การควบคุมอุปกรณ์ GPIB ผ่านเครือข่ายอินเตอร์เน็ต

A GPIB INSTRUMENTS CONTROL VIA INTERNET

โดย

นาย คมกฤษณ์ ชูเรือง 39013191

นาย แวฮาซัน แวหะมะ 39013210

นาย อนุชา หาญสมบัติเจริญ 39013222

อาจารย์ที่ปรึกษา

รศ.ดร. มนัส สังวรศิลป์

รายงานฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมอิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2541

ปริญญานิพนธ์ ปีการศึกษา 2541

ภาควิชา อิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การควบคุมอุปกรณ์ GPIB ผ่านเครือข่ายอินเตอร์เน็ต

ผู้จัดทำ

นาย คมกฤษณ์ ชูเรือง รหัส 39013191

นาย แวฮาซัน แวหะมะ รหัส 39013210

นาย อนุชา หาญสมบัติเจริญ รหัส 39013222

..... อาจารย์ที่ปรึกษา

(รศ.ดร. มนัส สังวรศิลป์)

การควบคุมอุปกรณ์ GPIB ผ่านเครือข่ายอินเตอร์เน็ต

A GPIB INSTRUMENTS CONTROL VIA INTERNET

นาย คมกฤษณ์ ชูเรือง 39013191

นาย แวฮาซัน แวหะมะ 39013210

นาย อนุชา หาญสมบัติเจริญ 39013222

โครงงานได้รับการตรวจสอบแล้ว พร้อมที่จะทำการสอบได้

.....

(รศ.ดร. มนัส สังวรศิลป์)

อาจารย์ที่ปรึกษา

กิตติกรรมประกาศ

ทางผู้จัดทำปริญญานิพนธ์ขอกราบขอบพระคุณ รศ.ดร. มนัส สังวรศิลป์ เป็นอย่างยิ่งซึ่งได้ให้คำแนะนำในการจัดทำโครงงานฉบับนี้ และขอขอบคุณพี่ กิตติ และพี่ ๆ ปริญญาโท ที่คอยช่วยเหลือในด้านต่าง ๆ ขอบคุณศศิวิมล ฮงมา ที่ช่วยจัดทำด้านเอกสาร

.....คมกฤษณ์ ชูเรือง.....
(นาย คมกฤษณ์ ชูเรือง)

.....
(นาย แวฮาซัน แวหะมะ)

.....อนุชา หาญสมบัติเจริญ.....
(นาย อนุชา หาญสมบัติเจริญ)
ผู้จัดทำ

การควบคุมอุปกรณ์ GPIB ผ่านเครือข่ายอินเตอร์เน็ต

นายคมกฤษณ์ ชูเรือง
นายแวฮาซัน แวหะมะ
นายอนุชา หาญสมบัติเจริญ
รศ. ดร. มนัส สังวรศิลป์
ปีการศึกษา 2541

บทคัดย่อ

เนื่องจากความก้าวหน้าทางเทคโนโลยีเครือข่ายคอมพิวเตอร์ในปัจจุบัน ส่งผลให้เครือข่ายอินเตอร์เน็ตแพร่หลาย และได้รับความนิยมเป็นอย่างมาก ซึ่งสามารถนำไปประยุกต์ใช้งานได้มากมาย โครงงานนี้จึงได้ประยุกต์การใช้เครือข่ายอินเตอร์เน็ต เพื่อใช้ประโยชน์ทางวิศวกรรมโดยนำมาประยุกต์ใช้ในการควบคุมเครื่องมือวัด GPIB (IEEE-488) ผ่านเครือข่ายอินเตอร์เน็ต ซึ่งสามารถทำให้สามารถควบคุมชุดเครื่องมือนี้ในระยะไกลได้ จากเครื่องคอมพิวเตอร์ที่ต่ออยู่กับระบบอินเตอร์เน็ต โดยใช้โปรโตคอล TCP/IP และระบบฐานข้อมูลเข้าช่วย โดยที่โปรแกรมของผู้ใช้จะส่งข้อมูลไปยังฐานข้อมูล และคอมพิวเตอร์ที่ต่ออยู่กับชุดเครื่องมือวัด จะนำข้อมูลที่ได้จากฐานข้อมูลมาประมวลผล ควบคุม และรับผลที่ได้ จากชุดเครื่องมือวัดส่งให้ผู้ใช้ต่อไป

A GPIB INSTRUMENT CONTROL VIA INTERNET

Mr. Komkrit Chooruang

Mr. Weahason Weahama

Mr. Anucha Harnsombutjaroen

Assoc.Prof.Dr.Manus Sungwarasin

Educational Year 1998

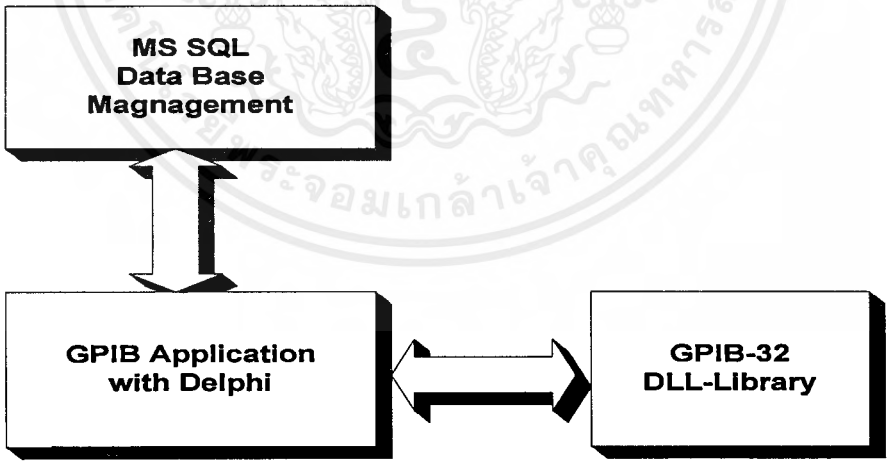
Abstract

Because of, a progressive of computer network technology, the internet system becomes a very popular and widely used in many applications. This project applies the advantages of internet system to control the IEEE-488 (GPIB) instruments. By using TCP/IP protocol and database system, we can control a GPIB instrument from the computer connected to the internet. Users have to send command of an instrument to SQL database server. The computer which interface with the GPIB instrument receive this command from database and process to control GPIB instrument and receive the reply (data) of instrument return to database and user will receive this data from database respectively.

สารบัญ

บทที่ 1
บทนำ

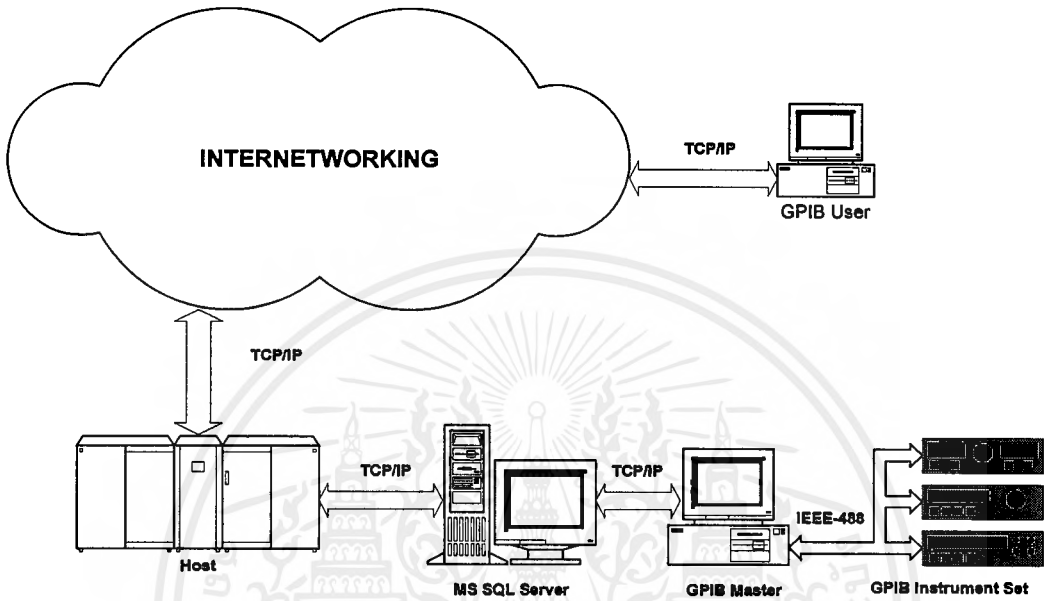
ความก้าวหน้าทางเทคโนโลยีเครือข่ายคอมพิวเตอร์ ในปัจจุบันส่งผลให้เครือข่ายอินเตอร์เน็ต ได้รับความนิยมจากผู้ใช้อย่างแพร่หลายทั่วโลก ซึ่งมีรูปแบบของการประยุกต์ใช้งานมากมาย ในโครงงานนี้จึงได้นำเอาความสามารถ ในการส่งข้อมูลติดต่อกันในระยะไกล ของระบบเครือข่ายอินเตอร์เน็ต มาประยุกต์ใช้ในการควบคุมชุดเครื่องมือวัด GPIB (General Purpose Interface Bus) ซึ่งประกอบด้วย HM8142 Programmable Function Generator, HM8130 Programmable Power Supply และ HM8112-2 Programmable Multimeter และ Oscilloscope โดยในโครงงานนี้ได้เขียนโปรแกรม เพื่อสร้างแบบจำลองหน้าปัดของอุปกรณ์ทั้ง 4 ตัวข้างต้น เพื่อให้ผู้ใช้สามารถใช้งานได้ง่ายและสะดวกขึ้น โดยใช้โปรแกรมภาษาเดลไฟล์ (Delphi) ซึ่งเป็นโปรแกรมแบบ visual programming languages ซึ่งสามารถสร้างแอปพลิเคชั่นบนวินโดว์ได้ง่ายและรวดเร็วด้วย Rapid Application Development (RAD) tools ซึ่งจะใช้เดลไฟล์ในการอินเตอร์เฟซกับ dynamic link libraries (DLLs) ของ NI-488 Software เพื่อควบคุมฮาร์ดแวร์ทั้ง 4 ตัวข้างต้น ซึ่งโครงสร้างของซอร์ฟแวร์ แสดงดังรูปที่ 1.1



รูปที่ 1.1 โครงสร้างของซอร์ฟแวร์โดยรวม

ส่วนในการเชื่อมต่อ ระบบควบคุมเครื่องวัดเข้ากับ เครือข่ายอินเตอร์เน็ตนั้นจะใช้ Delphi ในการติดต่อกับ MS SQL Sever ซึ่งเป็น software ที่เป็นตัวจัดการกับระบบฐานข้อมูล ที่มีลักษณะ

แบบ ไคล์เอนท์-เซิร์ฟเวอร์ (Client-Sever) ซึ่งจะใช้โพรโตคอล ICP/IP หรือ Transmission Control Protocol / Internet Protocol เป็นมาตรฐานในการติดต่อของเครือข่ายอินเตอร์เน็ต โดยภาพรวมของระบบแสดงได้ดังรูปที่ 1.2



รูปที่ 1.2 ระบบการควบคุม GPIB ผ่านเครือข่ายอินเตอร์เน็ต

โดยหลักการทำงานโดยรวมของระบบคือ คอมพิวเตอร์ของผู้ใช้ (GPIB User) จะทำการส่ง-รับ ข้อมูลกับฐานข้อมูล ซึ่งอยู่ที่เครื่อง MS SQL Sever และคอมพิวเตอร์ที่เป็นตัวอินเตอร์เฟซ (Interface) กับ ชุดเครื่องมือวัด GPIB ซึ่งรับข้อมูลมาจากดาต้าเบส และนำมาประมวลผลสั่งงานชุดเครื่องมือวัด และรับผลตอบสนองที่ได้รับจากเครื่องมือวัด ส่งให้กับดาต้าเบส (Database) เพื่อส่งให้กับคอมพิวเตอร์ของผู้ใช้ (GPIB User) ต่อไป

บทที่ 2

IEEE-488 (GPIB)

2.1 โครงสร้างของ IEEE-488

ในระบบพื้นฐานของ GPIB จะประกอบด้วยอุปกรณ์ คือ ผู้ส่ง (Talker), ผู้รับ (Listener) และ ผู้ควบคุม (Controller)

- Talker ทำหน้าที่ส่งข้อมูล โดยในระบบสามารถมี Talker ได้หลายตัว แต่จะมีเพียงตัวเดียว เท่านั้นที่กำลังทำงานอยู่

- Listener ทำหน้าที่เป็นตัวรับข้อมูล โดยในระบบเดียวกันสามารถมี Listener ได้หลายตัว เช่นเดียวกัน แต่ Listener สามารถทำงานได้ครั้งละหลาย ๆ ตัวได้

- Controller ทำหน้าที่ควบคุมอุปกรณ์ต่าง ๆ ในระบบ โดยจะกำหนดให้ Talker ทำการส่งข้อมูล หรือ กำหนดให้ Listener ทำการรับข้อมูล

อุปกรณ์ที่มี GPIB นั้นสามารถแบ่งตามหน้าที่ได้ดังนี้

1. ทำหน้าที่เป็น Talker เท่านั้น เช่น เครื่องมือวัด เป็นต้น

2. ทำหน้าที่เป็น Listener เท่านั้น เช่น เครื่องพิมพ์ (Printer), เครื่องบันทึก (Recorder) เป็นต้น

3. ทำหน้าที่เป็นทั้ง Talker และ Listener เช่น คอมพิวเตอร์, เครื่องมือวัดที่สามารถ ควบคุมได้จากภายนอก เป็นต้น

4. ทำหน้าที่เป็น Talker Listener และ Controller ในตัวเดียวกัน เช่น คอมพิวเตอร์ที่ทำหน้าที่ ควบคุมระบบ

2.2 ขีดจำกัดของ IEEE-488

1. จำนวนอุปกรณ์ในระบบ (Talker, Listener, Controller) ที่ต่อกับสายสัญญาณ 1 เส้น จะต้องไม่เกิน 15 เครื่อง

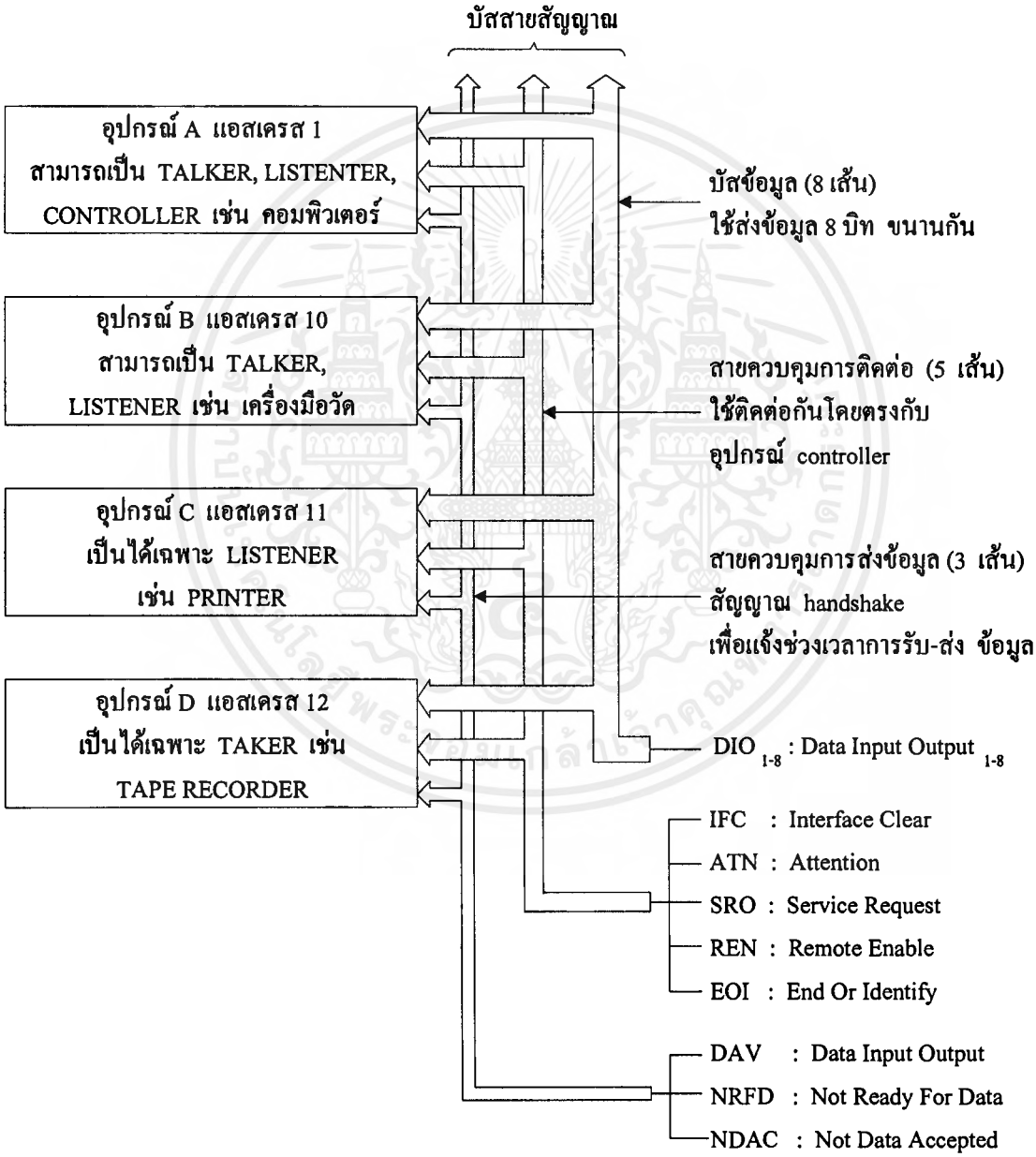
2. สายเคเบิ้ลที่ต่อระหว่างอุปกรณ์ จะต้องยาวไม่เกิน 4 เมตร และความยาวรวมของสายเคเบิ้ลในระบบจะต้องไม่เกิน 20 เมตร

3. ความเร็วในการส่งข้อมูลจะต้องไม่เกิน 1Mb/Sec (1 ล้านไบต์ต่อวินาที)

4. ต้องมีการจ่ายไฟให้กับอุปกรณ์มากกว่าครึ่งหนึ่งของระบบ

2.3 รายละเอียดเกี่ยวกับ IEEE-488

ลักษณะทางกายภาพ IEEE-488 นั้นคือ เป็นสายสัญญาณแบบ 24 เส้นขนานกัน และ มีขั้วต่ออยู่ทางปลายทั้งสองของสาย เพื่อต่อกับอุปกรณ์หรือต่อกันเพื่อให้สายสัญญาณมีความยาวเพิ่มขึ้น ในจำนวนสายสัญญาณ 24 เส้น มีเพียง 16 เส้นเท่านั้น ที่ทำหน้าที่นำสัญญาณ ส่วนเหลืออีก 8 เส้น ทำหน้าที่กราวนด์ (ground) และ ชีลด์ (shield)



