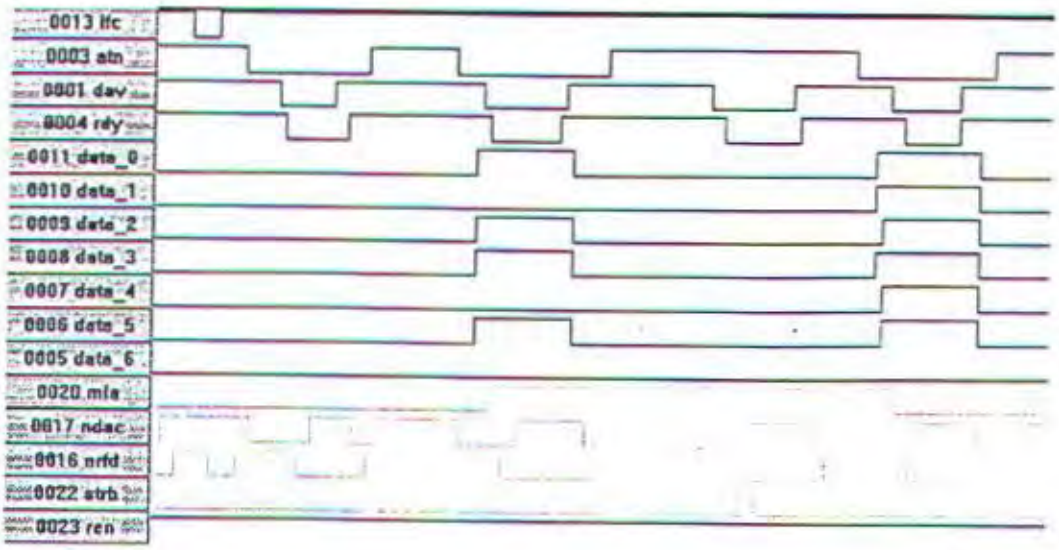
5.1. Pengujian Sistem Interfacing GPIB

Proses pengujian ini meliputi timing diagram yang ada pada bus GPIB dengan timing diagram yang terjadi pada peralatan, baik itu untuk interface functions maupun untuk device control functions.

5.1.1. AH – State Functions dan L – State Functions

Seperti yang telah diuraikan diatas suatu peralatan GPIB yang hanya untuk Listener saja bisa saja terdiri dari AH function untuk handshake ke controller atau Talker dan L function untuk State instrumen-nya sendiri dan untuk transfer data ke device functionnya. Sehingga disini modul peralatan yang hanya berfungsi sebagai Listener saja adalah Function generator dan Unit Switch yang hanya mendukung fungsi AH dan L saja.

Dari listing kode VHDL nya yang ada pada AH state adalah NRFD, NDAC, DAV, MLA, ATN, IFC, Data. Sedangkan untuk untuk L function yang ada adalah ATN, DAV, Data. Untuk pengujiannya dilakukan dengan simulasi pada program simulasi NOVA dan dengan mengambil langsung sinyalnya dengan frekwensi Generator dan oscilloscope. Adapun hasil simulasi yang didapatkan dari program simulasi NOVA adalah sebagai berikut :

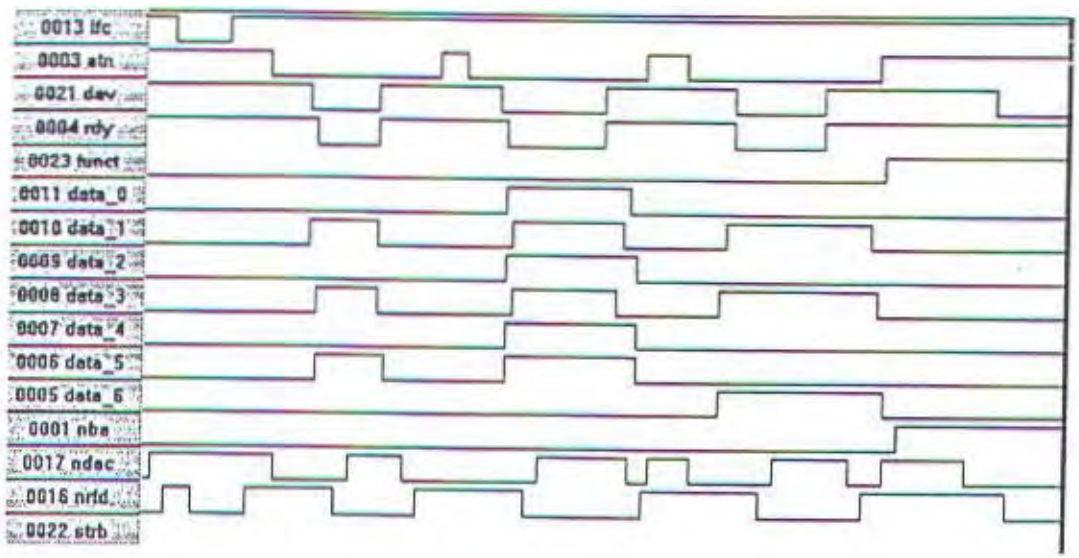


Gambar 5.1. timing AH dan L state

5.1.2. AH-SH function dan L-T function

Untuk peralatan yang bisa mengirim data memerlukan fungsi yang mengatur handshake dari peralatan ke listener yaitu SH function dan T function. Sedangkan untuk mengatur kondisi dan parameter yang ada pada peralatan seperti mode pengambilan dan lain lain maka lat tersebut perlu AH function dan L function. Dalam hal ini ADC yang dirancang difungsikan sebagai Listener pada saat inisialisasi dan pemilihan channel yang akan dibaca dan sebagai Talker pada saat mengirim data ke Listener yang dalam hal ini adalah komputer.

Kode VHDL yang terlampir akan menghasilkan timing diagram yang memperlihatkan saat berfungsi sebagai Listener dan Talker sebagai berikut :



Gambar 5.2 timing SH-AH dan T-L State

5.2. Pengujian Device Control Function.

Device control function merupakan blok untuk mengatur urutan data seperti yang diinginkan untuk mengatur fungsi dan meneruskan pesan dari kontroller. Pada perancangan alat terdapat 3 buah program untuk mengatur fungsi dan inisialisasi peralatan yang diatur yaitu :

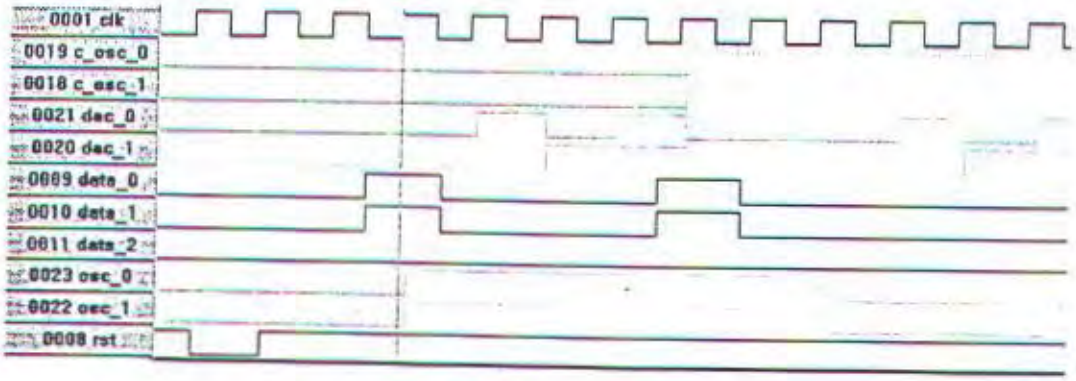
- Pengaturan switch yang aktif pada tiap group pada Unit Switch.
- Pengaturan fungsi output Function Generator.
- Pengaturan inisialisasi PIT untuk frekwensi sampling dan kontrol data keluar dari RAM ke bus GPIB pada Unit ADC / Multiprobe.