รูปที่ 2.1 แสดงการแบ่งเส้นสายนำสัญญาณ

โดยจำนวนสายที่ใช้นำสัญญาณ 16 เส้นนั้นยังแบ่งได้เป็น 3 ประเภท ตามรูปที่ 2.1 คือ

1. บัสข้อมูล (Data Bus) จำนวน 8 สาย คือ

DI01 - DI08

2. สายสัญญาณควบคุม (Control Line) จำนวน 5 สาย คือ

IFC (Interface Clear)

ATN (Attention)

SRQ (Service Enable)

REN (Remote Enable)

EOI (End or Identify)

3. สายแฮนด์เชค (Hand Shake) 3 สาย คือ

DAV (Data Valid)

NRFD (Not Ready For Data)

NDAC (Not Data Accepted)

2.4 ความหมายของสัญญาณต่าง ๆ ภายใน IEEE-488

ดังที่ได้กล่าวมาแล้วว่าสายสัญญาณต่าง ๆ ใน GPIB ได้แบ่งออกเป็น 3 กลุ่ม ในหัวข้อนี้ จะอธิบายความหมายของสัญญาณต่าง ๆ ดังนี้

2.4.1 กลุ่มสัญญาณข้อมูล

1. DI01-DI08 (Data Input/Output) สายสัญญาณทั้ง 8 เส้นนี้ ทำหน้าที่เป็นทางผ่านของ ข้อมูลในระบบ

2.4.2 กลุ่มสัญญาณควบคุมการเชื่อมต่อ (Interface)

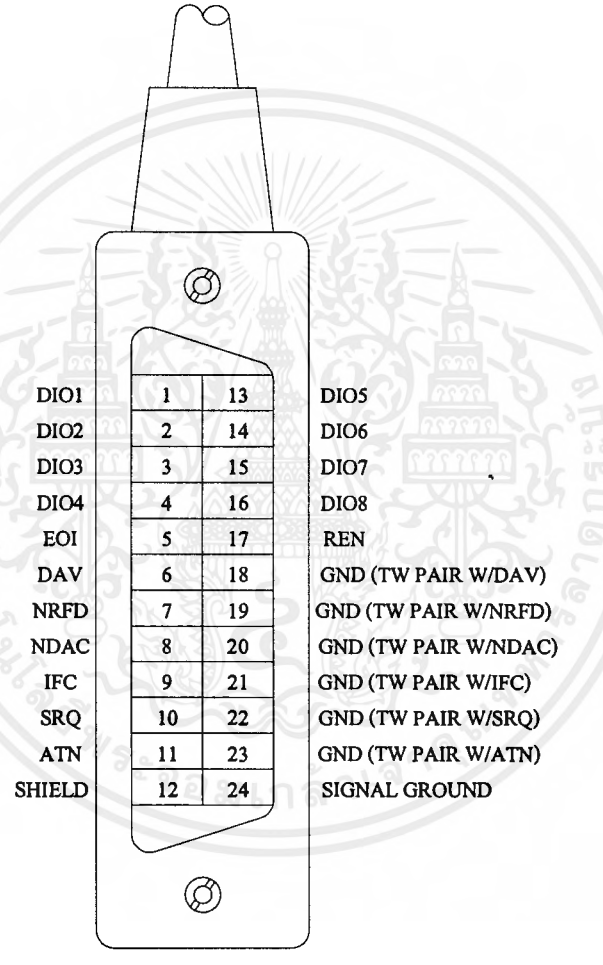
1. IFC (Interface Clear) เป็นสัญญาณรีเซต หรือ เคลียร์ระบบ กำเนิดได้โดยตัวควบคุม (Controller) เท่านั้น เมื่ออุปกรณ์ในบัสได้รับสัญญาณเคลียร์นี้จะกลับคืนสู่สภาวะเริ่มต้นใหม่ ซึ่งเป็น สภาวะแรกเริ่มก่อนการกำหนดฟังก์ชันเหมือนแรกเปิดสวิตซ์

2. ATN (Attention) เป็นสัญญาณที่ถูกส่งโดยอุปกรณ์ที่เป็นตัวควบคุมเช่นเดียวกัน ใช้ในการสั่งให้อุปกรณ์ทุกตัวในระบบเตรียมพร้อมเพื่อรอรับคำสั่งต่อไป

3. SRQ (Service Request) เป็นสัญญาณที่ถูกส่งจากอุปกรณ์ต่าง ๆ เพื่อเป็นการบอกแก่ระบบว่าขณะนี้อุปกรณ์ดังกล่าวต้องการติดต่อจากตัวควบคุม

4. REN (Remote Enable) สัญญาณนี้ เป็นสัญญาณที่ถูกส่งมาจากตัวควบคุมเพียงตัวเดียวเท่านั้น เพื่อใช้สั่งให้อุปกรณ์ต่าง ๆ เปลี่ยนจากโหมดที่ใช้งานปกติ มาเป็นการควบคุมโดยตัวควบคุม แทน

5. EOI (End or Identify) เป็นสัญญาณที่ถูกส่งได้โดยอุปกรณ์ที่เป็นตัวควบคุม (Controller) หรืออุปกรณ์ที่เป็นตัวส่ง (Talker) ก็ได้ ใช้สำหรับแสดงว่าข่าวสารที่ส่งเป็นชุดนั้นได้เสร็จสิ้นลงแล้ว



รูปที่ 2.2 ขั้วต่อของ GPIB และการจัดขาของสัญญาณต่าง ๆ

2.4.3 กลุ่มสัญญาณควบคุมการรับ-ส่งข้อมูล

1. DAV (Data Valid) เมื่อสัญญาณนี้ถูกดึงเป็นลอจิก "Low" โดยอุปกรณ์ที่เป็นตัวส่ง (Talker) เป็นการแจ้งแก่ระบบบัสว่า ขณะนี้ตัวส่งได้ทำการส่งข้อมูลลงไปที่สายสัญญาณข้อมูลเรียบร้อยแล้ว

2. NRFD (Not Ready For Data) เมื่อสัญญาณนี้มีลอจิกเป็น "Low" จะเป็นการแสดงว่าในขณะนี้ระบบบัสยังไม่พร้อมที่จะรับข้อมูล เนื่องจากอุปกรณ์ในระบบยังพร้อมไม่หมดทุกตัว ซึ่งสัญญาณเส้นนี้จะไม่เป็น "Hi" จนกว่าอุปกรณ์ทุกตัวให้ลอจิกที่เป็น "Hi" ครบถ้วนแล้ว สัญญาณนี้มีประโยชน์ในกรณีที่อุปกรณ์ในระบบมีความเร็วแตกต่างกัน

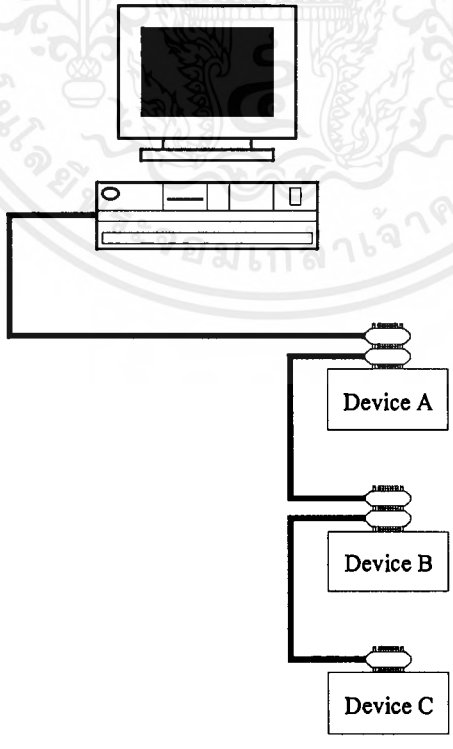
3. NDAC (Not Data Accepted) สัญญาณเส้นนี้เป็นสัญญาณที่ถูกควบคุมโดยอุปกรณ์ที่เป็น ตัวรับ (Listener) โดยสัญญาณนี้จะมีลอจิกเป็น "Low" ในขณะที่อุปกรณ์ที่เป็นตัวรับกำลังเก็บข้อมูล จากสายข้อมูล (Data Bus) และจะเป็น "Hi" เมื่ออุปกรณ์นั้นได้ทำการอ่านข้อมูลเสร็จเรียบร้อยแล้ว

โดยสัญญาณลอจิกที่ใช้ใน DATA BUS (D1-D8) ของ IEEE-488 มีลักษณะเป็นคอมพลีเมนต์ทั้งหมด คือ "1" เท่ากับ "Low" และ "0" เท่ากับ "Hi" ซึ่งตรงข้ามกับในวงจรที่เราคุ้นเคย

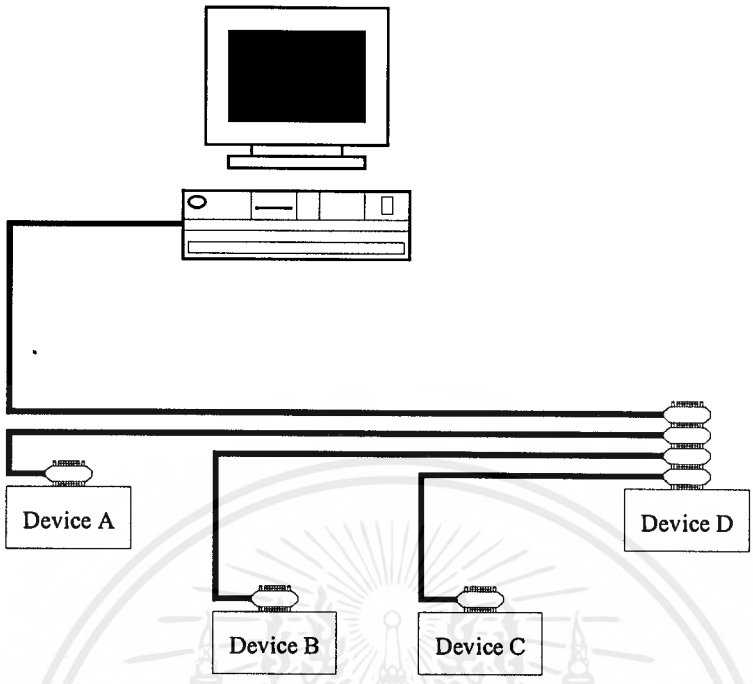
2.5 การเชื่อมต่ออุปกรณ์ต่าง ๆ ในระบบ IEEE-488 BUS

สำหรับการเชื่อมต่อระหว่างอุปกรณ์ต่าง ๆ ในระบบ IEEE-488 Bus นั้น มีอยู่ 2 วิธี คือ

1. การเชื่อมต่อแบบเรียงต่อเนื่องกัน (Daisy Chain Configuration)
2. การเชื่อมต่อแบบกระจาย (Star Configuration)



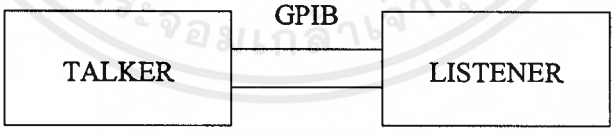
รูปที่ 2.3 การเชื่อมต่อแบบเรียงต่อเนื่องกัน (Daisy Chain Configuration)



รูปที่ 2.4 การเชื่อมต่อแบบกระจาย (Star Configuration)

2.6 ขบวนการแฮนด์เชค (Handshake Procedure)

เมื่อมีการเชื่อต่อระหว่างอุปกรณ์ต่าง ๆ ในระบบ ดังนั้น เมื่อระบบจะทำงานจึงต้องมีการตรวจสอบเสียก่อน ซึ่งการตรวจสอบนี้เราเรียกว่าขบวนการแฮนด์เชค (Handshake Procedure)



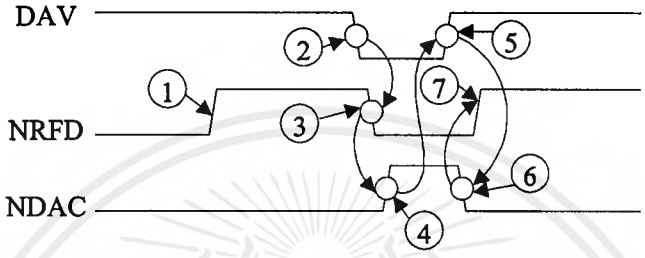
รูปที่ 2.6 ขบวนการแฮนด์เชค (Handshake Procedure)

พิจารณาระบบที่ไม่ซับซ้อนนัก โดยกำหนดให้มีตัวส่ง (Talker) และตัวรับ (Listener) อย่างละ 1 ตัว

การที่จะมีการสื่อสารระหว่างตัวส่งและตัวรับจะกระทำได้โดยการที่ตัวส่ง จะทำหน้าที่ส่งสัญญาณออกมาเพื่อให้ตัวรับทราบว่ามีข้อมูลเสร็จในสายสัญญาณ และตัวรับก็จะส่งสัญญาณเพื่อให้ ตัวส่งทราบว่าได้ทำการรับข้อมูลเป็นที่เรียบร้อยแล้ว

การส่งสัญญาณในการสื่อสารระหว่างตัวส่งและตัวรับนั้น จะส่งมาทางสายสัญญาณ 3 สาย ซึ่งสายสัญญาณดังกล่าวนั้นเป็นสายสัญญาณในกลุ่มควบคุมการรับ-ส่งข้อมูล คือ NRFD (Not Ready For Data), DAV (Data Valid) และ NDAC (Not Data Accepted)

DAV เป็นสัญญาณที่ถูกส่งโดยตัวส่ง NRFD และ NDAC เป็นสัญญาณที่ถูกส่งโดยตัวรับ โดยตัวรับจะใช้สัญญาณ NDAC เพื่อชี้ให้เห็นว่าพร้อม หรือไม่พร้อมที่จะรับข้อมูล

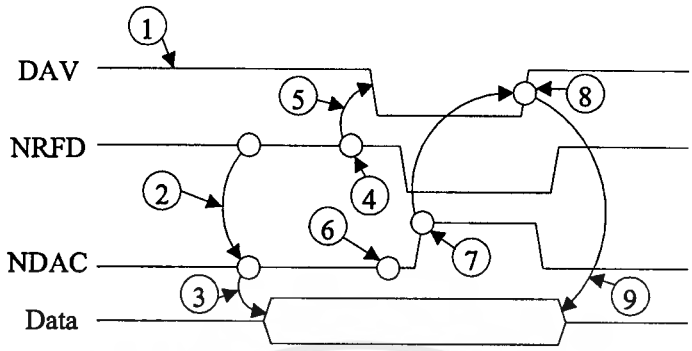


NOTE: The timing shown is relative

รูปที่ 2.7 timing diagram ของขบวนการแฮนด์เชค

ขบวนการแฮนด์เชคเริ่มขึ้นเมื่อ ตัวส่งมีข้อมูลที่จะถ่ายทอดลงสายสัญญาณไปยังตัวรับเริ่มด้วย การที่ตัวรับจะทำให้สัญญาณ NRFD เป็น "Hi" (มีค่าเป็น 0) นั่นคือ เป็นการบอกให้ทราบว่าตัวรับ พร้อมที่จะให้ตัวส่งทำการถ่ายข้อมูลลงในสายสัญญาณแล้ว (ในจุดที่ 1) ตัวส่งจะทำการถ่ายข้อมูล ลงในสายสัญญาณ DI01-DI08 หลังจากทำการรอเวลา เพื่อให้ตัวส่งถ่ายข้อมูลแล้วตัวส่งจะทำให้ สัญญาณ DAV มีลอจิกเป็น "Low" (มีค่าเป็น 1) เพื่อที่จะบอกให้ทราบว่าขณะนี้ได้มีข้อมูลในสาย สัญญาณทั้ง 8 เส้นแล้ว (ในจุดที่ 2) ต่อมาเมื่อผู้รับทราบว่ามีข้อมูลอยู่ในบัสก็จะทำให้สัญญาณ NRFD ให้มีค่าเป็น "Low" เพื่อบอกให้ตัวส่งทราบว่ายังไม่พร้อมที่จะให้ตัวส่งทำการส่งข้อมูลชุดต่อไป และตัวรับพร้อมที่จะเก็บข้อมูลจากสายสัญญาณ (ในจุดที่ 3) หลังจากตัวรับทำการเก็บข้อมูลจาก สายสัญญาณเสร็จแล้ว ตัวรับจะทำให้สัญญาณ NDAC ให้มีสถานะเป็น "Hi" เพื่อบอกให้ทราบว่าได้ ทำการเก็บข้อมูลเสร็จแล้ว (ในจุดที่ 4) เมื่อตัวส่งตรวจพบว่าสัญญาณ NDAC มีลอจิกเป็น "Hi" ก็จะ ทำให้สัญญาณ DAV ให้มีลอจิกเป็น "Hi" ด้วย เพื่อไม่ให้ตัวรับทำการรับข้อมูลในบัสอีก (ในจุดที่ 5) เมื่อตัวรับทราบว่าสัญญาณ DAV มีลอจิกเป็น "Hi" ก็จะทำให้สัญญาณ NDAC ให้เป็น "Low" ทำให้ ข้อมูลในบัสถูกกำจัดออกไป หลังจากนั้นก็จะทำให้สัญญาณ NRFD ให้เป็น "Hi" เพื่อบอกให้ทราบ ว่าพร้อมที่จะรับข้อมูลชุดต่อไปแล้วขบวนการแฮนด์เชคจึงเสร็จสิ้นลง และจะเริ่มขึ้นใหม่เมื่อมีสัญญาณ ชุดใหม่เข้ามา

2.7 ขบวนการแฮนด์เชคของตัวควบคุมซึ่งทำหน้าที่เป็นตัวส่ง

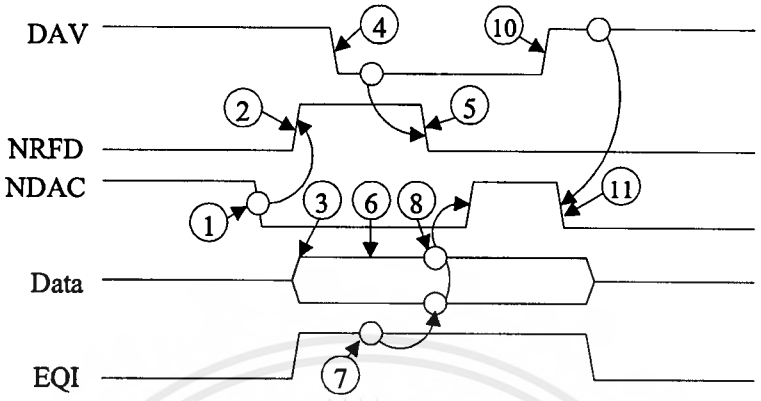


รูปที่ 2.8 timing diagram ของขบวนการแฮนด์เชคของตัวควบคุมซึ่งทำหน้าที่เป็นตัวส่ง

ขบวนการแฮนด์เชคเริ่มขึ้นโดยการที่ตัวควบคุม ทำการเซตค่าให้สัญญาณ DAV มีค่าเป็น "Hi" (ในจุดที่ 1) ต่อมาเป็นการตรวจสอบว่าสัญญาณ NDAC และ NRFD มีลอจิกเป็น "Hi" ทั้งคู่หรือไม่ (ในจุดที่ 2) ถ้าทั้งคู่เป็น "Hi" นั่นคือเกิดการผิดปกติ คือ อุปกรณ์นั้นไม่พร้อมที่จะทำงาน จึงทำการออกจากขบวนการแฮนด์เชค

กลับมาดูในจุดที่ 2 หากสัญญาณใดสัญญาณหนึ่งมีลอจิกเป็น "Low" ตัวควบคุมจะทำการส่ง ข้อมูลลงใน GPIB Bus (ในจุดที่ 3) หลังจากนั้นตัวควบคุมจะทำการตรวจสอบว่าสัญญาณ NRFD มี ลอจิกเป็น "Hi" (ในจุดที่ 4) หลังจากนั้นตัวควบคุมจะทำให้สัญญาณ DAV ให้มีลอจิกเป็น "Low" เพื่อให้ตัวรับทราบว่ามีข้อมูลอยู่ในบัสข้อมูล (ในจุดที่ 5) ตัวควบคุมจะรอเวลาเพื่อให้ตัวรับทำการเก็บ ข้อมูล เมื่อครบกำหนดเวลาตัวควบคุมจะทำการเชคว่าตัวรับทำงานเกินเวลาที่กำหนด หรือไม่หากเกิน เวลาก็จะทำการตรวจสอบต่อไปว่าสัญญาณ NDAC ทำลอจิกเป็น "Hi" ใช่หรือไม่ก็จะทำการ ตรวจสอบอีกครั้งว่าเกินเวลาหรือไม่ กลับไปดูที่จุดที่ 6 หากสัญญาณ NDAC เป็น "Hi" (ในจุดที่ 7) ตัวควบคุมจะทำการตอบสนองโดยการทำสัญญาณ DAV ให้มีลอจิกเป็น "Hi" เพื่อบอกให้ตัวรับ ทราบว่าให้หยุดการเก็บข้อมูลจากบัส (ในจุดที่ 8) หลังจากนั้นข้อมูลในบัสจะถูกนำออกไป (จุดที่ 9) จึงเป็นการเสร็จสิ้นขบวนการแฮนด์เชค

2.8 ขบวนการแฮนด์เชคของตัวควบคุมซึ่งทำหน้าที่เป็นตัวรับ



รูปที่ 2.9 timing diagram ของขบวนการแฮนด์เชคของตัวควบคุมซึ่งทำหน้าที่เป็นตัวรับ

ขบวนการแฮนด์เชคเริ่มขึ้น หลังจากการที่ตัวควบคุมรับรู้ว่าตัวส่งจะทำการส่งข้อมูลเมื่อตัวควบคุมทราบว่าตัวส่งจะทำการส่งข้อมูล ตัวควบคุมรับรู้ว่าตัวส่งจะทำให้สัญญาณ NDAC มีลอจิกเป็น "Low" เพื่อบอกให้ทราบว่าตัวควบคุมยังไม่ได้ทำการเก็บข้อมูล (ข้อมูลในที่นี้คือ แอดเดรส หรือ ข้อมูลทั่วไปซึ่งตัวส่งทำการส่งมาในบัสข้อมูล) นั่นคือในจุดที่ 1

ต่อมาตัวควบคุมจะทำการเซตค่าสัญญาณ NRFD ให้เป็น "Hi" เพื่อที่จะให้ตัวส่งทราบว่าขณะนี้ตัวควบคุมพร้อมที่จะรับข้อมูลแล้ว (ในจุดที่ 2) เมื่อสัญญาณ NRFD มีค่าเป็น "Hi" และสัญญาณ NDAC มีค่าเป็น "Low" แล้ว ตัวส่งจะทราบทันทีว่าทำการส่งข้อมูลลงมาในบัสข้อมูล ได้แล้ว (ในจุดที่ 3) หลังจากนั้นตัวควบคุมจะทำการรอเวลาให้ตัวส่งทำการส่งข้อมูลลงในบัสข้อมูลให้เสร็จ โดยจะทำการตรวจสอบไปด้วยว่าเกินเวลาที่กำหนดหรือยัง หากเกินเวลาที่กำหนดก็จะออก จากขบวนการแฮนด์เชค หากไม่เกินกำหนดเวลาจะทำการตรวจสอบเวลาว่าเกินเวลาที่ต้องคอยแล้ว หรือยัง หากยังไม่เกินเวลาก็จะทำการเชคต่อไปจนกระทั่งหมดเวลาหรือจนกว่าสัญญาณ DAV เป็น "Low" หรือไม่หากไม่ก็จะตรวจสอบเวลาว่าเกินเวลาที่ต้องรอแล้วหรือยัง หากยังไม่เกินเวลาก็จะทำ การเช็คต่อไปจนกระทั่งหมดเวลาหรือจนกว่าสัญญาณ DAV จะมีค่าเป็น "Low"

เมื่อสัญญาณ DAV เป็น "Low" แล้วตัวควบคุมจะตอบรับ โดยการทำให้สัญญาณ NRFD มีค่าเป็น "Low" (ในจุดที่ 5) นั่นเป็นการบอกว่าตัวควบคุมพร้อมที่จะเริ่มเก็บข้อมูล (ในจุดที่ 6) หลังจากนั้นตัวควบคุมจะทำการตรวจสอบสัญญาณ EOI เพื่อตรวจสอบว่าข้อมูลที่ตัวส่งทำการส่งลง มานั้นหมดแล้วหรือยัง (ถ้าตัวส่งทำการส่งข้อมูลลงในบัสหมดแล้วจะทำการตั้งค่าสัญญาณ EOI ให้เป็น "Low") (ในจุดที่ 7) หากสัญญาณ EOI เป็น "Low" ตัวควบคุมจะเซตสถานะว่า ขณะนี้ตัวส่งได้

ทำการส่งข้อมูลลงมาในบัสหมดแล้ว หากสัญญาณ EOI เป็น "Hi" ก็ไม่ทำการเซตสถานะ หลังจากนั้นตัวควบคุมจะทำการเป็นข้อมูลในบัส (ในจุดที่ 8) แล้วทำให้สัญญาณ NDAC มีค่าเป็น "Hi" เพื่อบอกให้ทราบว่าตัวควบคุมได้ทำการอ่านข้อมูลเสร็จเรียบร้อยแล้ว (ในจุดที่ 9) เมื่อตัวควบคุมทำให้สัญญาณ NDAC เป็น "Hi" แล้วตัวส่งจะลบข้อมูลในบัสข้อมูล โดยการทำให้สัญญาณ DAV เป็น "Hi" (ในจุดที่ 10) ซึ่งก่อนหน้านี้ตัวควบคุมจะทำการตรวจสอบอยู่แล้วว่าสัญญาณ DAV เป็น "Hi" หรือยัง

เมื่อสัญญาณ DAV เป็น "Hi" แล้วตัวควบคุมจะทำให้สัญญาณ NDAC มีค่าเป็น "Low" (ในจุดที่ 11) แล้วจึงออกจากขบวนการแฮนด์เชค

2.9 คำสั่งใช้งานของ GPIB

การสั่งการต่าง ๆ เพื่อกำหนดหน้าที่การทำงานและกำหนดฟังก์ชัน เช่น กำหนดช่วงการวัด โหมดการวัด หรืออื่น ๆ แก่เครื่องวัดที่ตกอยู่เหล่านี้นั้น ตัวควบคุมจะเป็นตัวกำหนดการส่งรหัสคำสั่ง ไปที่อุปกรณ์โดยผ่าน DI1-DI6 รหัสคำสั่งนี้จะถูกส่งไปในช่วงที่สายสัญญาณ ATN เป็น LOW

คำสั่งสำหรับกำหนดหน้าที่การงานต่าง ๆ ตามมาตรฐานของ GPIB มีอยู่ด้วย 128 คำสั่ง ดังแสดงในตารางที่ 2.1 ต่อไปนี้ โดยแบ่งเป็น 5 กลุ่มคำสั่ง

รหัสที่ใช้ในระบบ GPIB บัสนั้นใช้ร่วมกัน ทั้งรหัสข้อมูลและรหัสคำสั่ง นั่นคือรหัสเดียวกัน มีความหมายได้ 2 อย่าง คือ เมื่อ ATN เป็น LOW จะหมายถึงรหัสคำสั่ง แต่ ATN เป็น HIGH รหัสนี้ จะแทนข้อมูลที่เป็น ASCII แทน ซึ่งในตารางก็ได้แบ่งความหมายออกเป็น 2 คอลัมน์

1. กลุ่มคำสั่งเจาะจงจุดหมาย (addressed command group) เป็นคำสั่งที่ส่งไปยังอุปกรณ์ที่เป็นตัวส่งหรือตัวรับที่กำหนดไว้ล่วงหน้าแล้ว คำสั่งนี้ประกอบด้วย

GTL (got to local) สั่งให้อุปกรณ์กลับสู่สภาพการควบคุมปกติด้วยมือ

SDC (selected device clear) สั่งให้อุปกรณ์เคลียร์ตัวเองสู่สภาพเริ่มต้นใหม่

PPC (paralled poll configure) เป็นคำสั่งสำหรับทำการจัดสายสัญญาณของการทำการ จัดสรรสายสัญญาณของการทำกระบวนการตรวจสอบสภาพอุปกรณ์โดยวิธีขนาน โดยใช้กับกลุ่มคำสั่ง รอง

GET (group excute trigger) ใช้สั่งเริ่มต้นการทำงานของอุปกรณ์ทีละหลายตัว

TCT (take control) เป็นการกำหนดให้อุปกรณ์ตัวส่งทำหน้าที่เป็นตัวควบคุม

2. กลุ่มคำสั่งครอบคลุม (universal command group) เป็นคำสั่งที่ส่งไปยังอุปกรณ์ทุกตัวที่ต่ออยู่ในบัส ประกอบด้วย

LLO (local lockout) เป็นการสั่งให้อุปกรณ์ล็อคอยู่ที่สภาวะควบคุมโดยปุ่มปรับที่หน้าปัดตามปกติ

DCL (device clear) สั่งให้อุปกรณ์ทุกตัวกลับไปสู่สภาวะเริ่มต้น

PPU (parallel poll unconfigure) ใช้ยกเลิกกระบวนการตรวจสอบสภาพแบบขนาดทั้งหมด

SPE (serial poll enable) เปลี่ยนโหมดของการตรวจสอบสภาพเป็นแบบอนุกรมในโหมดนี้ จะเป็นการส่งสถานะของเครื่องแทนการส่งข้อมูล

SPD (serial poll disable) ยกเลิกโหมดการตรวจสอบแบบอนุกรม

3. กลุ่มคำสั่งกำหนดอุปกรณ์ตัวรับ (listener address group) เป็นคำสั่งสำหรับกำหนดให้อุปกรณ์เป็นตัวรับ ตามรหัสหมายเลขจาก 0-30 และมีคำสั่ง UNT (untalker) สำหรับยกเลิก

4. กลุ่มคำสั่งกำหนดอุปกรณ์ตัวส่ง (talker address group) สำหรับกำหนดให้อุปกรณ์เป็นตัวส่ง ตามรหัสหมายเลขจาก 0-30 และมีคำสั่ง UNL (unlisten) สำหรับยกเลิกเช่นกัน

คำสั่งกลุ่มที่ 1 ถึง 4 นั้น จัดเป็นกลุ่มคำสั่งหลักที่มีความหมายตายตัวยังมีคำสั่งอีกกลุ่มที่ขึ้นอยู่กับการกำหนดภายหลังนั่นคือ กลุ่มคำสั่งรอง

5. กลุ่มคำสั่งรอง (secondary command group) เป็นคำสั่งที่กำหนดรายละเอียดย่อยของอุปกรณ์แต่ละตัวที่ต่ออยู่ในระบบ ให้มีการทำงานอย่างไร ตามจุดประสงค์ใช้งานของเครื่องมือนั้น เช่นเดียวกับการปรับปุ่มต่าง ๆ ด้วยมือนั่นเอง คำสั่งรองนี้จะตามหลังคำสั่งหลัก คือจะใช้หลังจากอุปกรณ์ต่าง ๆ ถูกกำหนดวางตัวในระบบเรียบร้อยแล้ว

คำสั่งต่าง ๆ ที่กล่าวไป ซึ่งใช้ในการกำหนดสภาวะการทำงานของอุปกรณ์ แต่ละสภาวะ ที่กำหนดไปนั้นเป็นอย่างไร และมีจุดประสงค์เพื่ออะไร ดังต่อไปนี้

Device clear / Interface Clear

Device clear ใช้ในการทำให้อุปกรณ์ที่ต่ออยู่ในบัสกลับไปสู่สภาวะเริ่มต้น ยังไม่มีการกำหนดฟังก์ชันใด ๆ สภาวะเริ่มต้นนี้จะแตกต่างกันไป แล้วแต่ว่าอุปกรณ์นั้นออกแบบไว้อย่างไร Device clear มีอยู่ 2 ลักษณะ คือ เคลียร์หมดทุกตัวที่ต่ออยู่ (DCL) กับเคลียร์เจาะจงอุปกรณ์ตัวใดตัวหนึ่ง (SDC)

แต่ว่าในการเคลียร์อุปกรณ์ให้อยู่ในสภาวะเริ่มต้นนั้นไม่ได้หมายความว่า Interface function ของ GPIB จะถูกเคลียร์อุปกรณ์ให้ไปอยู่ในสภาวะเริ่มต้นด้วยแต่อย่างใด interface function คือ สภาพการ interface ที่ได้กำหนดไว้ในระบบประกอบด้วยฟังก์ชันต่าง ๆ ดังแสดงในตารางที่ 2.2

ตารางที่ 2.2 แสดง interface function

ฟังก์ชัน	สัญลักษณ์	การกลับสู่จุดเริ่มต้นโดย IFC
source handshake	SH	ได้
acceptor handshake	AH	ได้
talker or enlarge talker	T or TE	ได้
listener or enlarge listener	L or LT	ได้
service request	SR	ไม่ได้
remote / local	RL	ไม่ได้
parallel poll	PP	ไม่ได้
device clear	DC	ได้
device trigger	DT	ได้
controller	C	ได้

Remote / Local

remote เป็นการกำหนดให้อุปกรณ์ที่อยู่ในระบบเช่น เครื่องมือวัดให้อยู่ในการควบคุมของอุปกรณ์ตัวอื่นแทน ซึ่งปุ่มปรับต่าง ๆ บนหน้าปัดเครื่องจะไม่มีผลต่อการทำงาน ส่วน Local เป็นการ ควบคุมการทำงานของเครื่องมือวัดด้วยปุ่มปรับบนหน้าปัดตามปกติ

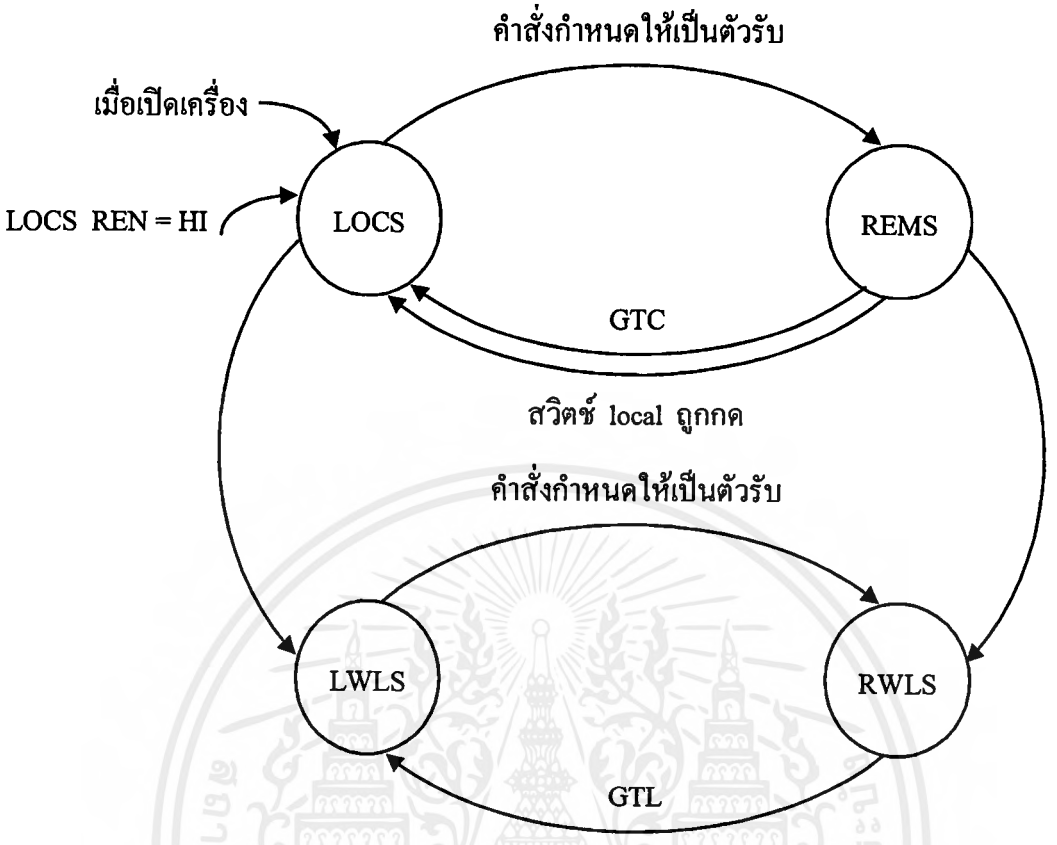
การใช้ remote มีประโยชน์ในแง่ที่ขณะที่ตัวการควบคุมเช่น คอมพิวเตอร์กำลังติดต่ออุปกรณ์ ตัวนั้นอยู่ หากไม่มีการตัดการควบคุมโดยปุ่มปรับบนหน้าปัดออก ถ้ามีใครมาปรับแต่งก็จะทำให้การ ทำงานผิดพลาดไปได้ การทำงานของ GPIB ใน remote and local มี 4 ลักษณะดังนี้

1. LOCS ก็คือ local นั่นเอง เป็นสภาพการควบคุมที่ปุ่มตามปกติ จะอยู่ในสภาพนี้เปิดตอนเครื่องหรือ REN เป็น HIGH หรือเมื่อได้รับคำสั่ง GTL

2. REMS คือ remote หมายถึงการตัดการควบคุมโดยปุ่มหน้าปัดออก จะเกิดขึ้นเมื่อ REN เป็น LOW และจะถูกล็อกไว้ เว้นแต่ว่าสวิตช์ local ที่ตัวอุปกรณ์จะถูกเปลี่ยนไปตำแหน่ง Local