Adapun hasil dari pengaturan tersebut pada output dari masing masing data controller diatas adalah sebagai berikut

1. Data Switch Controller

Gambar 5.3 timing control data switch

2. Data Flow untuk Function Generator



Gambar 5.4 timing control AFG

3. Data Controller untuk inisialisasi PIT dan ADC data tranfer

Gambar 5.5 timing transfer data

5.3. Pengujian Sistem Device function

Pengujian device function mencakup pengujian terhadap kebenaran peralatan yang dikontrol untuk merespon control yang diberikan dari device controller. Baik itu berupa fungsi ataupun data seting.

5.3.1. Unit Switching.

Unit Switching merupakan unit yang hanya terdiri dari switch analog dan latch sehingga untuk pengujiannya dilakukan dengan 2 langkah yaitu :

Pengujian tanpa interface: dengan langsung memberikan data digital pada input digital dan melihat output yang dihasilkan.

Pengujian dengan interface: dengan menghubungkan langsung ke bus GPIB dengan menggunakan interface untuk unit switch yang telah dirancang.

Hasil yang didapatkan adalah sebagai berikut :

SEL = "000" => Data = XXXXXXXX => Switch :

-Semua Channel terisolasi

SEL = "001" => Data = IIXXXOOO => Switch :

-Group Input diatur oleh bit bit data :

- Input Channel = II
- Output Channel = OOO

-Group Output dan Group Reserve terisolasi / tetap.

SEL = "010" => Data = IIXXXOOO => Switch :

-Group Output diatur oleh bit bit data :

- Input Channel = II

BAB VI

PENUTUP

6.1. Kesimpulan

Dari hasil pembahasan dan hasil pengujian maka dapat diambil beberapa kesimpulan sebagai berikut :

- Standar bus IEEE 488 merupakan standar bus yang banyak dipergunakan untuk menginterfacekan instrumen instrumen yang programmable ke komputer sehingga dapat dikontrol melalui remote host.
- Interface bus GPIB ada 3 tipe yaitu controller, talker dan llistener.Dengan fungsi dan kemampuan yang berbeda beda.Dalam Tugas akhir ini tidak semua fungsi yang ada pada standar 488 dapat direspon.
- Untuk mengontrol suatu instrumen yang dihubungkan dengan bus IEEE488 diperlukan suatu prosedur /cara yang sama untuk sembarang instrumen, perbedaan yang ada hanya pada format data. Sehingga sembarang instrumen yang dihubungkan dengan bus IEEE488 dapat diatur dengan mudah seperti alat yang lainnya.
- Untuk tiap instrumen yang akan dihubungkan dengan bus IEE488 harus dapat merespon perintah perintah yang diberikan, yang paling sederhana sebagai listener akan merespon fungsi AH dan L saja dan untuk talker akan mereson

fungsi SH dan T saja. Sedangkan untuk fungsi yang lainnya tergantung kepada perancannya.

- Tiap modul yang dirancang telah memenuhi target yang diinginkan, namun terjadi kekurangan pada pengendaliannya yang menggunakan program Delphi 2 yang memanfaatkan fasilitas 32 bit.
- Dengan menggunakan perancangan VHDL maka proses perancangan menjadi lebih mudah dan simple, penggunaan komponen menjadi lebih efesien.
- VHDL merupakan bahasa pemrograman hardware tingkat tinggi, sehingga untuk mendiskripsikan suatu rangkaian menjadi sangat mudah dan sedikit memperhatikan jenis komponen yang perlu dipakai. Yang sangat perlu diperhatikan adalah kapasitas dan jumlah pin dari komponen PLD yang dipakai.

6.2.Saran saran

Dari hasil yang didapatkan dalam tugas akhir ini maka untuk perancangan suatu interface GPIB sebaiknya mengetahui timing diagram dari tiap perintah dan tidak hanya proses handshakingnya saja. Karena jika hanya mengandalkan proses handshaking maka proses yang terjadi secara detail tidak terlihat.

Jika dalam perancangan tidak langsung menggunakan IC talker/listener sebaiknya menggunakan Hardware Discription Language (HDL) karena akan mempermudah merumuskan pemecahan masalah yang berhubungan dengan proses squensial yang berurutan yang dipengaruhi oleh beberapa input seperti state machine.

Yang perlu diperhatikan dalam menggunakan HDL adalah jenis pemrograman , kapasitas IC PLD beserta jumlah pinnya.

Jika akan menggunakan jenis pemrograman dengan bahasa tingkat atas maka diperlukan suatu driver khusus yang bisa mengakses hardware GPIB II/IIA secara mudah. Namun Algoritma yang dipakai tidaklah berbeda.

Jika diinginkan suatu sistem yang stand alone maka sebaiknya mempergunakan sebuah mikrokontroller untuk menangani permasalahannya, karena dengan menggunakan mikrokontroller semua fungsi dan rutin rutin yang ada pada GPIB dapat ditampung dalam sebuah EPROM dan dijalankan. Penggunaan teknologi VHDL atau HDL lainnya akan hanya bisa menangani beberapa saja, kecuali kita memiliki suatu komponen PLD yang memiliki makrosell yang besar dan mempunyai kemampuan yang lebih tinggi dari CPLD.

DAFTAR PUSTAKA

1. Peatman, John B., Microcomputer Based Design, McGrow Hill Book Company, Singapore, 1977.
2. Hall,Douglas V., Microprocessor and Digital System, McGrow Hill Inc., Singapore 1983.
3. Stover, Allan C.,ATE: Automatic Tes Equipment, McGrow Hill Book Company, New York, 1984
4. Uffenback, John, The 8086/8088 Family: Design, Programming dan Interfacing, Pratice-Hall,Inc, Englewood Cliffs, New Jersey, 1977.
5. Pieper, Ir John M., Higher Performance Protocol For GPIB,©ACEA, Wierden The Netherland, 1994
6. Skahill,Kevin, VHDL for Programmable Logic, Addison-Wesley,Menlo Park, 1996.
7. ----,IEEE488 and VXI Bus Control Data Acuisition and Analisis, National Instrument Corporation, Austin, Texas, 1994.
8. ----,NI-488.2 Software Reference Manual for MS-DOS, National Instrunent Corporation, Austin, Texas, 1994.
9. ----,Getting Started with Your GPIB PCII/IIA and the NI-488.2 Software for MS-DOS, National Instrument Corporation, Austin, Texas,1994.

10. ----,Using Your NI-488.2 Software with Microsoft Windows, National Instrument Corporation, Austin, Texas, 1992.

11. ----,WARP VHDL Synthesis Reference, Cypress Semiconductor Co.,Q2 1997.

12. ----,MAXIM Product DataSheet, ver1.0, MAXIM, 1997

13. ----,National Data Acquisition DataBook, National Semiconductor,1995.

14. ---,National Interface DataBook,National Semiconductor,1995.