3. RWLS เป็นสภาพ remote ที่ถูกล็อกเอาไว้เช่นกัน แต่ว่าจะตัดการควบคุมตรงสวิตช์ local ที่ตัวอุปกรณ์ออกไป สภาพ remote โดย RWLS จึงมีความสำคัญสูงกว่า REMS อย่างไรก็ตามยังถูกยกเลิกได้ด้วยคำสั่ง LLO

4. LWLS มีสภาพเช่นเดียวกับ local แต่จะแตกต่างกันตรงที่สภาพ local โดย LWLS นี้เมื่อได้รับคำสั่งกำหนดอุปกรณ์ตัวรับจะเปลี่ยนไปอยู่ในสภาพแบบล็อกหรือ RWLS ทันทีในการที่จะ มาที่สภาพ LWLS นี้ได้ก็มี 2 กรณีคือ เมื่ออยู่ในสภาพ local ธรรมดา (LOCS) แล้วได้รับคำสั่ง LLO หรืออยู่ใน RWLS แล้วได้รับคำสั่ง GTL



รูปที่ 2.10 แสดง loop การทำงานของ GPIB ใน remote/local

2.10 การขอบริการและการตรวจสอบ (Service Request and Polling)

เมื่อตัวควบคุมได้รับ SRQ เป็น LOW จะให้อุปกรณ์ส่งข้อมูลแสดงสถานะการทำงาน ซึ่งมี อยู่ 2 วิธีคือ

1. การตรวจสอบแบบอนุกรม ซึ่งมีขั้นตอนดังนี้

1.1 ATN ถูกดึงเป็น LOW หลังจากได้รับ LOW จากสายสัญญาณ SRQ

1.2 คำสั่ง UNL ถูกสั่งไปยังอุปกรณ์

1.3 ตัวควบคุมจะแจ้งรหัสตัวรับของตน และกำหนดรหัสตัวส่งอุปกรณ์ที่จะตรวจสอบไป ที่บัส

1.4 ตามด้วยคำสั่ง SPE และสาย ATN กลายเป็น HI ซึ่งอุปกรณ์ที่ถูกเรียกจะส่งข้อมูลแสดงสถานะออกมา 1 ไบต์ โดยบิตที่ 7 จะเป็นตัวชี้ว่าอุปกรณ์เส้นเป็นตัวขอบริการ ถ้าใช่จะเป็น LOW ส่วนบิตอื่น ๆ ก็ใช้บอกข้อมูลอื่น ๆ ซึ่งมิได้กำหนดเฉพาะ

1.5 สาย ATN ถูกดึงเป็น LOW อีกที เพื่อส่งคำสั่งยกเลิกการตรวจสอบคือ SPD

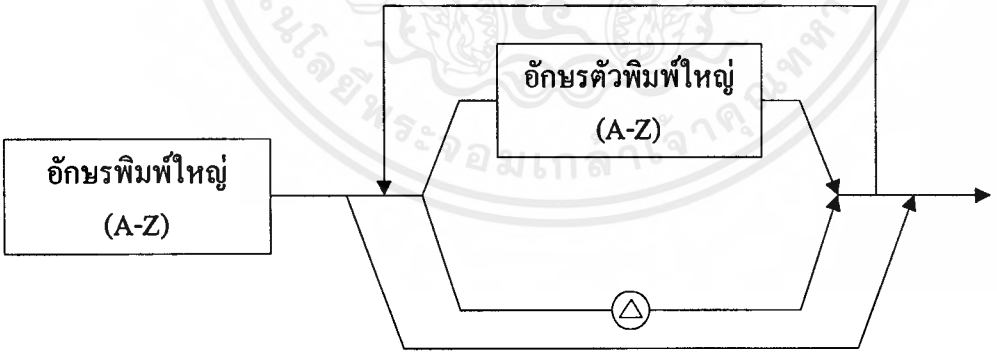
1.6 จากนั้นคำสั่ง UNT ก็ถูกส่งไปยังอุปกรณ์เพื่อยกเลิกการเป็นตัวส่ง ซึ่งถ้าหาก SQR ยังคงเป็น LOW อยู่ ก็จะมีการตรวจสอบไปยังอุปกรณ์ตัวอื่น ๆ ต่อไปตามขั้นตอนเดิม

2. การตรวจสอบแบบขนาน สามารถทำได้เร็วกว่าแบบอนุกรมทั้งนี้เพราะสามารถอ่านข้อมูล เพียงไบต์เดียวก็สามารถรู้ได้ทันที ว่าอุปกรณ์ตัวใดเป็นผู้ขอบริการ

2.11 รูปแบบของข้อมูล

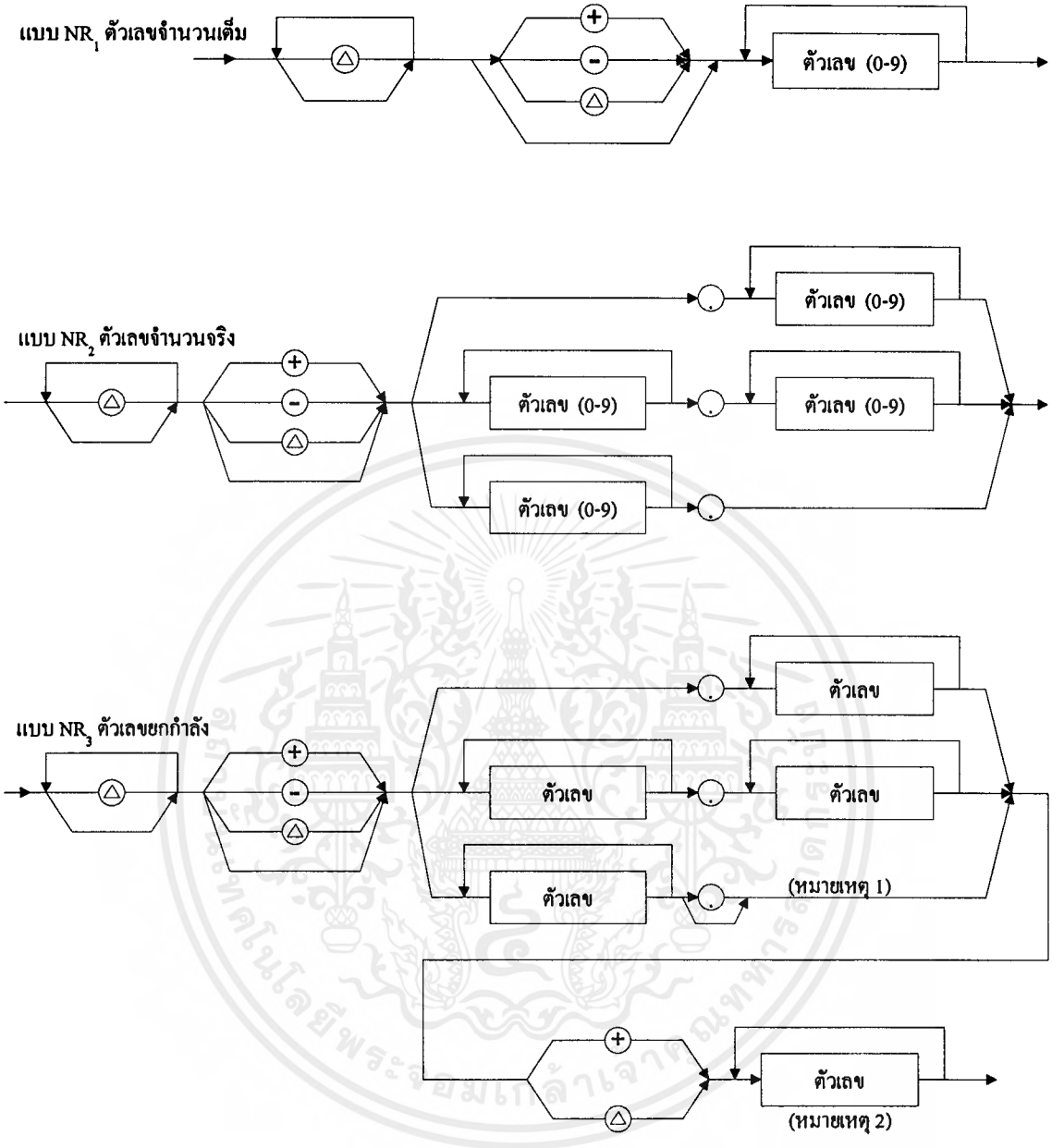
โดยทั่วไปข้อมูลจากอุปกรณ์ (device message) แบ่งได้ออกเป็น 3 ส่วนดังแสดงในรูปที่ 2.11 อันได้แก่ส่วนหัว (HR) ซึ่งจะอยู่ส่วนหน้าสุด เป็นตัวบอกชนิดข้อมูล ส่วนประกอบ HR แสดงในรูป ที่ 10 จะเห็นว่าประกอบด้วยตัวอักษรภาษาอังกฤษตัวพิมพ์ใหญ่ช่องว่างที่เป็นเว้นวรรค ( ) ปกติจะมี อักษรประมาณ 1-3 ตัว

รูปที่ 2.11 รูปแบบของข้อมูล



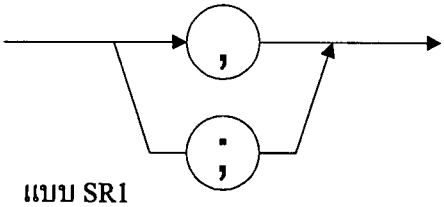
รูปที่ 2.12 รูปแบบของ HR

ส่วนที่ 2 คือ เนื้อหาข้อมูล (NR) ซึ่งใช้แสดงค่าตัวเลข มีอยู่ 3 แบบ คือ NR1, NR2 และ NR3 ดังแสดงในรูปที่ 2.13 ส่วนท้ายของ NR ยังอาจมีตัวอักษรแสดงหน่วยตามมา

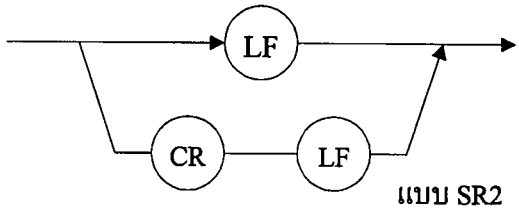


รูปที่ 2.13 แสดงเนื้อหาข้อมูล

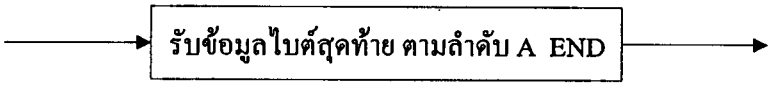
ส่วนที่ 3 คือ สัญญาณแบ่งข้อมูลแต่ละชุด (SR) ดังแสดงในรูปที่ 2.14 โดน SR1 ใช้แสดงการ ต่อเนื่องของข้อมูล (ข้อมูลยังมีต่อ) SR2 และ SR3 แสดงการสิ้นสุดของข้อมูล แต่ SR3 เป็นการ บอกการเสร็จสิ้นข้อมูลทั้งหมดจากการวัด



แบบ SR1



แบบ SR2



แบบ SR3

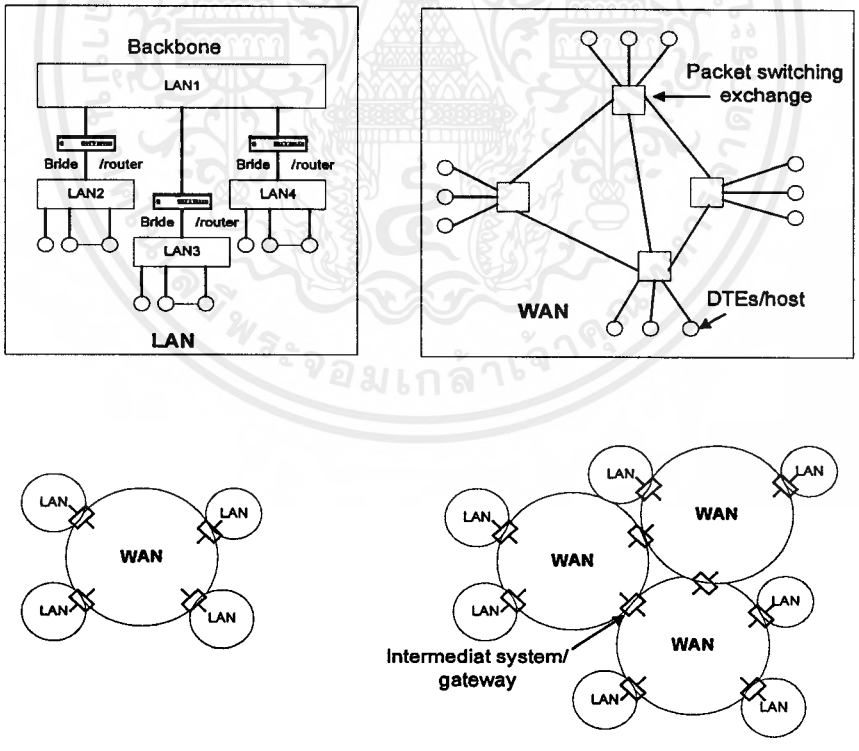
รูปที่ 2.14 แสดงสัญญาณแบ่งข้อมูลแต่ละชุด

บทที่ 3
เครือข่ายอินเตอร์เนต

Internet คือ การที่เครือข่าย 2 หรือมากกว่า เชื่อมต่อเข้าด้วยกัน และทำงานเสมือนเป็นเครือข่ายเดียวกัน โดย network ที่เป็นส่วนประกอบของ internet คือ Subnetwork (Subnet) ซึ่งอาจจะเป็นเครือข่าย Local area network (LAN) หรือ Wide area network (WAN) อุปกรณ์ที่ใช้ในการเชื่อมต่อ 2 เครือข่าย เข้าด้วยกันก็คือ intermediate system (IS) หรือ internetworking unit (IWU) การเชื่อมโยงระหว่างระบบที่แตกต่างกัน จำเป็นต้องมีมาตรฐานในการติดต่อกัน ซึ่งเรียกเป็นศัพท์เฉพาะว่า โพรโตคอล (Protocol)

3.1 สถาปัตยกรรม อินเตอร์เน็ต (Internet Architectures)

สถาปัตยกรรมพื้นฐานของ Internet แสดงดังรูปที่ 3.1

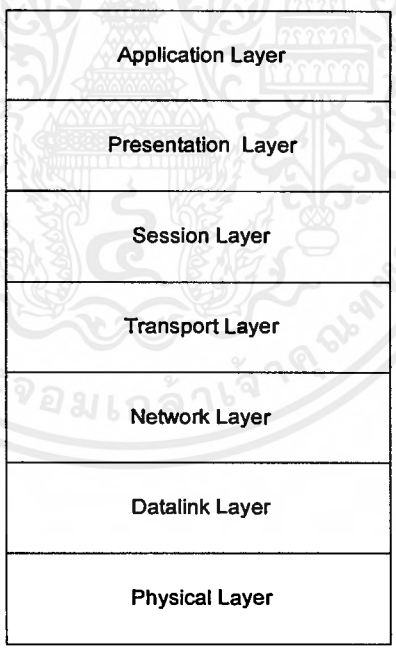


รูปที่ 3.1 แสดงสถาปัตยกรรมของ Internet (a) Single LAN and WAN (b) Interconnected LAN/WAN

ในรูป (a) แสดงตัวอย่าง 2 ตัวอย่างของเครือข่ายเดี่ยว (Single network) ซึ่งอย่างแรกเป็น site-wide LAN ซึ่งประกอบขึ้นมาจากชุดของ LANs ซึ่งถูกต่อเข้ากับเครือข่ายหลัก (backbone) ซึ่งอุปกรณ์ที่ใช้ต่อ LAN เข้ากับเครือข่ายหลัก ถ้า LAN ทุกเครือข่ายมีระบบเดียวกันก็จะใช้ bridge ถ้าเป็น LAN ที่แตกต่างกันก็จะใช้ router ตัวอย่างที่ 2 เป็นตัวอย่างของ WAN เดี่ยว ๆ ในรูป (b) แสดงถึงเครือข่ายอินเตอร์เน็ต ซึ่งประกอบด้วย network ทั้ง 2 ชนิด ข้างต้น

3.2 OSI โมเดล

องค์กรมาตรฐานสากล ISO (International Organization for Standardization) ได้กำหนดมาตรฐานของเครือข่าย โดยจัดแบ่งกิจกรรมของเครือข่าย ออกเป็นงานย่อย ๆ และกำหนดโมเดลแบ่งเป็นชั้น ๆ ตามลำดับเรียกว่ามาตรฐาน OSI (Open System Interconnection) โดยที่จะแบ่งกิจกรรมที่ซับซ้อนในเครือข่าย ออกเป็นงานย่อย ๆ ก็จะช่วยในการออกแบบ และการใช้งานเครือข่าย รวมถึงการเชื่อมโยงกัน เป็นไปได้ด้วยความสะดวก และมีวิธีการทำงานอยู่ในกรอบเดียวกัน ดังในรูปที่ 3.2



รูปที่ 3.2 แสดงการแบ่งการทำงานของเครือข่ายออกเป็น OSI model

ในแต่ละชั้นของ OSI model จะมีการติดต่อสื่อสารกันเป็นชั้น ๆ ตามลำดับลงมาเช่น Application Layer ก็จะติดต่อสื่อสารกับ Presentation Layer ตามลำดับไปจนถึงชั้นแรกสุดคือ Physical Layer

Application Layer เป็นชั้นบนสุดของ โมเดลเป็นส่วนที่จะทำให้การติดต่อระหว่างเครือข่ายกับผู้ใช้ เป็นไปได้ตามต้องการ ตัวอย่างแอปพลิเคชั่นของเครือข่าย เช่น ระบบ e-mail, การโอนถ่ายข้อมูล (File Transfer), การขอเข้าใช้ระบบคอมพิวเตอร์ในเครือข่ายเป็นต้น

Presentation Layer มีการกำหนดหน้าที่ไม่ชัดเจนนักและมีการนำไปใช้ไม่มาก ซึ่งหน้าที่หลักก็คือ เป็นส่วนที่จัดรูปแบบและนำเสนอข้อมูล ให้เป็นไปตามต้องการ รวมไปถึงการจัดแปลงข้อมูล ในรูปมาตรฐาน ASCII หรือ EBCDIC, การลดขนาดข้อมูล (data compression) การเข้ารหัสหรือถอดรหัสของข้อมูล แต่ส่วนใหญ่แล้ว แอปพลิเคชั่นจะจัดการแทนได้

Session Layer เป็นชั้นที่จัดการในเรื่อง "การติดต่อแต่ละครั้ง" หรือ session ให้ระบบคอมพิวเตอร์ทั้งสองฝั่ง โดยทำหน้าที่ตั้งแต่เริ่มการติดต่อ ดูแลในการส่งผ่านข้อมูล ในการติดต่อครั้งนั้น ๆ เป็นไปได้โดยไม่มีปัญหา จนถึงเลิกการติดต่อเมื่อเสร็จงาน

Transport Layer ทำหน้าที่ควบคุมปริมาณ และรายละเอียดวิธีการรับส่งข้อมูล ให้เป็นไปตามกำหนดที่ตั้งไว้ และจัดการให้การเชื่อมโยงเครือข่ายเป็นไปด้วยความราบรื่น Transport Layer จะเป็นชั้นสุดท้าย ที่จัดการเรื่องเส้นทางในการส่งข้อมูล และจัดการตรวจสอบความผิดพลาดของข้อมูล ซึ่งส่วนของ TCP (Transmission Control Protocol) ในโพรโตคอล TCP/IP ทำงานที่ระดับนี้

Network Layer ทำหน้าที่ควบคุมวิธีการส่งผ่านข้อมูลระหว่างเครือข่ายให้ถูกต้อง และเป็นไปตามเส้นทางที่กำหนด โดยจะจัดการส่งผ่าน packet ข้อมูล ผ่านอุปกรณ์ต่าง ๆ ไปยังเครือข่ายย่อยได้อย่างถูกต้องตามที่ต้องการ นอกจากนี้ยังจัดการดูแลเส้นทางในการส่งข้อมูล (Routing table) และกั้นหรือกรอง packet ข้อมูลที่ส่งไปยังเครือข่ายเดียวกันไม่ให้ข้ามไปยังเครือข่ายอื่น ซึ่งจะช่วยลดปริมาณข้อมูลที่จะวิ่งบนเครือข่ายได้ส่วนหนึ่ง โพรโตคอล IP, TCP/IP และ IPx เป็นโพรโตคอลที่ทำงานอยู่ใน layer นี้

Data link Layer ทำหน้าที่เรียกใช้หรือกำหนดช่องทาง ในการส่งข้อมูลที่ถูกต้อง เช่น Ethernet, Tokenring หรือ FDDI เป็นต้น รวมถึงการลำดับและอัตราการรับส่งข้อมูลหรือ flow control และสถานที่ ที่จะส่งข้อมูลไป (address) ทั้งนี้ Data link layer จะเป็นชั้นแรกที่จัดการแปลงข้อมูลจาก bit ให้เป็น packet โดยจะมีการเพิ่มข้อมูลเพื่อตรวจสอบความถูกต้อง ในกรณีที่ส่งข้อมูลออกไป หรือในกรณีที่อ่านข้อมูลที่เข้ามา ก็จะตรวจสอบผ่าน checksum เพื่อดูว่าข้อมูลที่ได้รับมาถูกต้องครบถ้วน และถ้าได้รับ packet ข้อมูลที่ไม่ถูกต้องก็จะไม่เอาข้อมูลนั้นไปใช้งาน และจะบอกให้ต้นทางส่งข้อมูลเดิมมาใหม่

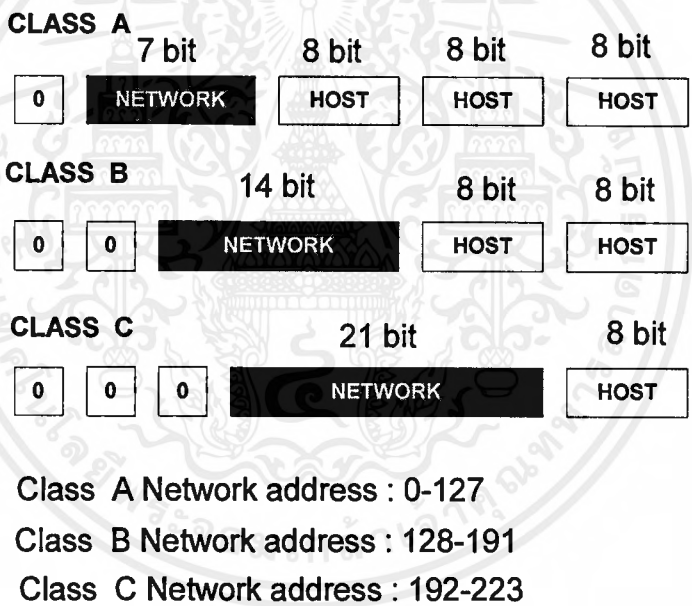
Physical Layer รับผิดชอบดูแลในรายละเอียดในการส่งข้อมูลในด้าน hardware เช่น การควบคุม Network Interface Card การส่งสัญญาณผ่านสายสัญญาณแบบต่าง ๆ การเชื่อมต่อเข้ากับเครือข่ายแบบต่าง ๆ โดยใช้ Physical Layer จะจัดสร้างสัญญาณทางไฟฟ้า, สัญญาณเสียง หรือ

IP เป็น internetwide protocol ซึ่งทำให้สอง transport protocol ที่ต่างสถานที่และต่าง ESs / hosts กันสามารถแลกเปลี่ยนหน่วยข้อมูล (NSDUS) กันได้ ซึ่งหมายถึงว่า หลาย ๆ network / subnet และ ISs / gateways ที่แตกต่างกันสามารถ ติดต่อสื่อสารกันได้อย่างสมบูรณ์

3.5 Internet IP

3.5.1 โครงสร้างแอดเดร(Address structure)

ในศัพท์ของ ISO เมื่อ 2 network ติดต่อกันด้วย host/ES ที่ต่อกับอินเตอร์เน็ต network เหล่านี้ ติดต่อกันได้โดยใช้ network service access point (NSAP) address และ subnet point of attachment (SNPA) สำหรับใน TCP/IP ก็จะมี IP address และ NPA address ตามลำดับ โดย NPA address จะแตกต่างกันในแต่ละชนิดของ network/subnet ขณะที่ IP address



รูปที่ 3.5 โครงสร้างของ Address ที่ใช้ใน class ต่าง ๆ ของเครือข่ายทั้งหมดความยาว 32 bit

IP address นี้มีการจัดแบ่งออกเป็นทั้งหมด 5 ระดับ (class) แต่ที่ใช้งานทั่วไปจะมีเพียง 3 ระดับคือ Class A, Class B และ Class C ซึ่งจะแบ่งตามขนาดความใหญ่ของเครือข่าย ถ้าเครือข่ายใดมีจำนวนเครื่องคอมพิวเตอร์ที่เชื่อมต่ออยู่มาก จะมีหมายเลขอยู่ใน Class A และลดหลั่นกันมาใน Class B และ Class C ตามลำดับ

จากรูปจะเห็นว่าหมายเลข IP ของ Class A มีตัวแรกเป็น 0 และหมายเลขของเครือข่าย (network number) ขนาด 7 bit และมีหมายเลขเครื่องคอมพิวเตอร์ (Host number) ขนาด 24 bit ทำให้ในหนึ่งเครือข่ายของ Class A สามารถมีคอมพิวเตอร์เชื่อมต่ออยู่ในเครือข่ายได้ถึง 2^{24} =16 ล้าน

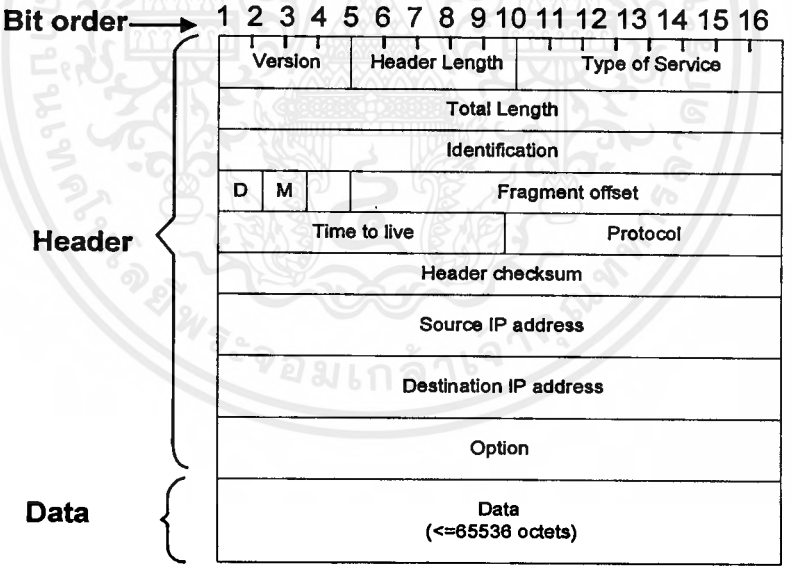
เครื่อง แต่ใน Class A นี้ จะมีหมายเลขเครือข่ายได้ 128 ตัวเท่านั้นทั่วโลก ซึ่งก็คือจะมีเครือข่ายใหญ่แบบนี้เพียง 128 เครือข่ายเท่านั้น

สำหรับ Class B จะมีหมายเลขเครือข่ายแบบ 14 bit และหมายเลขเครื่องคอมพิวเตอร์แบบ 16 bit (ส่วนอีก 2 bit ที่เหลือบังคับว่าต้องขึ้นต้นด้วย 10_2) ดังนั้นจึงสามารถมีคอมพิวเตอร์เชื่อมต่อในเครือข่าย Class B แต่ละเครือข่ายได้ถึง 2^{16} = กว่า 65,000 เครื่อง และสุดท้ายคือ Class C ซึ่งมีหมายเลขคอมพิวเตอร์แบบ 8 bit และมีหมายเลขเครือข่ายแบบ 21 bit ส่วน 3 bit แรกบังคับว่าต้องเป็น 110_2 ดังนั้นในแต่ละเครือข่าย Class C จะมีจำนวนเครื่องต่อเชื่อมได้เพียงไม่เกิน 254 เครื่องในแต่ละเครือข่าย (2^8=256 แต่หมายเลข 0 และ 255 จะไม่ถูกใช้งาน จึงเหลือเพียง 254)

จะเห็นได้ว่าเมื่อเครือข่ายและเครื่องคอมพิวเตอร์ ที่ต่ออยู่ในอินเตอร์เน็ตมีหมายเลข IP address ให้ใช้อ้างอิงได้ไม่ซ้ำกัน และมีความหมายให้ทราบถึงขนาดเครือข่ายแล้ว การติดต่อส่งผ่านข้อมูล จึงกระทำได้ไม่สับสน

3.5.2 รูปแบบของ ข้อมูล(Datagrams)

รูปแบบของ IP data unit ก็คือ datagrams ซึ่งโครงสร้างของ datagrams เป็นดังรูปที่ 3.6



รูปที่ 3.6 Internet datagrams format and contents

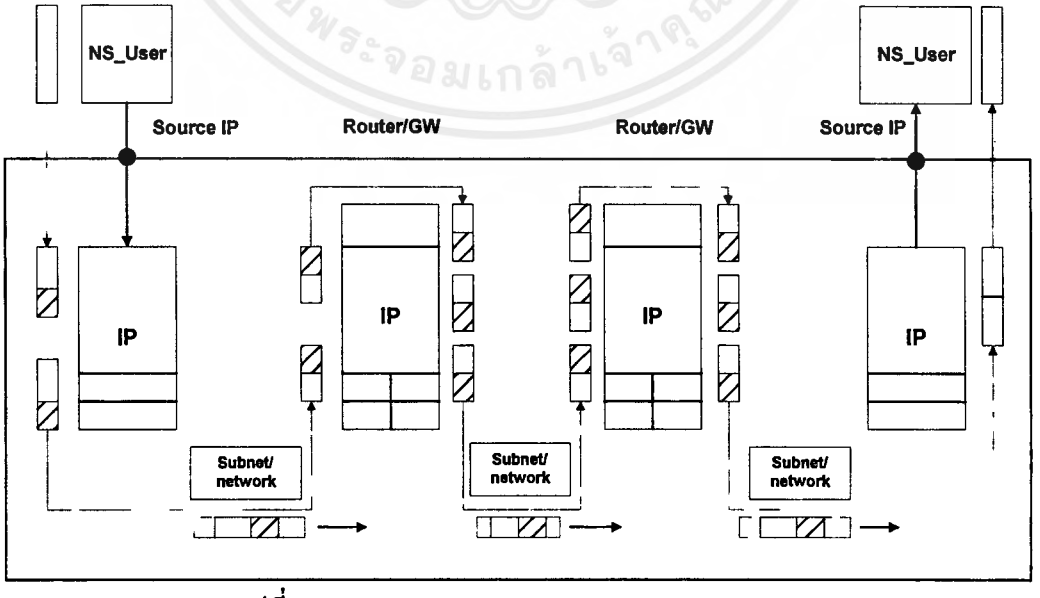
หน่วยข้อมูล IP (IP datagrams) แต่ละหน่วยจะประกอบด้วย ส่วนของข้อมูลที่รับมาจากส่วนของงาน TCP หรือ UDP และส่วนของข้อมูลนำทาง (Header) ซึ่งมีรายละเอียดดังนี้

- Version หมายเลขรุ่นของข้อกำหนด IP
- Header Length ความยาวของข้อมูลนำทาง

- Type of service วิธีการจัดการกับข้อมูล
- Total Length ความยาวของหน่วยข้อมูล
- Identification, Flags และ Fragment offset รายละเอียดที่เกี่ยวกับการแบ่งย่อยข้อมูล ซึ่งจะถูกนำมาใช้ในการรวบรวมข้อมูล
- Time to live เวลาสูงสุดที่ใช้ในการเดินทาง ซึ่งกำหนดมาจากต้นทาง เวลานี้จะลดลงเรื่อย ๆ ในระหว่างทาง ถ้าลดลงไปถึงศูนย์ หน่วยข้อมูลนั้นจะถูกกำจัดไป
- Protocol ชนิดของข้อมูลเป็น UDP หรือ TCP
- Header Checksum ค่าตรวจสอบข้อมูลนำทาง
- IP address หมายเลข internetwide IP (NSAP) ของเครื่องต้นทางและปลายทาง
- Option ข้อมูลอื่น ๆ เช่น ข้อมูลเกี่ยวกับการรักษาความปลอดภัย บันทึกเส้นทางเดินของข้อมูล และเวลาที่ข้อมูลเดินทางมาถึง เป็นต้น

3.5.3 การแบ่งส่วนย่อยของข้อมูล และการประกอบขึ้นใหม่ (Fragmentation and Reassembly)

ขนาดข้อมูลของผู้ใช้ซึ่งอ้างอิงกับ NSDU มีความจุได้ถึง 64k หรือ 65,536 bytes แต่ขนาดของหน่วยข้อมูล (packet size) ที่สามารถติดต่อกันในระบบที่ต่างกัน สามารถมีได้ตั้งแต่ 128 byte สำหรับระบบ X.25 packet switching จนถึง 8000 byte สำหรับบาง LAN ดังนั้นกระบวนการ Fragmentation และ Reassembly จึงถูกนำมาใช้เพื่อ ทำให้ขนาดของข้อมูลเล็กลง และสามารถส่งไปในระบบได้ และเมื่อถึงปลายทาง IP ก็จะทำการประกอบข้อมูล (reassembly) ขึ้นมาใหม่ก่อนที่จะส่งผ่านไปยังผู้ใช้ปลายทาง ดังรูปที่ 3.7



รูปที่ 3.7 internet fragmentation and reassembly

อันดับแรก IP ใน Host ต้นทางจะแยกข้อมูลของผู้ใช้ (NS -User), NSDU เป็น Datagram ซึ่งมี Address กำกับเป็นเฉพาะส่วน ๆ ไปซึ่งจะถูกออกคำสั่งโดย Network ที่มันติดต่ออยู่ด้วยและส่ง Datagram ไปยัง IP ใน Gateway ตัวแรก โดยที่ IP ใน Gateway จะไม่ Reassemble NSDU แต่จะปรับปรุงในขอบเขตที่เหมาะสม และส่ง Datagram ที่ได้รับตรงไปยัง Network ที่สอง (ถ้า Network ที่สองสามารถรองรับขนาด Datagram นี้) หรือทำการ Fragment datagram ให้มีขนาดเล็กลง ซึ่งขั้นตอนนี้จะถูกทำซ้ำที่ Gateway ตัวต่อไป โดยในรูปที่ 3.7 Network หลังสุดสามารถรองรับขนาดของ Packet ได้มากกว่า Packet ที่มันได้รับข้อมูลจึงถูกส่งได้โดยตรง โดยที่จะมีการปรับปรุงในบางส่วน Header ของ Datagram เท่านั้น จากนั้น IP ใน Host ปลายทางจะทำการประกอบข้อมูล (Reassemble) ที่มันได้รับขึ้นมาใหม่ และส่งผลที่ได้รับก็คือ NSDU ไปยังผู้ใช้ (NS-user)

ในการคิดค่าเวลาสูงสุดที่ Host ต้นทางกำหนดให้ Gateway รอ Datagram (NSDU) ระหว่างแต่ละการ Assembly ซึ่งก็คือ time-to-live ซึ่งจะถูกนำติดไปที่ส่วน Header ของ Datagram ซึ่งจะถูกตั้งค่าโดย IP ใน Host ต้นทาง ซึ่งจะมีค่าลดลงเรื่อยๆในแต่ละขั้นตอนของการ Process datagram ถ้า Datagram ถูก Fragment ค่าปัจจุบันจะถูกนำไปใส่ในส่วน Header ของ Datagram ตัวใหม่ ถ้ามันถึงค่า 0 ที่จุดใด ๆ ระหว่างการ Process ใน Gateway (หรือ Host) การ Reassembly ก็จะล้มเหลวและทุกๆการ Fragment ที่เกี่ยวกับ NSDU ก็จะถูกตัดทิ้ง

ค่า Time-to-live ในแต่ละ Datagram จะเป็นจำนวนเท่าของ 1 วินาที โดยที่จำนวนของมันจะถูกลดลงโดยแต่ละ IP ซึ่งจะเปลี่ยนแปลงไปตามค่าเวลาจริงในการส่งถ่ายข้อมูลของ Network ที่ติดต่อด้วย