Lampiran 1: Listing VHDL untuk Unit Sinyal Generator

```
-- State Machine of AH, RL and L function
-- and massege decoder for GPIB interface
-- include with Listener Address => 01101b

ENTITY Listener is port (
        ATN,DAV,IFC,REN,rdy         : in bit;
        Data                        : in bit_vector
(6 downto 0);
        LDAC                        : out bit;
        GTL,strb            : inout bit;
        NRFD,NDAC                   : inout bit);
ATTRIBUTE part_name of Listener:ENTITY IS
"PALC22V10-20PC/PI";
ATTRIBUTE pin_numbers of Listener:ENTITY IS
"Data(0):11 Data(1):10 Data(2):9 Data(3):8 Data(4):7
Data(5):6 "
&"Data(6):5 DAV:1 REN:23 ATN:3 rdy:4 IFC:13
NRFD:16 NDAC:17 strb:22 LDAC:21 ";
END Listener;
-- 14
Architecture ArchListener of Listener is
Type StateMachine is ( ANRS, ACRS, ACDS, AWNS);
Signal State        : StateMachine := ACRS;
Signal MLA          : bit;
BEGIN    --20
State_AH: Process (ATN,IFC,DAV)
        begin
        if (IFC = '0') then
                State <= ANRS;
        else
        Case State is
        When ANRS =>
        if (MLA = '1' or ATN = '0' or rdy='1') then
        State <= ACRS;
        else
        State <= ANRS;
        end if;
        When ACRS =>
        if ((MLA = '1' or ATN = '0') and (DAV = '0'))
then --40
        State <= ACDS;
        else
        State <= ACRS;
        end if;
        When ACDS =>
        if (rdy = '0' or ATN ='1') then
        State <= AWNS;
        else
        State <= ACDS;
        end if; --50
        When AWNS =>
        if (DAV = '1')then
        State <= ANRS;
        else
        State <= AWNS;
        end if;
        End Case;
        end if;
        end process State_AH;

msg_decoder: Process (DAV,REN,IFC,ATN)--- L State if
modified
        begin
        if (DAV = '0') then
        if (ATN = '0' and Data = "0101011") then --
MLA
                MLA <= '1';
                elsif (ATN = '0' and Data = "0111111") then --
UNL
                MLA <= '0';
                end if;
                if (ATN = '0' and Data = "0000001" and MLA =
'1') then --GTL
                GTL <= '1';
                elsif (ATN = '0' and Data = "0010001" and
MLA = '1') then --LLO
                GTL <= '0';
                end if;
                elsif (IFC = '0') then
                MLA <= '0';
                GTL <= '0';
                elsif (REN = '0' and Data = "0001101") then
                GTL <= '1';
                MLA <= '0';
                end if;
end process msg_decoder;

NRFD <= '1' when (State = ACRS) else '0';
NDAC <= '1' when ((MLA = '0' and ((ATN = '0' and State
= AWNS)or(ATN = '1' and DAV = '1')))or  (MLA = '1' and
State = AWNS))else '0';
--NDAC <= '1' when ((MLA = '0' and (State = AWNS or
(State = ANRS and ATN = '1'))) or (MLA = '1' and State =
AWNS)) else '0';
strb <= '1' when (DAV = '0' and ATN = '1' and MLA = '1'
and NDAC = '1') else '0';
LDAC <= '0' when (DAV = '0' and ATN = '1' and MLA =
'1' and NDAC = '1') else '1';
END ArchListener;


-- Squential Data Control for Device function
-- in order to function generator

Entity Dataflow is port(
        Clk,rst,GTL         : in bit;
        LDAC                : in bit;
        Data                : in bit_vector (2 downto
0);
        OSC,DAC,C_osc       :  out  bit_vector  (1
downto 0));
Attribute pin_numbers of Dataflow:ENTITY is
"Data(0):9 Data(1):10 Data(2):11 rst:8 OSC(0):23
OSC(1):22 "
&"DAC(0):21 DAC(1):20 C_osc(0):19 C_osc(1):18
LDAC:2";
End Dataflow;
-- 9
use work.int_math.all;
Architecture DataControl of Dataflow is
type StateMachine is (Idle,Form,Freq,dF,DutyC,RangeS);
Signal State : StateMachine;
SIgnal Step  : integer range 0 to 1;
-- 15
Begin
Process (clk,rst)
        begin
        if rst = '0' then
                State <= Idle;
                Step <= 0;
        elsif (clk'event and clk = '1') then
                Case State is
```

```
When Idle =>
if Data = "001" then
State <= Form;
Step <= 0;
elsif Data = "010" then
Step <= 0;
State <= Freq;
elsif Data = "011" then
Step <= 0;
State <= dF;
elsif Data = "100" then
Step <= 0;
State <= DutyC;
elsif Data = "101" then
State <= RangeS;
Step <= 0;
else
Step <= 1;
State <= Form;
end if;
When Form =>
OSC(0) <= Data(0);
OSC(1) <= Data(1);
if Step = 0 then
State <= Idle;
else
State <= Freq;
end if;
When Freq =>
if Step = 0 then
State <= Idle;
else
State <= dF;
end if;
When dF =>
if Step = 0 then
State <= Idle;
else
State <= DutyC;
end if;
When DutyC =>
if Step = 0 then
State <= Idle;
else
State <= RangeS;
end if;
when RangeS =>
C_osc (0) <= Data (0);
C_osc (1) <= Data (1);
State <= Idle;
end case;
end if;
end process;
DAC <=  "00" when (State = freq and LDAC = '0') else
        "01" when (State = dF and LDAC = '0') else
        "11" when (State = DutyC and LDAC = '0') else
        "10";
End DataControl;
```

Lampiran 2: Listing VHDL untuk Unit Switching

```
-- State Machine of AH, RL and L function
-- and massege decoder for GPIB interface
-- include with Listener Address => 01101b

ENTITY Listener is port (
        ATN,DAV,IFC,REN,rdy     : in bit;
        Data                    : in bit_vector
(6 downto 0);
        GTL,strb        : inout bit;
        NRFD,NDAC               : inout bit);
ATTRIBUTE part_name of Listener:ENTITY IS
"PALC22V10-20PC/PI";
ATTRIBUTE pin_numbers of Listener:ENTITY IS
 "Data(0):11 Data(1):10 Data(2):9 Data(3):8 Data(4):7
Data(5):6 "
&"Data(6):5 DAV:1 REN:23 ATN:3 rdy:4 IFC:13
NRFD:16 NDAC:17 strb:22 GTL:21 ";
END Listener;
-- 14

Architecture ArchListener of Listener is
Type StateMachine is ( ANRS, ACRS, ACDS, AWNS);
Signal State    : StateMachine := ACRS;
Signal MLA      : bit;
BEGIN   --20
State_AH: Process (ATN,IFC,DAV)
        begin
                if (IFC = '0') then
                        State <= ANRS;
                else
                        Case State is
                                When ANRS
=>
                                if
(MLA = '1' or ATN = '0' or rdy='1') then

        State <= ACRS;

        else

        State <= ANRS;

        end if;
                                When ACRS
=>
                                if
((MLA = '1' or ATN = '0') and (DAV = '0')) then --40

        State <= ACDS;

        else

        State <= ACRS;

        end if;
                                When ACDS
=>
                                if
(rdy = '0' or ATN ='1') then

        State <= AWNS;


        else

        State <= ACDS;

        end if; --50
                                When AWNS
=>
                                if
(DAV = '1')then

        State <= ANRS;

        else

        State <= AWNS;

        end if;
                        End Case;
                end if;
        end process State_AH;

msg_decoder: Process (DAV,REN,IFC,ATN)          ---
L State if modified
        begin
                if (DAV = '0') then
                        if (ATN = '0' and Data =
"0101101") then --MLA
                                MLA <= '1';
                        elsif (ATN = '0' and Data
= "0111111") then --UNL
                                MLA <= '0';
                        end if;

                        if (ATN = '0' and Data =
"0000001" and MLA = '1') then --GTL
                                GTL <= '1';
                        elsif (ATN = '0' and Data
= "0010001" and MLA = '1') then --LLO
                                GTL <= '0';
                        end if;
                elsif (IFC = '0') then
                        MLA <= '0';
                        GTL <= '0';

                elsif (REN = '0' and Data =
"0001101") then
                        GTL <= '1';
                        MLA <= '0';
                end if;
end process msg_decoder;

NRFD <= '1' when (State = ACRS) else '0';
NDAC <= '1' when ((MLA = '0' and ((ATN = '0' and State
= AWNS)or(ATN = '1' and DAV = '0')or
                (ATN = '1' and DAV = '1')))or(MLA
= '1' and State = AWNS))else '0';
strb <= '1' when (DAV = '0' and ATN = '1' and MLA = '1'
and NDAC = '1') else '0';

END ArchListener;

-- Squential Data Control for Device Function
-- in order to Switcing Device

Entity DataChanelFlow is port(
        Clk,rst,GTL             : in bit;
        inbit                   : in bit;
        Data                    : in bit_vector (1 downto
0);
        S_Enable  : in bit;
```

```
        Selector1   : out x01z_vector (2 downto 0));
ATTRIBUTE pin_numbers of DataChanelFlow:ENTITY
IS
" Selector1(0):21 Selector1(1):22 Selector1(2):23 Data(0):9
Data(1):10 rst:8 S_Enable:7 ";

End DataChanelFlow;
-- 14
use work.int_math.all;
use work.rtlpkg.all;
Architecture DataControl of DataChanelFlow is
type StateMachine is (Idle,InGroup,OutGroup,RsvGroup);
Signal State          : StateMachine;
Signal Step           : integer range 0 to 1;
Signal Selector,sele  : bit_vector (2 downto 0);
Signal Ctrl_Out       : bit;
-- 19
Begin
Process (clk,rst)
        begin
        if rst = '0' then
                State <= Idle;
        elsif (clk'event and clk = '1') then
                Case State is
                        When Idle =>
                                if Data = "01"
then

        State <= InGroup;

        Step <= 0;
                                elsif  Data  =
"10" then

        Step <= 0;

        State <= OutGroup;
                                elsif  Data  =
"11" then

        Step <= 0;

        State <= RsvGroup;
                                elsif  Data  =
"00" then

        Step <= 1;
        State <= InGroup;
                                else

        State <= Idle;

                                end if;
                        When InGroup=>
                                if  Step  =  0
then

        State <= Idle;
                                else

        State <= OutGroup;
                                end if;
                        When OutGroup =>
                                if  Step  =  0
then

        State <= Idle;
                                else

        State <= RsvGroup;
                                end if;
                        When RsvGroup =>
                                State <= idle;
                end case;
        end if;
end process;
Selector <= "001" when (State = InGroup and inbit = '0')
else
            "010" when (State = OutGroup and inbit =
'0') else
            "100" when (State = RsvGroup and inbit =
'0') else
          "000";
Ctrl_Out <= '1' when (S_Enable = '0') else '0';

Pin_Out:triout                                  port
map(Selector(0),Ctrl_Out,Selector1(0));
Pin_Out1:triout                                 port
map(Selector(1),Ctrl_Out,Selector1(1));
Pin_Out2:triout                                 port
map(Selector(2),Ctrl_Out,Selector1(2));
End DataControl;
```