3.5.4 Routing

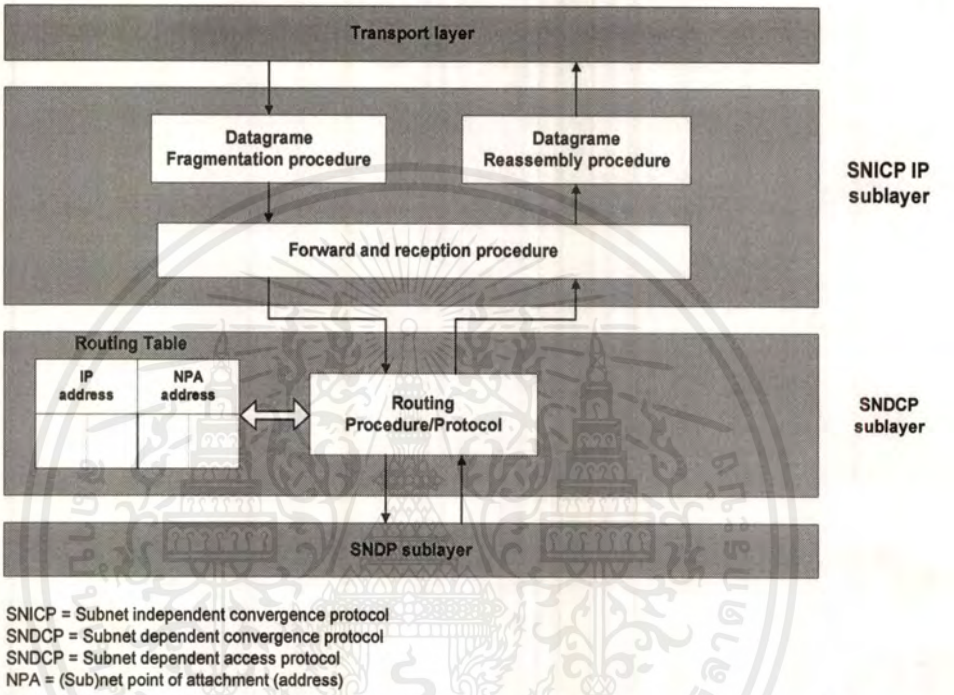
ในแต่ละ Network (หรือ subnet) ใน Internet จะมีชนิดของ PA address ที่แตกต่างกัน ซึ่งระบบ (system)-host หรือ gateway ที่ถูกต่อเข้ากับ network จะสามารถส่ง datagram ไปยังระบบอื่นได้โดยตรงเฉพาะ network ที่เหมือนกันเท่านั้น ในการเลือกเส้นทาง (routing) ให้ datagram ข้ามไปยังหลาย ๆ network IP ในแต่ละ internetwork gateway ต้องรู้ PA address ของ host ปลายทาง

ซึ่งมี 2 วิธีการพื้นฐานที่ถูกใช้ในการหาเส้นทางภายใน Internet คือ centralized และ distributed ด้วยวิธีการ centralized routing ข้อมูลเกี่ยวกับการเลือกเส้นทาง ที่เกี่ยวข้องกับแต่ละ gateway จะถูก download จาก site ส่วนกลางโดยใช้ข้อมูล network และ special network management โดย network management จะพยายามตรวจสอบ network และ host ที่ถูกเพิ่มเข้าและถอดออก และข้อบกพร่องที่จะถูกวินิจฉัยและตรวจซ่อม

ด้วยวิธีการ Distributed routing ทุก ๆ host และ gateway จะร่วมกันในการแบ่งปัน วิธีการในการรับประกันว่า ข้อมูลเกี่ยวกับการเลือกเส้นทางในแต่ละ system, host และ gateway จะถูกทำ

ให้ทันสมัย และสอดคล้องกัน ข้อมูลเกี่ยวกับการเลือกเส้นทางจะถูกจดจำไว้โดยแต่ละระบบ ในรูปของ routing table ซึ่งจะมี NPA address ไว้ใช้ในการส่งแต่ละ datagram ซึ่ง Internet จะใช้วิธีการแบบนี้

ขั้นตอนการ Routing ที่เกี่ยวกับ IP ขั้นตอนแรกจะอ่าน IP address (NSAP) ปลายทางจากภายใน datagram และใช้มันในการหาการตอบสนอง PA address ของ host หรือ gateway จาก



routing table ในส่วนที่เพิ่มเติมชุดของ routing protocol จะถูกใช้เพิ่มและรักษาส่วนที่อยู่ในแต่ละ routing table ในแบบของ distributed ซึ่งรูปแบบทั่วไปที่ถูกใช้ภายใน host IP แสดงดังรูปที่ 3.8

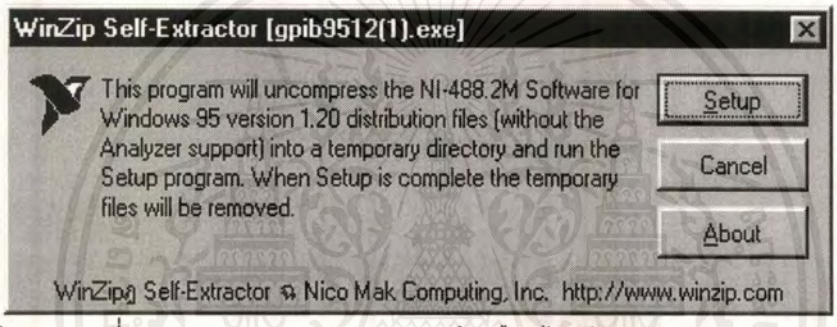
รูปที่ 3.8 รูปแบบทั่วไปของการเลือกเส้นทางภายใน Host

บทที่ 4
การติดตั้งการ์ด GPIB

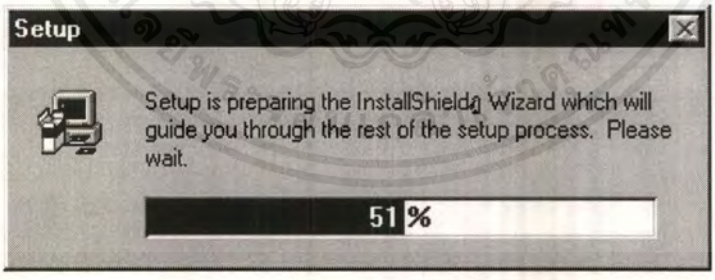
Project ในส่วนนี้จะใช้การ์ดของ National ซึ่งเป็นรุ่น GPIB-PCII/A การเซตการ์ดรุ่นนี้จำเป็นที่จะต้องไม่เสียบการ์ดกับเครื่องคอมพิวเตอร์ของเราซึ่งเป็นกฎเกณฑ์ของทาง National และการเซตการ์ด GPIB มีขั้นตอนดังนี้

1. ส่วนของการ Install Program Software

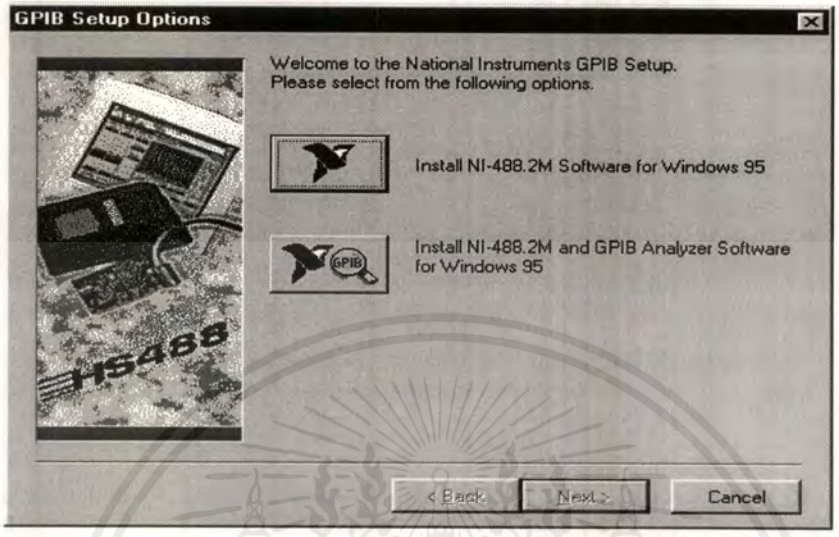
1.1 เลือก Start -> Setting -> Control Panel



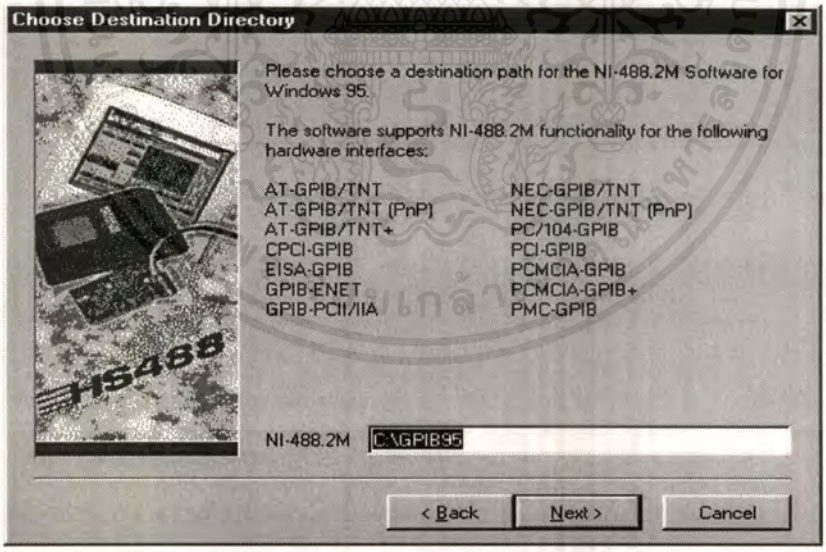
1.2 แล้ว Click ที่ Add/Remove Programs แล้วเลือกไฟล์ Install Program ของ GPIB จะได้ดังรูปที่ 1



1.3 ให้เลือก Setup โปรแกรมจะทำการ Setup ดังรูป



1.4 เมื่อ Setup เสร็จแล้วจะปรากฏหน้าต่าง GPIB Setup Option ดังรูปให้เลือก Install NI-488.2M Software for Windows 95 แล้ว Click Next



1.5 หลังจากนั้นจะปรากฏหน้าต่าง Choose Destination Directory ให้ Click Next

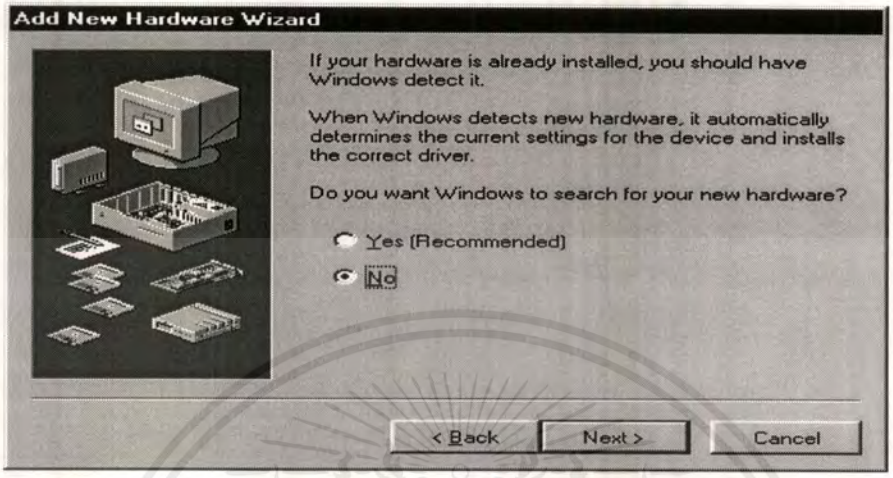
1.6 โปรแกรมจะทำการ Setup ให้เสร็จแล้ว Click Next จนจบ

1.7 ให้ทำการ Restart

Note: ในขั้นตอนการ Install Program นั้นเราจะไม่เสียบการ์ด GPIB ที่เครื่อง Computer

2. ส่วนการ Install New Hardware

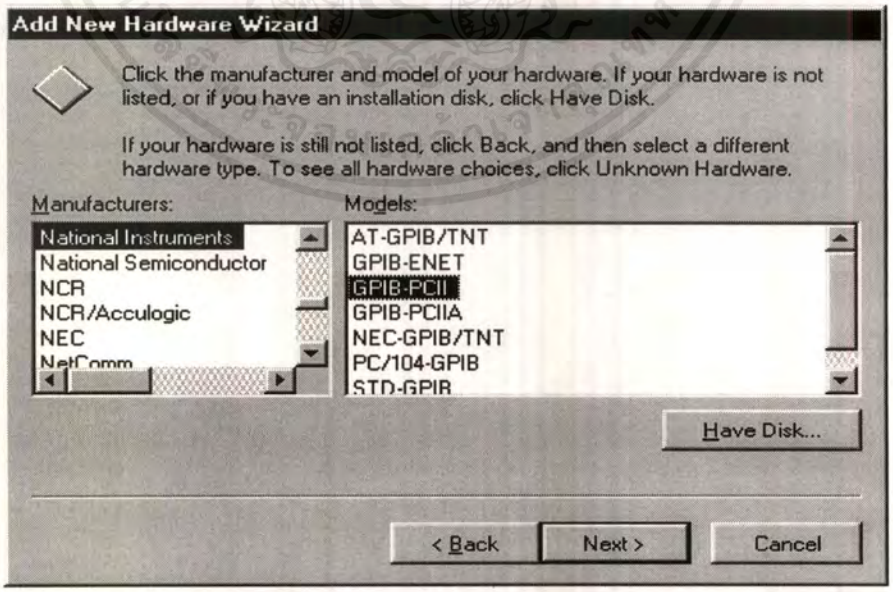
2.1 เลือก Start -> Setting -> Control Panel



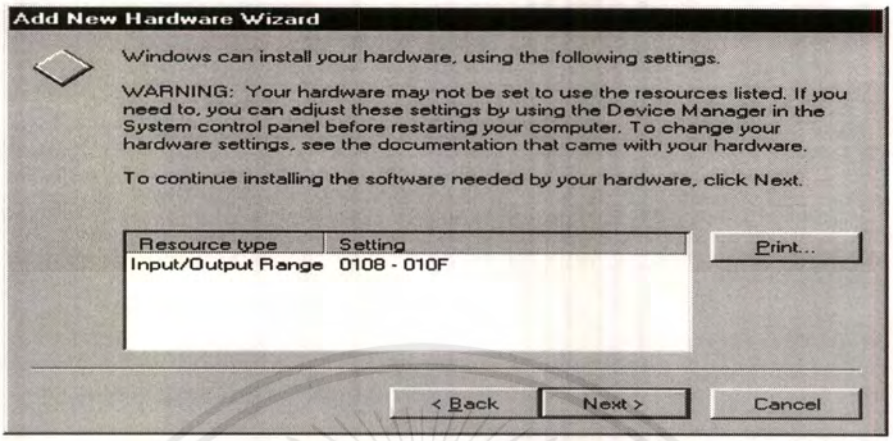
2.2 เลือก Add New Hardware จะปรากฏหน้าต่างดังรูปให้เลือก NO แล้ว Click Next

2.3 ให้เลือกไปที่ National Instruments GPIB Interface จะได้ดังรูป

2.4 แต่ถ้าไม่มีส่วนที่ 2.3 ปรากฏให้เลือกที่ Other Deivce แล้วจะปรากฏหน้าต่าง Add New Hardware Wizard ให้เลือกไปที่ National Intrument แล้วให้เลือก GPIB-PCII ในส่วนของ Model ให้ Click Next ดังรูป



2.5 คอมพิวเตอร์จะทำการตรวจสอบระบบ Hardware ของเครื่องเราและจะให้รายละเอียดดังรูป แล้วเอารายละเอียดไปเซตกับการ์ดของเรา



2.6 ให้ปิดเครื่องคอมพิวเตอร์ของเรา

2.7 ให้ทำการเซตการ์ด GPIB ตามข้อมูลที่ได้ในข้อ 2.5

2.8 เสียบการ์ด GPIB ที่เครื่องคอมพิวเตอร์

บทที่ 5
การออกแบบและสร้าง

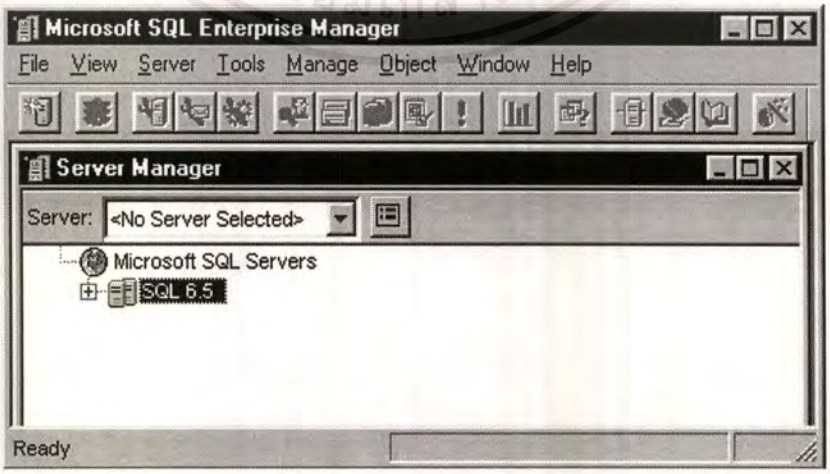
5.1 การเขียนโปรแกรมเพื่อควบคุมเครื่องมือวัด

ในโครงงานนี้ได้เขียนโปรแกรม เป็นแบบจำลองของทั้ง 4 เครื่องเพื่อง่ายและสะดวกต่อการใช้งาน ตัวโปรแกรมควบคุมเครื่องมือวัด ได้แบ่งออกเป็น 2 ส่วนคือ โปรแกรมในส่วนของผู้ใช้ (user) ที่มีหน้าที่คือ รับค่าและฟังก์ชันต่าง ๆ ที่ผู้ใช้ต้องการส่งไปยัง MS SQL Server และส่วนที่สองคือ ส่วนโปรแกรมที่อยู่ที่เครื่องคอมพิวเตอร์ ที่เป็นตัวควบคุมชุดเครื่องมือวัด GPIB ซึ่งมีหน้าที่รับค่าจาก ดาต้าเบสใน MS SQL Server มาประมวลผลและสั่งงาน ชุดเครื่องมือวัด และรับค่าที่ได้จากเครื่องมือวัด ส่งกลับไปยังดาต้าเบส เช่นในกรณีของมัลติมิเตอร์ เนื่องจากการทำงานของโปรแกรมทั้งทางด้านผู้ใช้ และเครื่องคอมพิวเตอร์ที่ควบคุมชุด GPIB จำเป็นต้องทำงานร่วมกับดาต้าเบส ในการเขียนโปรแกรม จึงต้องมีการสร้างตารางมาตรฐานข้อมูล เพื่อเก็บข้อมูลต่าง ๆ ซึ่งเราจะใช้โปรแกรม MS SQL 6.5 เป็นตัวจัดการเกี่ยวกับฐานข้อมูล

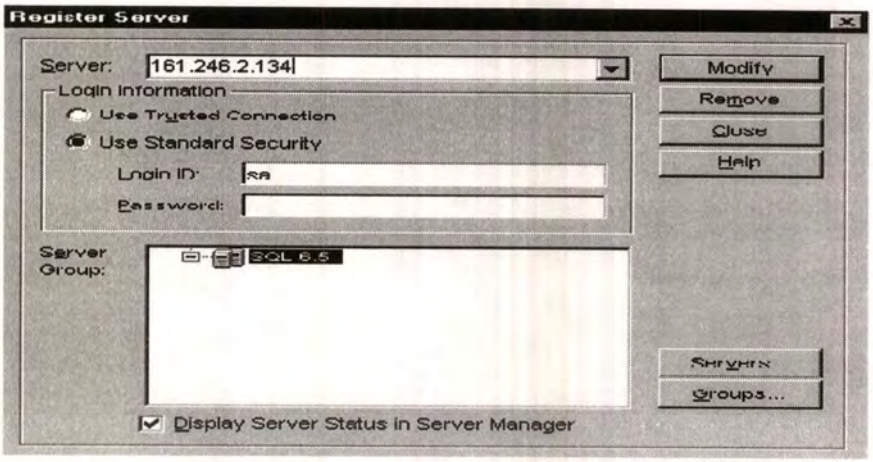
การสร้างตารางฐานข้อมูล

สำหรับการสร้างตาราง (table) ฐานข้อมูล (database) สามารถทำได้โดยใช้ Microsoft SQL Enterprise Manager ใน MS SQL 6.5 ดังนี้

1. ทำการติดตั้งตัว Sever ฐานข้อมูลที่เราต้องการใช้ซึ่งใน MS SQL Sever ซึ่งในที่นี้คือ หมายเลข IP 161.246.2.134 โดยกำหนด login ID เป็น SA ดังรูปที่ 5.1และ รูปที่ 5.2 ตามลำดับ



รูปที่ 5.1 แสดงสถานะเริ่มต้นของ SQL Enterprise

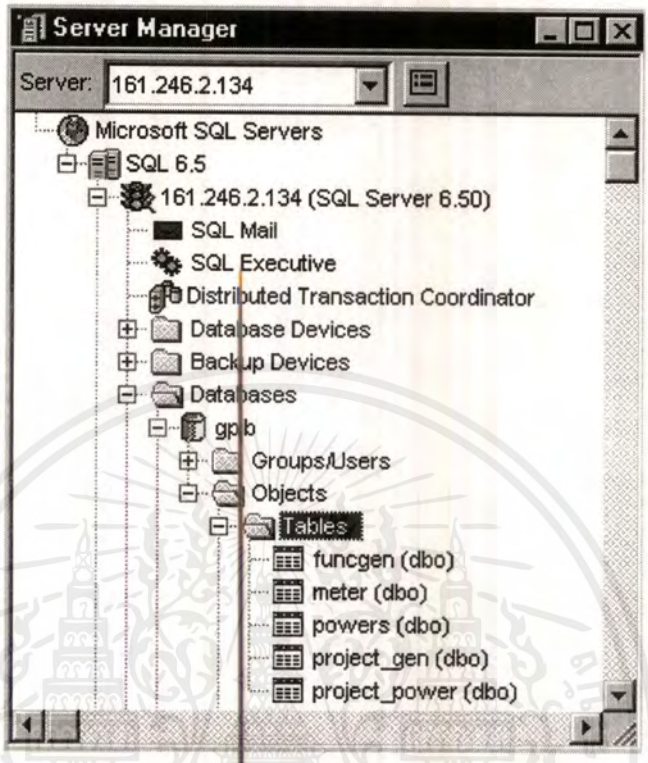


รูปที่ 5.2 แสดงการติดตั้ง Sever ลงในโปรแกรม

2. ทำการสร้างฐานข้อมูลโดยสร้างในส่วนของฐานข้อมูลใหม่ ที่ New Database ดังรูปที่ 5.3 โดยได้ทำการสร้างดาต้าเบสชื่อ gpib

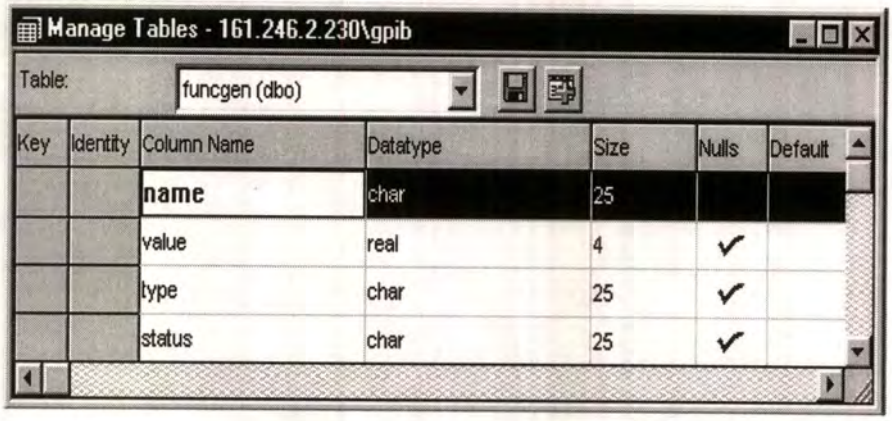
รูปที่ 5.3 แสดงการสร้างฐานข้อมูลใหม่

3. ทำการสร้างตาราง (table) เพื่อเก็บข้อมูลที่เราต้องการ โดยจะอยู่ในส่วน ของ object ดังรูปที่ 5.4



รูปที่ 5.4 แสดงการสร้างตารางเพื่อเก็บข้อมูลใหม่

4. ทำการกำหนดรูปแบบ ข้อมูลที่ต้องการใช้ซึ่งในที่นี้ได้กำหนด column ของข้อมูลไว้ 4 ชนิด คือ Name, Value, Type และ Status ซึ่งเป็นตัวอย่างตารางของ Function Generator โดยส่วนที่ให้เช็คว่าเป็น Nulls ได้หรือไม่ คือส่วนที่จะกำหนดว่า สามารถให้มีข้อมูลว่างได้หรือไม่ ดังรูปที่ 5.5



รูปที่ 5.5 แสดงการกำหนดรูปแบบข้อมูลใน table ใหม่

นอกจากนี้เรายังสามารถจัดการ เกี่ยวกับขนาดของดาต้าเบสที่เราต้องการ จองพื้นที่ของหน่วยความจำไว้ โดย set ค่าใน Manage Database ซึ่งขนาดของ ดาต้าเบส ขึ้นอยู่กับจำนวนข้อมูลที่ต้องใช้

5.2 โปรแกรมควบคุม Programable Power Supply

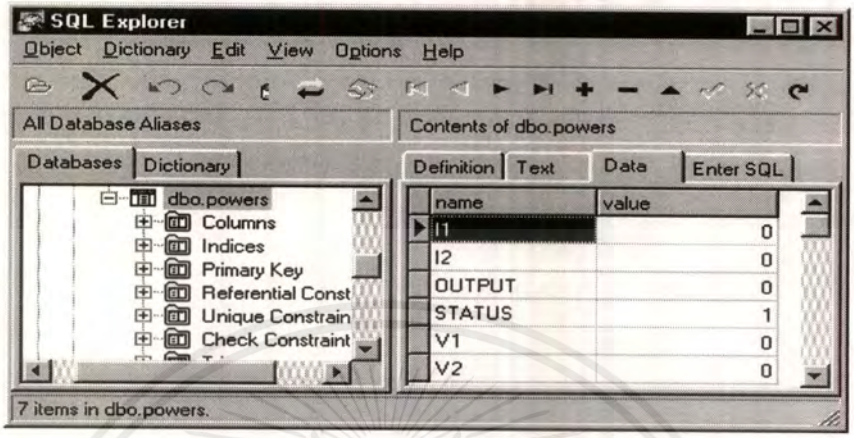
โปรมแกรมในส่วนของ Programable Power Supply ประกอบด้วยโปรแกรม User Power Supply ซึ่งเป็นส่วนตัวโปรแกรมที่ผู้ใช้ และ Master Power Supply เป็นโปรแกรมที่อยู่ที่เครื่องคอมพิวเตอร์ที่เป็นตัวอินเตอร์เฟซ กับ Power Suppiy (Hm 8142)

5.2.1 โปรแกรม User Power Supply

โปรแกรม User Power Supply มีหน้าที่การทำงานคือ รับค่าจากฟังค์ชั่นจากผู้ใช้ และนำค่าที่ได้ส่งไปให้กับ ดาต้าเบส โดยค่าที่มีการเปลี่ยนแปลงของตัว Power Supply ประกอบด้วย

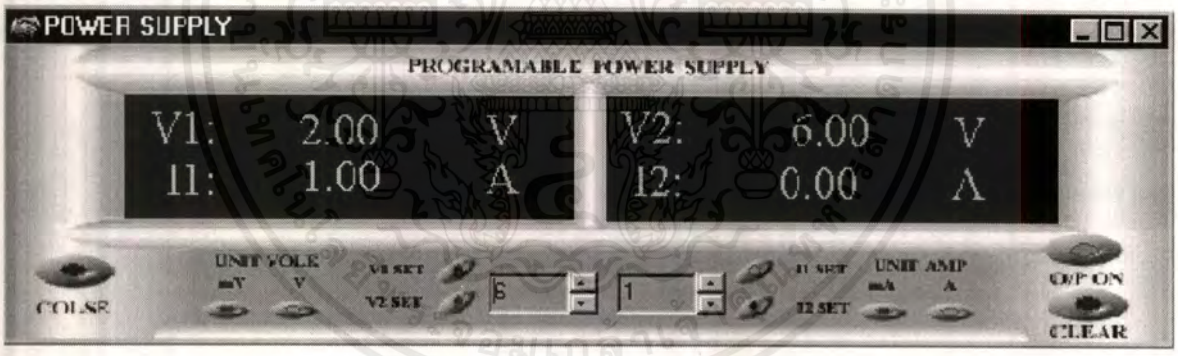
1. ค่าแรงดัน V1
2. ค่าแรงดัน V2
3. ค่ากระแส I1
4. ค่ากระแส I2
5. สถานะทาง Output ของเครื่อง (On / Off)
6. สถานะเคลียร์ (ถ้าอยู่ในสถานะเคลียร์จะ Set ทุกค่าคือ V1 , V2, I1, และ I2 มีค่าเท่ากับศูนย์

โดยเราต้อง นำค่าเหล่านี้ ไปใส่ไว้ใน ฐานข้อมูลซึ่งได้สร้างตารางชื่อ Dbo.Powers ใน ดาต้าเบสชื่อ Gpib ดังรูปที่ 5.7



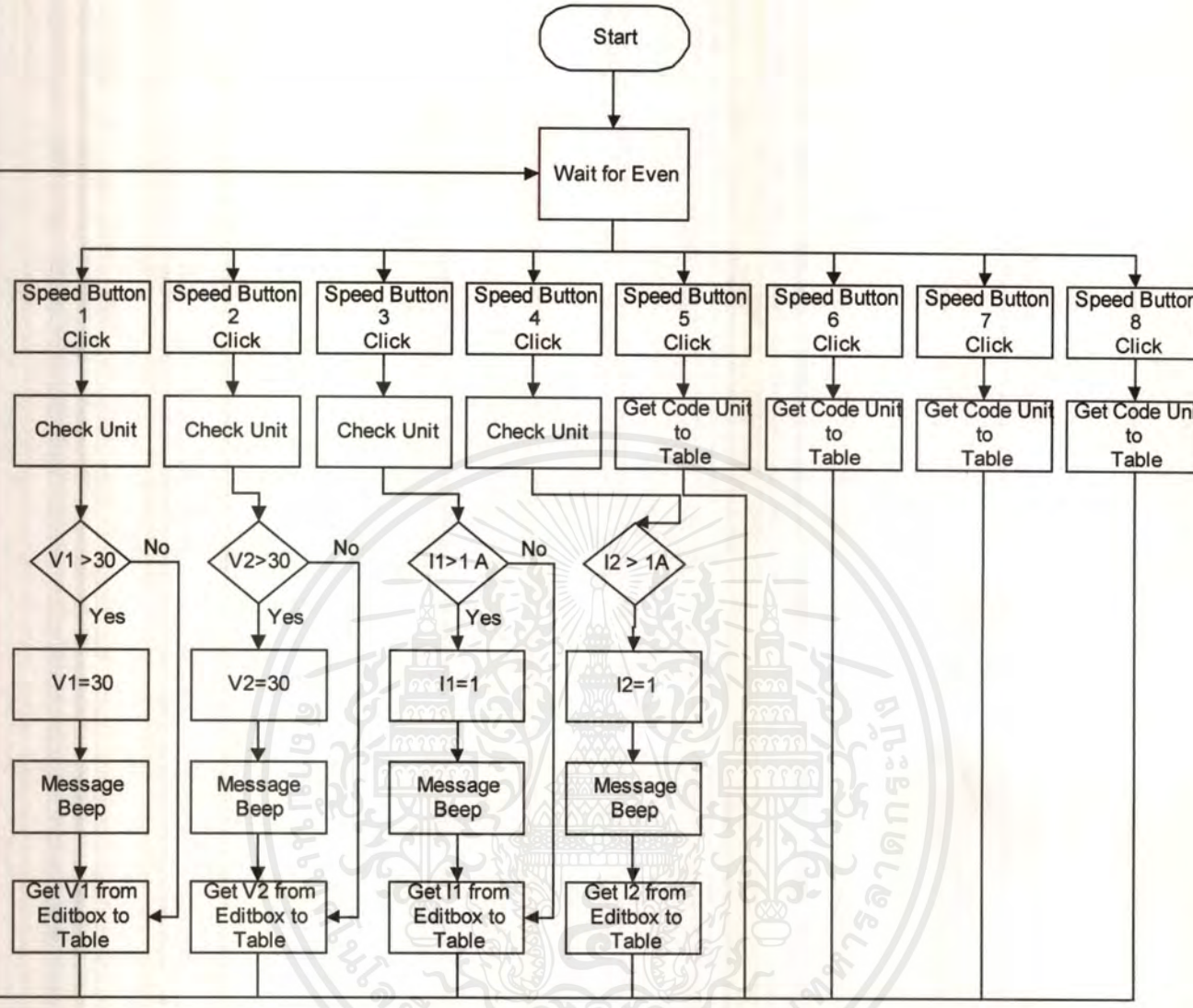
รูปที่ 5.7 แสดงข้อมูลของ Hm 8142 ในตาราง Dbo.Powers

โดยรูปร่างของแบบจำลองที่ส่วนของ User แสดงไว้ในรูปที่ 5.8



รูปที่ 5.8 แบบจำลองของ Programable Power Supply

สำหรับการทำงานของโปรแกรม แสดงใน ผังการทำงาน โปรแกรมดังรูปที่ 5.9 ซึ่ง จะเป็นลักษณะการทำงานตามเหตุการณ์ (Event) คือ จะทำงานเฉพาะตอนผู้ใช้ กระทำกับส่วนประกอบต่าง ๆ โดยเป็นอิสระไม่ขึ้นกับตัวอื่น ๆ โดยจะมีการจำกัดค่าของ แรงดัน และ กระแสไว้ที่ 30 V และ 1A ตามคุณสมบัติของเครื่อง

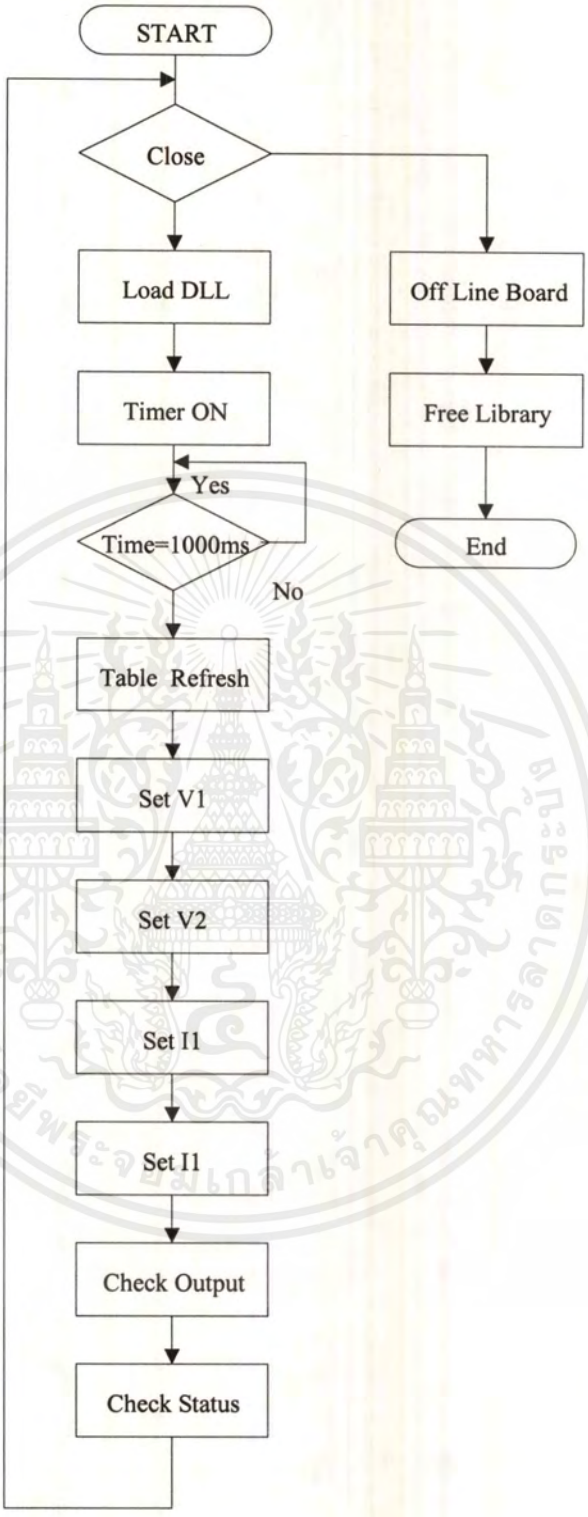


รูปที่ 5.9 ผังการทำงานของ User Power Supply

5.2.2 โปรแกรม Master Programable Power Supply

โปรแกรมส่วนนี้จะมีหน้าที่ รับค่าจาก ตาราง dbo.powers ในฐานข้อมูล มาสั่งงานตัวpower supply โดยจะนำค่าจากตารางมาสั่งงานทุกๆ 1 วินาที เพื่อให้การทำงานของpower supply มีการตอบสนอง ทันต่อข้อมูลใหม่ๆจากผู้ใช้

ในการอินเตอร์เฟซ กับ ตัว power supply นั้น เนื่องจากตัว บอร์ดของ GPIB จะมี คำสั่งเฉพาะของบอร์ดที่สนับสนุน เฉพาะภาษา C และ BASIC ดังนั้นจึงต้องมีการเรียกใช้ ไลบรารี่ของ GPIB 32 แล้ว นำมาแปลงเป็นฟังก์ชั่นในรูปTObject ซึ่งเป็นรูปแบบของโปรแกรมภาษา Delphi การทำงานของโปรแกรมแสดงดังรูปที่ 5.10



รูปที่ 5.10 ผังการทำงานของ Master Programable Power Supply

5.3 โปรแกรมควบคุม Programable Function Generator

โปรแกรมในส่วนของ Programable Function Generator ประกอบด้วยโปรแกรม User Function Generatorซึ่งเป็นโปรแกรมในส่วนของผู้ใช้ และ Master Function Generator เป็นโปรแกรมที่อยู่ที่เครื่องคอมพิวเตอร์ ที่เป็นตัวอินเตอร์เฟซกับ Function Generator HM 8130