Lampiran 3: Listing VHDL untuk Unit Multiprobe

```
-- State Machine of AH, RL and L function
-- Combine with SH, T and SRQ function
-- and massege decoder for GPIB interface
-- include with Listener Address => 01010b
-- and Talker Address at => 01010b

use work.rtlpkg.all;
ENTITY Listener is port (
        ATN,IFC,rdy                     : in bit;
        Data                            : in bit_vector
(6 downto 0);
        rqs,nba                         : in bit;
        Strb,triger             : out bit;
        Funct,MLA,MTA                   : inout bit;
        NRFD,NDAC,DAV                   : inout x01z);
ATTRIBUTE pin_numbers of Listener:ENTITY IS
 "Data(0):11  Data(1):10  Data(2):9  Data(3):8  Data(4):7
Data(5):6 Data(6):5 "
&"MTA:18 MLA:15 nba:1 rqs:2 ATN:3 rdy:4 "
&"DAV:21 IFC:13 funct:14 NRFD:16 NDAC:17 strb:22 ";
END Listener;
-- 17

Architecture ArchListener of Listener is
Type StateMachine is ( ANRS, ACRS, ACDS, AWNS);
Signal State                            :
StateMachine := ACRS;
Signal notFunct,noStb                   : bit;
Signal NRFDo,NDACo,DAVo                 : bit;
BEGIN    --24
State_AH_SH:Process
(ATN,IFC,DAV,State,Data,MLA,MTA)
        begin
        if (IFC = '0' ) then
        State <= ANRS;
        else
        Case State is
        When ANRS =>
        if (Funct = '0') then
        if (ATN = '0' or rdy='1' or MLA = '1') then
        State <= ACRS;
        else
        State <= ANRS;
        end if;
        elsif (Funct = '1' and RQS = '1') then
        if (nba = '1' and MTA = '1' and MLA = '0') then
        State <= ACRS;
        else
        State <= ANRS;
        end if;
        else
        State <= ANRS;
        end if;
        When ACRS =>
        if (Funct = '0') then
        if ((MTA = '1' or MLA = '1' or ATN = '0') and
(DAV = '0')) then --40
        State <= ACDS;
        else
        State <= ACRS;
        end if;
        elsif (Funct = '1') then
        if (NRFD = '1' and NDAC = '0') then
        State <= ACDS;
        else
        State <= ACRS;
        end if;
        else
        State <= ACRS;
        end if;
        When ACDS =>
        if (Funct = '0') then
        if (rdy = '0' or ATN ='1') then
        State <= AWNS;
        else
        State <= ACDS;
        end if; --50
        elsif (Funct = '1') then
        if (NDAC = '1' and NRFD = '0') then
        State <= AWNS;
        else
        State <= ACDS;
        end if;
        else
        State <= ACDS;
        end if;
        When AWNS =>
        if (Funct = '0') then
        if (DAV = '1')then
        State <= ANRS;
        else
        State <= AWNS;
        end if;
        elsif (Funct = '1') then
        if (NRFD = '1') then
        State <= ANRS;
        else
        State <= AWNS;
        end if;
        else
        State <= AWNS;
        end if;
        End Case;
        end if;
        end process State_AH_SH;

State_L_T: Process (DAV,ATN,IFC,MTA) -- L and T
State if modified
        begin
                if (DAV = '0' and ATN = '0') then
                        Case data is
-- Listener Address
        when "0101010"      =>      MLA <= '1';
                                    MTA <= '0';
        when "0111111"      =>      MLA <= '0';
-- Talker Address
        when "1001010"      =>      MTA <= '1';
                                    MLA <= '0';
        when "1011111"      =>      MTA <= '0';
        when others         =>      MTA<=MTA;
                                    MLA<=MLA;
        end case;
-- Reset All State on IFC
        elsif (IFC = '0') then
        MLA <= '0';
        MTA <= '0';
        else
        if (ATN = '1' and MTA = '1') then
        Funct <= '1';
        else
        Funct <= '0';
        end if;
        end if;
end process State_L_T;
```

```
        -- Triger Device
triger <= '1' when (DAV = '0' AND ATN = '0' AND MTA
= '1' AND Data = "0001000") else '0';
                -- Out Equations
notFunct <= not(Funct);
DAVo  <= '0' when (State = ACDS) else '1';
NRFDo <= '1' when (State = ACRS) else '0';
NDACo <= '1' when ((MLA = '0' and ((ATN = '0' and State
= AWNS)or(ATN = '1' and DAV = '0')or(ATN = '1' and
DAV = '1'))or(funct = '0' and State = AWNS))else '0';
        -- Equation out to pin
NRFD_out:triout port map(NRFDo,notFunct,NRFD);
NDAC_out:triout port map(NDACo,notFunct,NDAC);
DAV_out:triout port map(DAVo,Funct,DAV);
Strb <= '0' when (DAV = '0' and ATN = '1' and (MLA = '1'
or MTA = '1')) else    '1';
END ArchListener;


-- State Machine of Data flow PIT Control

use work.rtlpkg.all;
ENTITY PIT_CTRL is port (
        STRB,Triger,MTA,MLA          : in bit;
        Full,DCL                     : in bit;
        Data,ATT                     : in bit_vector(1 downto
0);
        A                                    :          inout
bit_vector(1 downto 0);
        CS_PIT                               : inout bit;
        RstADC,RdADC,Count           : inout bit);
Attribute Pin_numbers of PIT_CTRL:ENTITY is
" STRB:1 full:3 Triger:2 DCL:4 MTA:5 MLA:6 Data(0):9
Data(1):10 "
&" RdADC:19 RstADC:18 A(0):21 A(1):22 CS_PIT:23
Count:20";
END PIT_CTRL;
-- 12

Architecture ArchListener of PIT_CTRL is
Type StateMachine is (CW, SelCnt, Data0, Data1);
Type StateSend       is  (  Idle,  GetData,  ReqInt,
SendData);
Signal State                 : StateMachine := CW;
Signal SendState     : StateSend := Idle;
BEGIN    --19
Init_PIT: Process (MLA,STRB)
        begin
        if (MLA = '0') then
        State <= CW;
        elsif (STRB'Event and STRB = '1') then
        if MLA = '1' then
        Case State is
        When CW =>
        State <= SelCnt;
        When SelCnt =>
        Case Data is
        when "00" =>
        State <= Data0;
        when "01" =>
        State <= Data0;
        when others =>
        State <= SelCnt;
        end case;
        When Data0 =>
        State <= Data1;
        When Data1 =>
        State <= CW;
        End case;
        end if;
        end if;
end Process Init_PIT;
CS_PIT <= '0' When (State /= SelCnt and STRB = '0' and
MLA = '1') else '1';
A(0) <= Data(0) when (State = SelCnt and strb = '0' and
MLA = '1') else'1' When (State = CW and MLA = '1') else
'1' When (MTA = '1' and MLA = '0' and (Full = '1' or
DCL= '1')) else -- for RQS to SH-State
        '0' When (MTA = '1' and Triger = '1' ) else A(0);
A(1) <= Data(1) when (State = SelCnt and Strb = '0' and
MLA = '1') else'1' when (State = CW and MLA = '1') else
RdADC When (MTA = '1' and Count = '0') else  A(1); --
for nba to SH-State

Aq_Data: rocess (Triger,MTA,SendState,DCL,Full,STRB)
        Begin
        if MTA = '0' then
        SendState <= idle;
        Count <= '0';
        RdADC <= '1';
        else
        Case SendState is
        When Idle =>
        if Triger = '1' then
        Count <= '1';
        SendState <= GetData;
        elsif DCL = '1' then
        SendState <= SendData;
        else
        RdADC <= '1';
        Count <= '0';
        SendState <= Idle;
        end if;
        When GetData =>
        if Full = '1' then
        Count <= '0';
        SendState <= ReqInt;
        else
        RdADC <= '1';
        Count <= '1';
        SendState <= GetData;
        end if;
        When ReqInt =>
        if DCL = '1' then
        SendState <= SendData;
        else
        SendState <= ReqInt;
        end if;
        When SendData =>
        if Triger = '1' then
        SendState <= GetData;
        else
        Count <= '0';
        RdADC <= STRB;
        SendState <= SendData;
        end if;
        End Case;
        end if;
End Process Aq_Data;
RstADC <= '1' when SendState = Idle else
        '1' When DCL = '1' else '0';
END ArchListener;

-- Fungsi Tambahan Untuk data control ADC

ENTITY Sum_of_ADC is port(
        A                    : in bit_vector(1 downto
0);
```

```
Data                    : in bit_vector(5 downto 0);
ATN,triger: in bit;
funct                   : in bit;
dir                     : out bit;
ATT1,ATT2               : inout bit_vector(1 downto 0);
Mode,Dcl,ch             : inout bit);
Attribute pin_numbers of Sum_of_ADC:ENTITY is
" Data(0):1 Data(1):2 Data(2):3 Data(3):4 Data(4):5
Data(5):6 A(1):7 A(0):8 "
&" funct:9 ATN:11 dir:19 ATT1(0):13 ATT1(1):14
ATT2(0) Dcl:12 Ch:17 ATT2(1):18 ";
end Sum_of_ADC;
--11
Architecture ArchPad of Sum_of_ADC is
```

```
Begin

ATT1 <= Data (1 downto 0) when (ATN = '1' and A =
"10") else ATT1;
ATT2 <= Data (3 downto 2) when (ATN = '1' and A =
"10") else ATT2;
Dir <= not(funct);
Dcl <= '1' when (Data = "010100" and ATN = '0') else
       '0';
Ch <= '1' when (Data = "010110" and ATN = '0') else
   '0' when (Data = "011001" and ATN = '0') else Ch;

end;
```

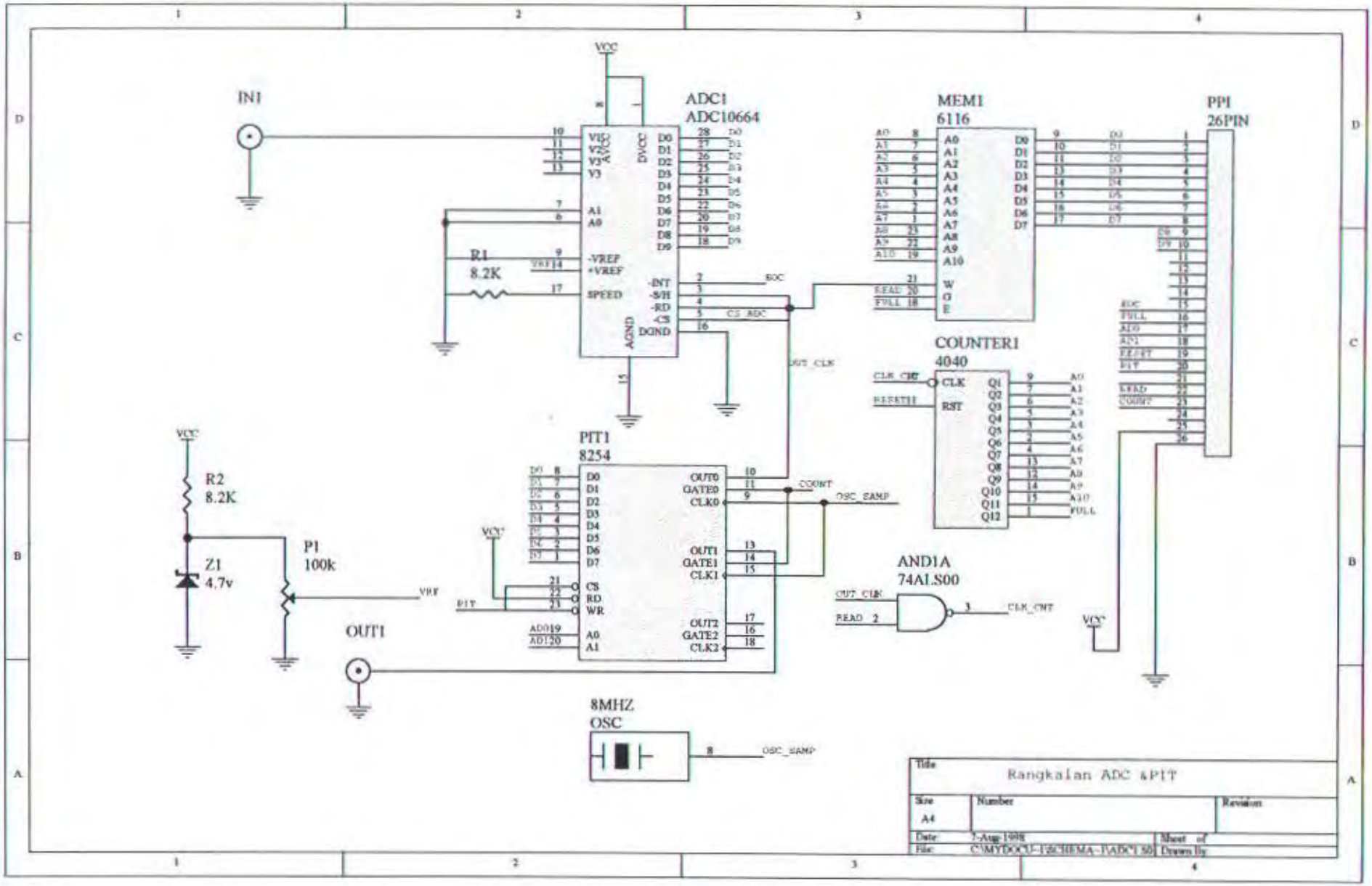
Lampiran 4: Listing Program Control Untuk Sistem

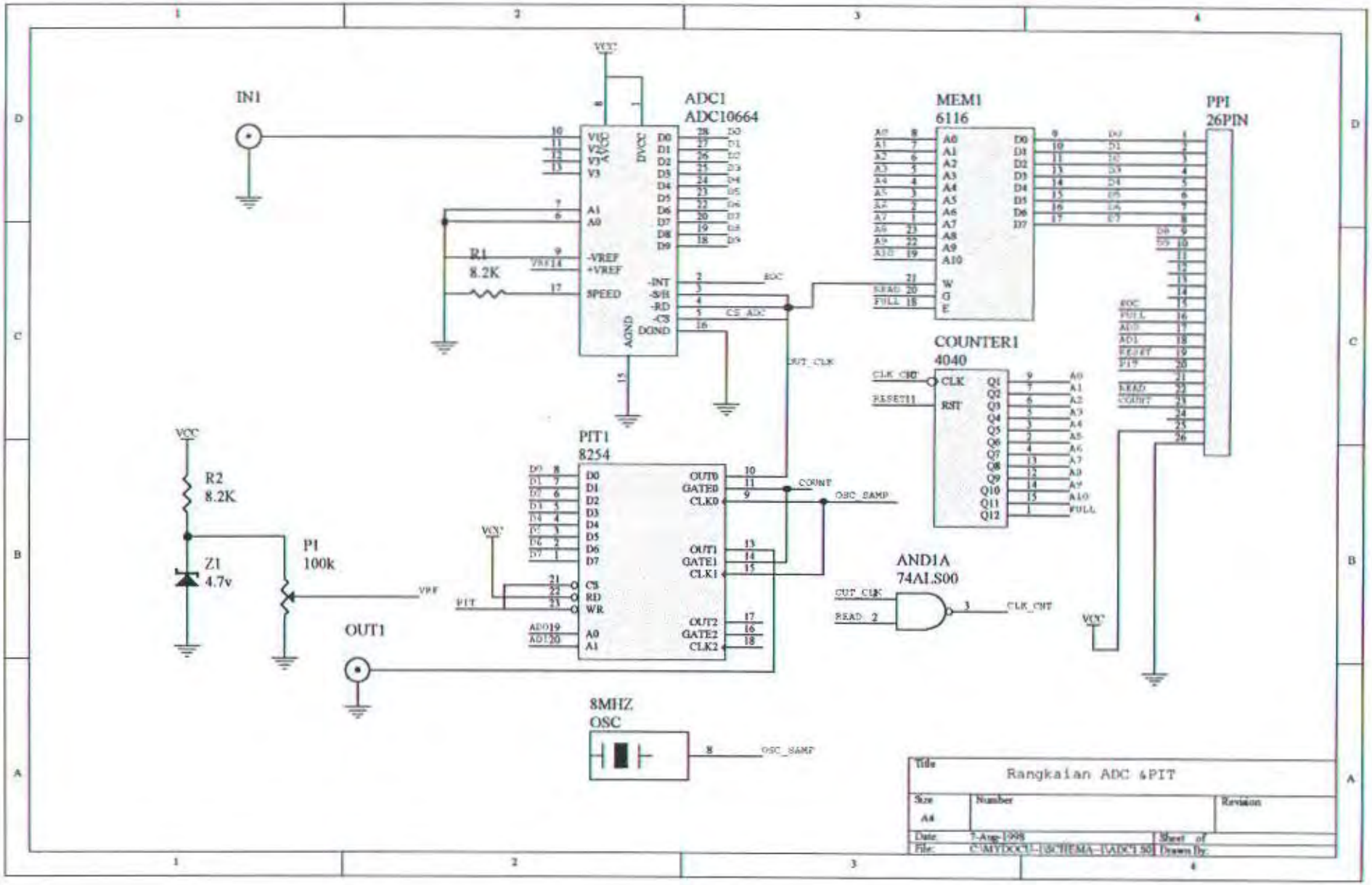
```
unit control;
interface
uses
  winprocs,Dialogs,GpibDec,SysUtils;
const
  trigger = $08;
  ClearADC = $04;
  Channel1 = $16;
  Channel2 = $17;
  ADCTalk  = $4a;
  ADCUntalk  = $5f;
  Unlisten  = $3f;

type
  BuffData = array[1..2048] of char;
  DataKeGPIB = array[1..10] of byte;
  TDataSwitch = record
              input,output,reserv:byte;
            end;
  TDataADC = record
              FrekSampling : integer;
              FrekOutSyn   : integer;
              Attenuator   : byte;
           end;
  TDataAFG = record
              form,frek,dF,dutyC,range: byte;
           end;

var
(*  NI-488.2  GPIB  global  status  variables
*)
  instruments   : AddrList4882_t;
  results       : AddrList4882_t;
  BuffCmd       : array[0..10] of Char;
  GPIBOut       : array[0..10] of char;

(* Personal used variable*)
  DataCh1,DataCh2 : array[1..1024] of char;
  DataAnalisa     : array[1..8] of real;
  DataGrapic1     : array[1..512] of real;
  DataGrapic2     : array[1..512] of real;
  DataTable       : array[1..512] of real;
  NodeDiskripsi   : array [1..8] of String;
  Switch          : TDataSwitch;
  ADC             : TDataADC;
  AFG             : TdataAFG;
procedure Control_Switch(mode:byte);
procedure Control_ADC(Fs :integer;AT1,AT2 :byte);
procedure Control_AFG(mode:byte);
procedure Sampling_Data(channel:byte;var stat:word);
procedure reset;
procedure DC_Respon_Analisis;
Procedure AC_Respon_Analisis(AC_range : byte);
Procedure Transient_Respon_Analisis;
Procedure Frekwensi_Respon_Analisis;

implementation
uses Gpib,U_gpib1,U_gpib2,intro1;

{Personal declaration}
procedure Control_Switch(mode:byte);
var
  JumCmd  : byte;
  Data_ctrl : PChar;

begin
   Case mode of
      0 : begin
            JumCmd := 2;
            GPIBOut[0]  :=  Chr($1);  {mode all switch
controlled}
            GPIBOut[1] := Chr(Switch.input);
            Send(0,13,GPIBOut,JumCmd,0);
            GPIBOut[0] := Chr($2);
            GPIBOut[1] := Chr(Switch.output);
            Send(0,13,GPIBOut,JumCmd,0);
            GPIBOut[0] := Chr($3);
            GPIBOut[1] := Chr(Switch.reserv);
            Send(0,13,GPIBOut,JumCmd,0);
          end;
      1 : begin
            GPIBOut[0]  :=  Chr($1);  {input  switch
controlled}
            GPIBOut[1] := Chr(Switch.input);
            JumCmd := 2;
            Send(0,13,GPIBOut,JumCmd,0);
          end;
      2 : begin
            GPIBOut[0]  :=  Chr($2);  {output  switch
controlled}
            GPIBOut[1] := Chr(Switch.output);
            JumCmd := 2;
            Send(0,13,GPIBOut,JumCmd,0);
          end;
      3 : begin
            GPIBOut[0]  :=  Chr($3);  {reserv  switch
controlled}
            GPIBOut[1] := Chr(Switch.input);
            JumCmd := 2;
            Send(0,13,GPIBOut,JumCmd,0);
          end;
   end;
end;

procedure Control_ADC(Fs :integer;AT1,AT2 : byte);
var
  Oplogic : byte;
  PITMSB,PITLSB : byte;
  Devider     : word;
  ATTe1,ATTe2  : byte;
begin
  with ADC do
  Begin
     Case AT1 of
        1 : ATTe1 := $02;
        2 : ATTe1 := $00;
        3 : ATTe1 := $01;
        4 : ATTe1 := $03;
     end;
     Case AT2 of
        1 : ATTe2 := $02;
        2 : ATTe2 := $00;
        3 : ATTe2 := $01;
        4 : ATTe2 := $03;
     end;
     FrekSampling := Fs;
     Devider := Round(5000000/FrekSampling);
     Oplogic := $00;
     PITMSB := Hi(Devider);
     PITLSB := Lo(Devider);
     ATTe1 := ATTe1 shl 4;
     ATTe2 := ATTe2 shl 2;
     Oplogic := ATTe1 or ATTe2;
     Attenuator := Oplogic;
```

```
    end;
    GPIBOut[0] := Chr($75);
    GPIBOut[1] := Chr(Oplogic or $2);
    GPIBOut[2] := Chr(Oplogic or $3);
    GPIBOut[3] := Chr($01);
    GPIBOut[4] := Chr(PITLSB);
    GPIBOut[5] := Chr(PITMSB);
    Send(0,10,GPIBOut,6,0);
end;

procedure Control_AFG(mode:byte);
var
  JumCmd : byte;
begin
   Case mode of
      0 : begin
            JumCmd := 2;
            GPIBOut[0] := Chr($01); {form controlled}
            GPIBOut[1] := Chr(AFG.form);
            Send(0,11,GPIBOut,JumCmd,0);

            GPIBOut[0] := Chr($02); {frek controlled}
            GPIBOut[1] := Chr(AFG.frek);
            Send(0,11,GPIBOut,JumCmd,0);

            GPIBOut[0] := Chr($03); {dF controlled}
            GPIBOut[1] := Chr(AFG.dF);
            Send(0,11,GPIBOut,JumCmd,0);

            GPIBOut[0] := Chr($04); {dutyC controlled}
            GPIBOut[1] := Chr(AFG.dutyC);
            Send(0,11,GPIBOut,JumCmd,0);

            GPIBOut[0] := Chr($05); {range controlled}
            GPIBOut[1] := Chr(AFG.range);
            Send(0,11,GPIBOut,JumCmd,0);
          end;
      1 : begin
            GPIBOut[0] := Chr($1); {itch controlled}
            GPIBOut[1] := Chr(AFG.form);
            JumCmd := 2;
            Send(0,11,GPIBOut,JumCmd,0);
          end;
      2 : begin
            GPIBOut[0] := Chr($2); {output switch
controlled}
            GPIBOut[1] := Chr(AFG.frek);
            JumCmd := 2;
            Send(0,11,GPIBOut,JumCmd,0);
          end;
      3 : begin
            GPIBOut[0] := Chr($3); {reserv switch
controlled}
            GPIBOut[1] := Chr(AFG.dF);
            JumCmd := 2;
            Send(0,11,GPIBOut,JumCmd,0);
          end;
      4 : begin
            GPIBOut[0] := Chr($4); {reserv switch
controlled}
            GPIBOut[1] := Chr(AFG.dutyC);
            JumCmd := 2;
            Send(0,11,GPIBOut,JumCmd,0);
          end;
      5 : begin
            GPIBOut[0] := Chr($5); {reserv switch
controlled}
            GPIBOut[1] := Chr(AFG.range);
            JumCmd := 2;
            Send(0,11,GPIBOut,JumCmd,0);
          end;
    end;
end;

procedure Sampling_Data(channel:byte;var stat:word);
var
  HasilSRQ : smallint;
begin
   stat := 0;
   {Send talk address and trigger command to start
sampling}
   BuffCmd[0] := Chr(Unlisten);
   BuffCmd[1] := Chr(ClearADC);
   BuffCmd[2] := Chr(ADCTalk);
   BuffCmd[3] := Chr(Trigger);
   ibCmd(0,BuffCmd,4);
   {Wait for SRQ from ADC}
   WaitSRQ(0,HasilSRQ);
   if (HasilSRQ) = 0 then
   begin
      stat := 1 ;
      messageDlg('Sampling                    ADC
Error',mtError,[mbOK],0);
      exit;
   end;
   {Send DevClear and init read to channel 1}
   BuffCmd[0] := Chr(ClearADC);
   BuffCmd[1] := Chr(Channel1);
   ibCmd(0,BuffCmd,2);
   {Start Receive ADC to buffer}
   Case channel of
      0 : begin
            {Receive Channel1}
            Receive(0,10,DataCH1,510,STOPend);
            if (ibsta AND ERR) <> 0 then
            begin
               stat := 1;
               messageDlg('Receive      Data      ADC
Error',mtError,[mbOK],0);
               exit;
            end;
            {Reset ADC and init read to channel 2}
            BuffCmd[0] := Chr(ClearADC);
            BuffCmd[1] := Chr(Channel2);
            ibCmd(0,BuffCmd,2);
            {Receive Channel1}
            Receive(0,10,DataCH2,510,STOPend);
            if (ibsta AND ERR) <> 0 then
            begin
               stat := 1;
               messageDlg('Receive      Data      ADC
Error',mtError,[mbOK],0);
               exit;
            end;
          end;
      1 : begin
            {Receive Channel1}
            Receive(0,10,DataCH1,510,STOPend);
            if (ibsta AND ERR) <> 0 then
            begin
               stat := 1;
               messageDlg('Receive      Data      ADC
Error',mtError,[mbOK],0);
               exit;
            end;
          end;
      2 : begin
            {Reset ADC and init read to channel 2}
```

```
            BuffCmd[0] := Chr(ClearADC);
            BuffCmd[1] := Chr(Channel2);
            ibCmd(0,BuffCmd,2);
            {Receive Channel1}
            Receive(0,10,DataCH2,510,STOPend);
            if (ibsta AND ERR) <> 0 then
            begin
              stat := 1;
              messageDlg('Receive Data ADC
Error',mtError,[mbOK],0);
              exit;
            end;
          end;
    end;
    BuffCmd[0] := Chr(ADCUntalk);
    BuffCmd[1] := Chr(Unlisten);
    ibCmd(0,BuffCmd,2);
end;

procedure reset;
var
  stat,err,ictrl : integer;
begin
    {SendIFC(0,stat,err,ictrl); }
end;

procedure DC_Respon_Analisis;
var
  inp4,inp8,out4,out8,resv :byte;
  ATT1,ATT2          : byte;
  i,j    : integer;
  stat   :word;
  temp   : longint;
begin
    {==========process==========}
    {setting switch positions, primary input system is
ground}
    inp4 := $3 shl 6;
    inp8 := $0;
    Switch.input := inp4 or out8;
    Switch.Output := $00;
    Control_Switch(1);
    Control_Switch(2);
    {for number of node do get data form ADC with squen.
: send any command}
    for i := 1 to Jumlah_Node do
    begin
      Sampling_Data(0,stat);
      temp := 0;
      for j := 1 to 510 do temp := ord(DataCH1[j]) + temp;
      DataAnalisa[i] := temp / 510;
      Switch.output := Switch.output + $1;
      Control_switch(0);
    end;
    {==========visual==========}
    OutputA.show;
    OutputA.BtnGraph.enabled := false;
    OutputA.timer1.enabled := false;
    MainForm.Enabled := false;
    with OutputA do
    for i := 1 to jumlah_node do
    begin
      OutStrGrid1.cells[0,i] := ' ' + IntToStr(i);
      OutStrGrid1.cells[1,i]                    :=
FloatToStrF(DataAnalisa[i],ffFixed,5,10);
      OutStrGrid1.cells[2,i] := NodeDiskripsi[i];
    end;
end;

Procedure AC_Respon_Analisis(AC_range : byte);
var
  inp4,Out8,ATT1,ATT2 : byte;
  inp_frek,inp_range : byte;
  i,j          : integer;
  stat         : word;
  temp         : longint;
  DataHi1,DataLo1   : byte;
  DataHi2,DataLo2   : byte;

begin
    {==========process==========}
    {setting switch positions, primary input system is AFG}
    inp4 := 0;
    Out8 := 0;
    Switch.input := (inp4 shl 6) or (Switch.input and $0f);
    Switch.output := out8 or (Switch.output and $c0);
    Control_switch(1);
    Control_switch(2);
    {choose range frek. field for test : 50 Hz 1Khz 100Khz
or user define}
    Case AC_range of
      0 : begin
            AFG.frek  := $10;
            AFG.range := 01;
          end;
      1 : begin
            AFG.frek  := $f0;
            AFG.range := 01;
          end;
      2 : begin
            AFG.frek  := $40;
            AFG.range := 02;
          end;
      3 : begin
            AFG.frek  := inp_frek;
            AFG.range := inp_range;
          end;
    end;
    {setting frek. output AFG for test}
    AFG.form := $02;
    Control_AFG(1);
    Control_AFG(2);
    Control_AFG(5);
    {get numbers of node being sampling }
    {setting ADC function to AC mode and any value of
ATT and fs
    {for number of node do get data form ADC with squen.
: send any command}
    for i := 1 to Jumlah_Node do
    begin
      Sampling_Data(0,stat);
      temp := 0;
      DataHi1 := 0;
      DataLo1 := 255;
      DataHi2 := 0;
      DataLo2 := 255;
      for j := 1 to 510 do
      begin
        DataGrapic1[j] := ord(DataCh1[j]);
        DataGrapic2[j] := ord(DataCh2[j]);
        if ord(DataCh1[j])> DataHi1 then DataHi1 :=
ord(DataCh1[j]);
        if ord(DataCh1[j])< DataLo1 then DataLo1 :=
ord(DataCh1[j]);
        if ord(DataCh2[j])> DataHi2 then DataHi2 :=
ord(DataCh2[j]);
        if ord(DataCh2[j])< DataLo2 then DataLo2 :=
ord(DataCh2[j]);
```

```
      end;
      DataAnalisa[i] := DataHi1 - DataLo1;
      Switch.output := Switch.output + 1;
      Control_switch(2);
    end;
    {----------visual----------}
    OutputA.enabled := true;
    OutputA.show;
    OutputA.BtnGraph.enabled := true;
    OutputA.timer1.enabled := false;
    MainForm.Enabled := false;
    with OutputA do
    begin
      OutStrGrid1.RowCount := jumlah_node +1;
      for i := 1 to jumlah_node do
      begin
        OutStrGrid1.cells[0,i] := ' ' + IntToStr(i);
        OutStrGrid1.cells[1,i]                    :=
FloatToStrF(DataAnalisa[i],ffFixed,5,10);
        OutStrGrid1.cells[2,i] := NodeDiskripsi[i];
      end;
    end;
end;

Procedure Transient_Respon_Analisis;
var
  i,j    : integer;
  stat   : word;
  out8   : byte;
begin
    Out8 := Node;
    {----------process----------}
    Jumlah_node := 1;

    {setting ADC function to DC mode and any value of
ATT and fs}
    for i := 1 to Jumlah_Node do
    begin
      Sampling_Data(0,stat);
      for j := 1 to 500 do DataGrapic1[j] := round
(ord(DataCH1[j])/255 *1024);
      for j := 1 to 500 do DataGrapic2[j] := round
(ord(DataCH2[j])/255 *1024);
    end;

{----------visual----------
    }
    OutputA.show;
    OutputA.BtnGraph.enabled := true;
    MainForm.Enabled := false;

end;

Procedure Frekwensi_Respon_Analisis;
var
  OpLogic : byte;
  i,j,step : integer;
  rangeTest : integer;
  k,temp   : integer;
  DataLo,Datahi    : byte;
  stat    : word;
begin
    Intro.Show;
    intro.panel1.show;
    intro.panel2.show;
    Intro.Gauge1.progress := 0;
    Intro.ListBox1.clear;
    intro.enabled := true;
    Intro.label1.caption := 'Prosess Sampling Data ';
    Intro.listbox1.items.Add('Process :');
    intro.listbox1.items.Add('Analisa Frek..');
    {----------proccess----------}
    messageDlg('Analisa Frekwensi AKan di
Jalankan',mtInformation,[mbOK],0);
    {setting switch positions, primary input system is AFG}
    Switch.input := Switch.input and $3f;
    Switch.output := Switch.output and $00;
    Switch.reserv := $c0;
    Control_switch(0);
    {get numbers of node that being sampling
    ADC.FrekSampling := 1000000;
    Control_ADC(ADC.Freksampling,1,1);
    {setting ADC function to DC mode and any value of
ATT and fs}
    {for number of node do get data form ADC with squen.
: send any command}
    RangeTest := 1;
    Jumlah_Node := 1;
    for i := 1 to Jumlah_Node do
    begin
      temp := 0;
      intro.gauge1.progress := 0;
      j := 1;
      While ( j < 500) do
      begin
        Sampling_Data(1,stat);
{         SendIFC(0);}
        if (j < 500) then
        begin
          AFG.frek := Round(j/2);
          if AFG.frek = $3f then AFG.frek := $40;
          AFG.form := $02;
          Control_AFG(0);
        end;
        intro.gauge1.progress := round(((j+1)*100)/500);
        if Stat = 1 then exit;
        DataLo := 255;
        DataHi :=0;
        for k:= 1 to 500 do
        begin
          if ord(DataCh1[k])> DataHi then DataHi :=
ord(DataCh1[k]);
          if ord(DataCh1[k])< DataLo then DataLo :=
ord(DataCh1[k]);
        end;
        for k := j to k+5 do DataGrapic1[k] := DataHi -
DataLo;
        inc(j,5);
      end;
    end;
    intro.hide;
    OutputA.show;
    OutputA.OutStrGrid1.Cells[0,0] := ' Step';
    OutputA.OutStrGrid1.RowCount := 501;
    for i:= 1 to 500 do
    begin
      OutputA.OutStrGrid1.Cells[0,i] := ' ' + IntToStr(i);
      OutputA.OutStrGrid1.Cells[1,i]    :=    '    '    +
FloatToStr(DataGrapic1[i]);
    end;
    OutputA.BtnGraph.enabled := true;
    MainForm.Enabled := false;
    Y_Scale := 1;
    X_Scale := 1;

end;

end.
```


VCC
IN1
ADC1
ADC10664
MEM1
6116
PPI
26PIN
R1
8.2K
-VREF
+VREF
SPEED
SOC
-INT
-S/H
-RD
-CS
DGND
AGND
CS_ADC
OUT_CLK
COUNTER1
4040
CLK
RST
PIT1
8254
D0
D1
D2
D3
D4
D5
D6
D7
OUT0
GATE0
CLK0
OUT1
GATE1
CLK1
OUT2
GATE2
CLK2
CS
RD
WR
A0
A1
COUNT
OSC_SAMP
R2
8.2K
Z1
4.7v
P1
100k
VREF
OUT1
AND1A
74ALS00
CLK_CNT
8MHZ
OSC
Title
Rangkaian ADC &PIT
Size
A4
Number
Revision
Date
7-Aug-1998
Sheet of
File
Drawn By


IN1
ADC1
ADC10664
MEM1
6116
PPI
26PIN
R1
8.2K
COUNTER1
4040
PIT1
8254
R2
8.2K
Z1
4.7v
P1
100k
OUT1
AND1A
74ALS00
8MHZ
OSC
Rangkaian ADC 4PIT
Size
A4
Number
Revision
Date:
7-Aug-1998
Sheet of
File:
C:\MYDOCS\...\SCHEMA\ADC4.SCH
Drawn By:

DAFTAR RIWAYAT HIDUP

I Nengah Widianarta lahir di Klungkung, Bali pada tanggal 12 Februari 1973, dari pasangan I Nengah Sudana dan Ni Nyoman Sawit yang saat ini bertempat tinggal di Jl. Cempaka III / 6 Klungkung dan merupakan anak kedua dari empat bersaudara. Telah menempuh pendidikan dasar di SD Negeri 1 Pekandelan dan lulus tahun 1985. Pendidikan menengah pertama di SMPN 1 Klungkung dan lulus tahun 1988. Kemudian melanjutkan pendidikan di STM Negeri Denpasar dan lulus tahun 1991. Melanjutkan pendidikan S1 di ITS jurusan Teknik Elektro tahun 1992 dan diharapkan lulus pada ujian tugas akhir periode wisuda September 1998. Selama berada di Teknik Elektro ITS, penulis menjadi koordinator Laboratorium Mikroelektronika, pengganti Koordinator asisten Elektronika, asisten praktikum Rangkaian Listrik, Elektronika dan Elektronika Lanjutan II di laboratorium Bidang Studi Elektronika.