5.3.1 โปรแกรม User Function Generator

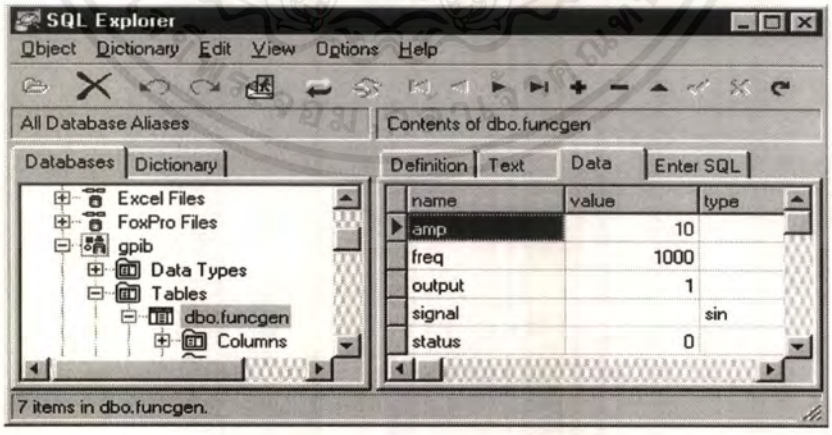
ตัวโปรแกรม User Function Generator มีหน้าที่การทำงาน คือ รับค่าและฟังชั่นก์จากผู้ใช้ และนำค่าที่ได้ส่งไปยัง ดาต้าเบส โดยค่าที่มีการเปลี่ยนแปลงของ Function Generator ประกอบด้วย

1. Freq: ค่าความถี่ของสัญญาณ
2. Amp: ค่าแอมปลิจูด
3. Signal: ชนิดของสัญญาณ
4. Output: สถานะทาง Output กำหนดให้มีสองค่าคือศูนย์และหนึ่ง

   0 = Output on

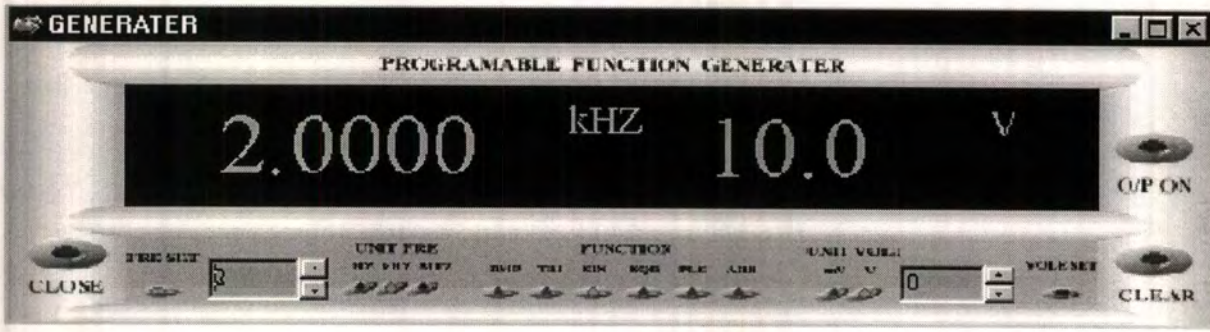
   1 = Output off
5. Status: สถานะเคลียร์ (ถ้าอยู่ในสถานะเคลียร์จะ เซตให้มีค่าความถี่ 1 KHZ ค่าแอมปริจูด 10 V สัญญาณชนิด Sine)

ค่าเหล่านี้ที่ User กำหนดจะถูกนำไปใส่ไว้ที่ดาต้าเบส ซึ่งได้สร้างตารางชื่อ Dbo. Funcgen ใน ดาต้าเบส ชื่อ Gpib ดังรูปที่ 5.11



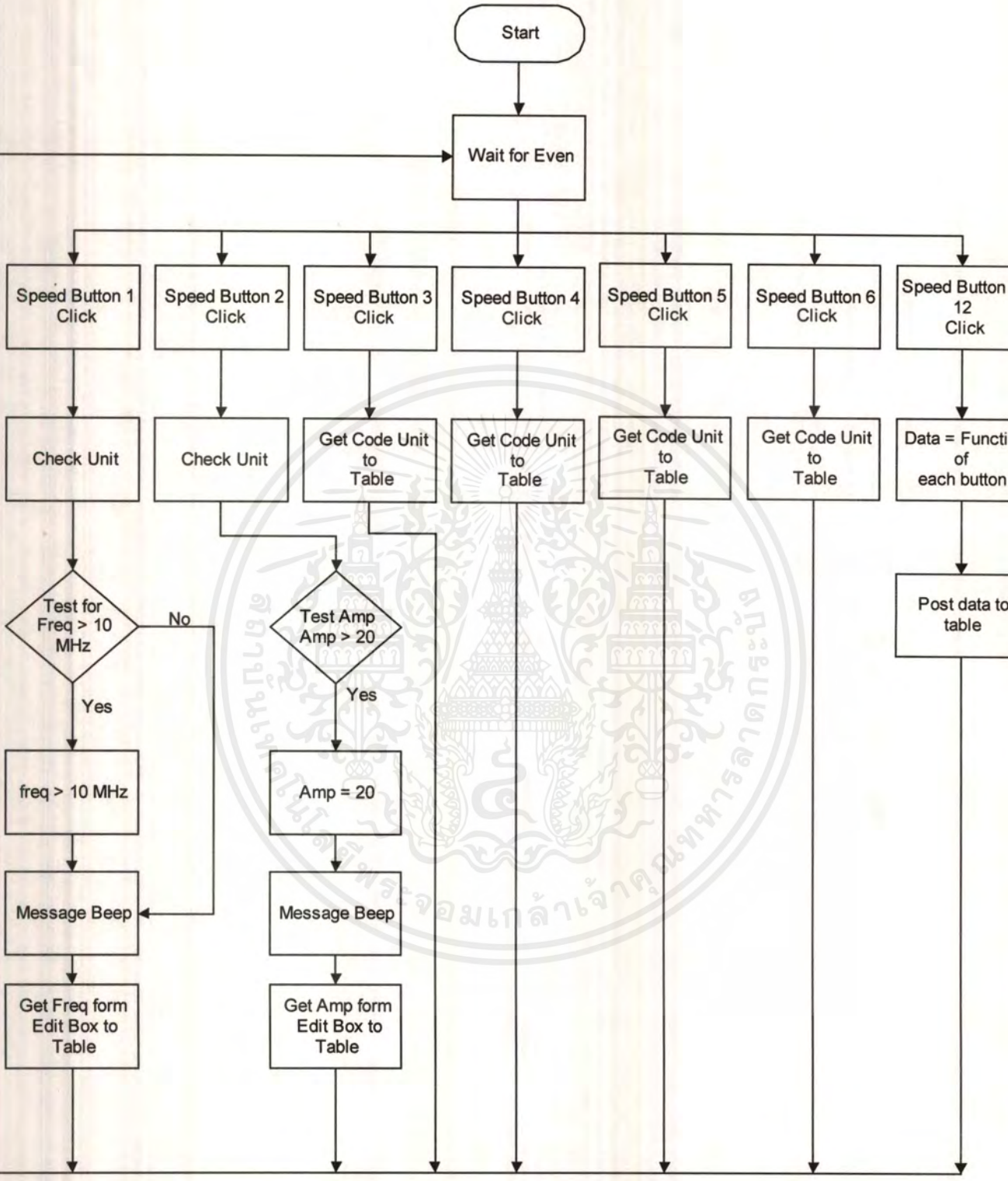
รูปที่ 5.11 แสดงข้อมูลในตาราง Dbo . Funcgen

รูปร่างของแบบจำลองหน้าปิดของ Function Generator แสดงไว้ในรูปที่ 5.12



รูปที่ 5.12 แบบจำลองของ Programable Function Generator

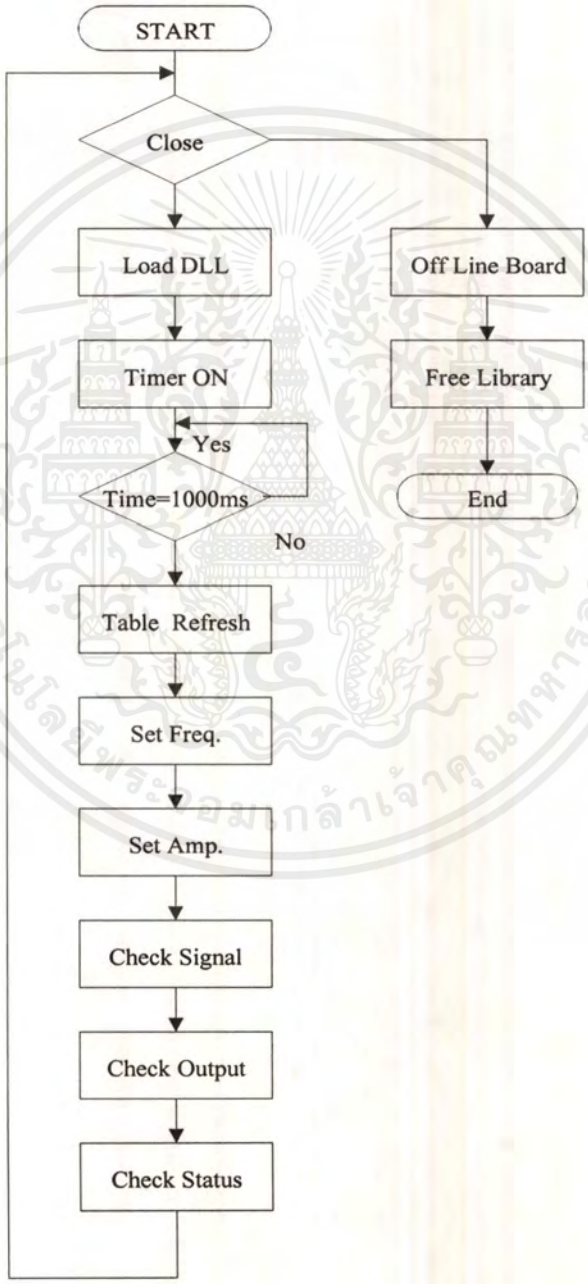
สำหรับการทำงานของ โปรแกรมมีลักษณะการทำงานคล้ายกับ Power Supply คือ ทำงานตาม Event และส่งค่าไปยังดาต้าเบส แตกต่างกันที่ชนิดของข้อมูล ซึ่งการทำงานของโปรแกรม แสดงตัวผังการทำงาน ในรูป 5.13



รูปที่ 5.13ผังการทำงานของ User Programable Function Generator

5.3.2 โปรแกรม Master Function Generator

การทำงานจะรับค่าจากตาราง Dbo. Funcgen ในฐานข้อมาสั่งงาน Function Generator ให้ทำงาน โดยจะมีการ อัพเดทข้อมูลทุกๆ 1วินาที เพื่อให้ทันต่อการเปลี่ยนแปลงของผู้ใช้ ซึ่งขั้นตอนแรกก็ต้องโหลดฟังก์ชั่นการทำงานของ ไลบรารี่ของ GPIB มาใช้ ซึ่งรายละเอียดของแต่ละคำสั่งดูได้จากคู่มือของเครื่อง ซึ่งจะไม่เหมือนกันในแต่ละเครื่อง การทำงานของโปรแกรม แสดงในรูปที่ 5.14



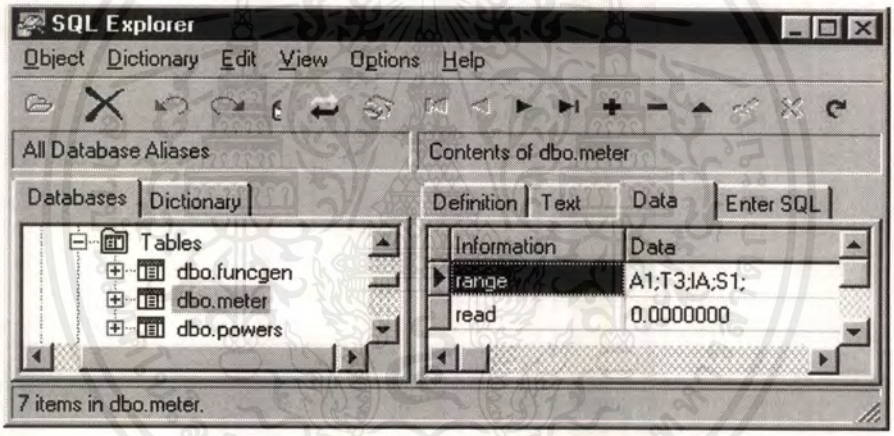
รูปที่ 5.14 ผังการทำงานของ Master Programable Function Generator

5.4 โปรแกรมควบคุม Programable Multimeter

โปรแกรมในส่วนของ Programable Multimeterประกอบด้วย โปรแกรม User Multimeter ซึ่งเป็นโปรแกรมด้านผู้ใช้และ Master Multimeter เป็นโปรแกรมที่อยู่ที่เครื่องคอมพิวเตอร์ที่เป็นตัวอินเตอร์เฟซ กับ Multimeter (HM 8112-2) ดังนี้

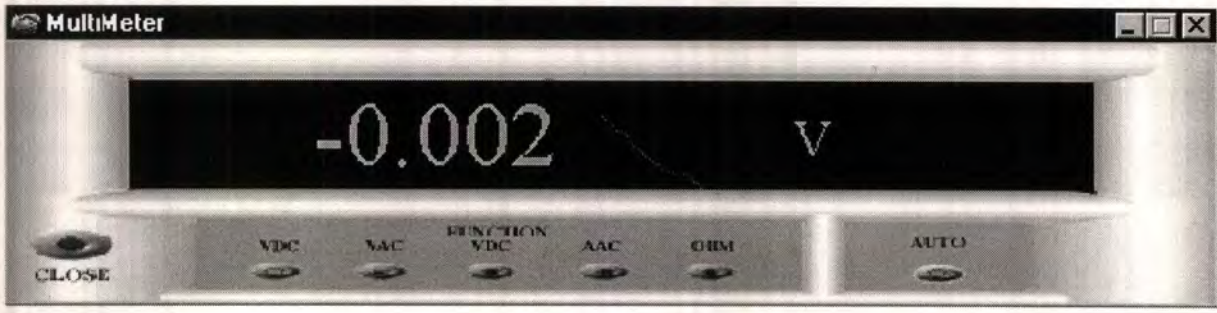
5.4.1 โปรแกรม User Multimeter

ตัวโปรแกรม User Multimeterมีลักษณะการทำงานแตกต่างไปจาก อุปกรณ์สองตัวที่กล่าวมา คือ มีหน้าที่ส่งค่า ฟังก์ชั่นที่ใช้กำหนด ไปยังดาต้าเบส และนำข้อมูลที่รับมาจากดาต้าเบส ซึ่งเป็นข้อมูลที่ตอบสนองมาจาก Multimeter มาแสดงผลที่หน้าจอของผู้ใช้โดยข้อมูลต่าง ๆของ Multimeter จะประกอบด้วยสองส่วน คือ ข้อมูลคำสั่งและข้อมูลที่อ่านมาจาก Multimeter ดังแสดงในรูป 5.15 ซึ่งเป็นข้อมูล ภายในตาราง Dbo. Meter



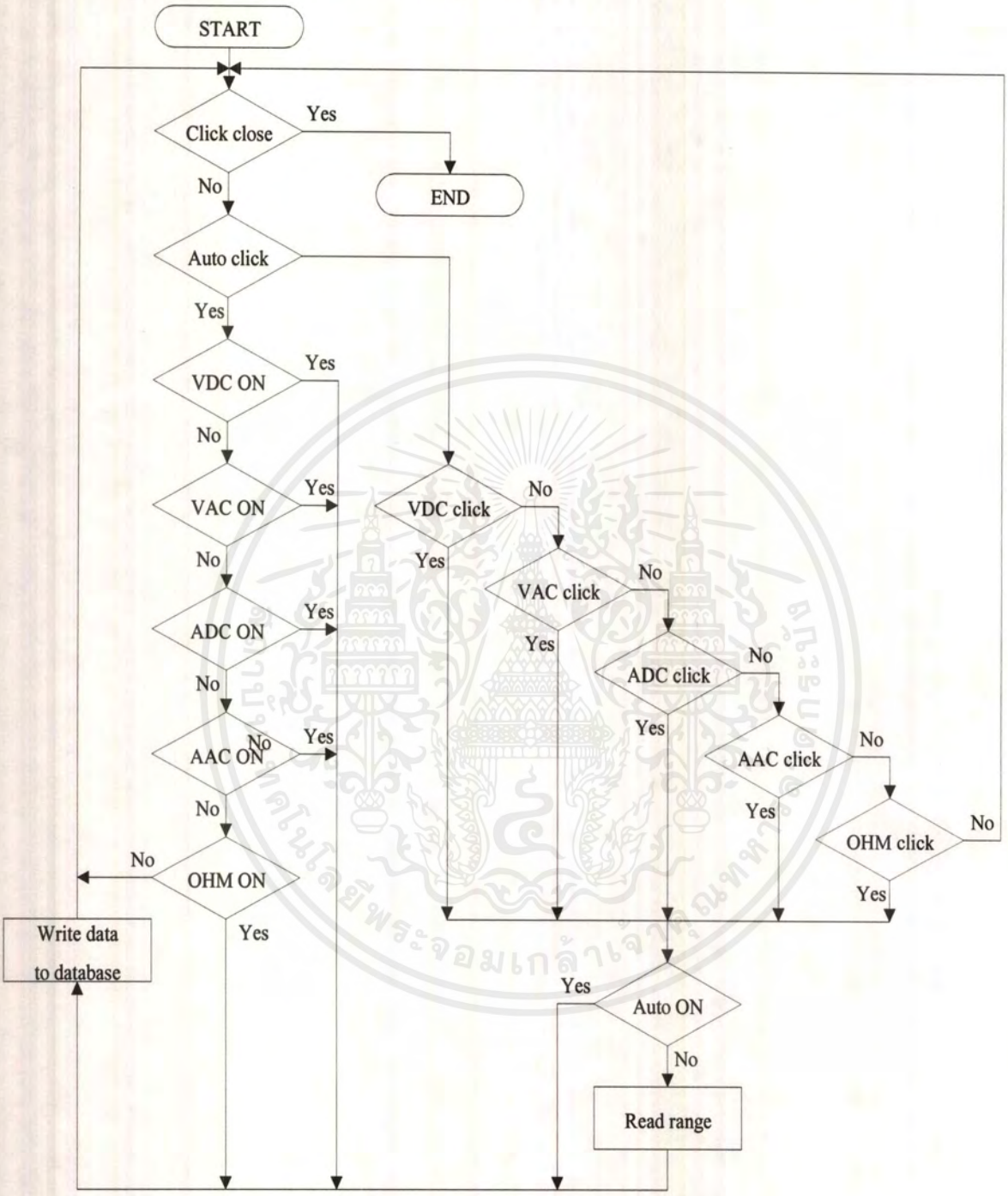
รูปที่ 5.15 ข้อมูล ในตาราง Dbo. Meter

โดยรูปร่างของแบบจำลองหน้าปัดของ Multimeterแสดงไว้ในรูปที่ 5.16



รูปที่ 5.16 แบบจำลองของ Programable Multimeter

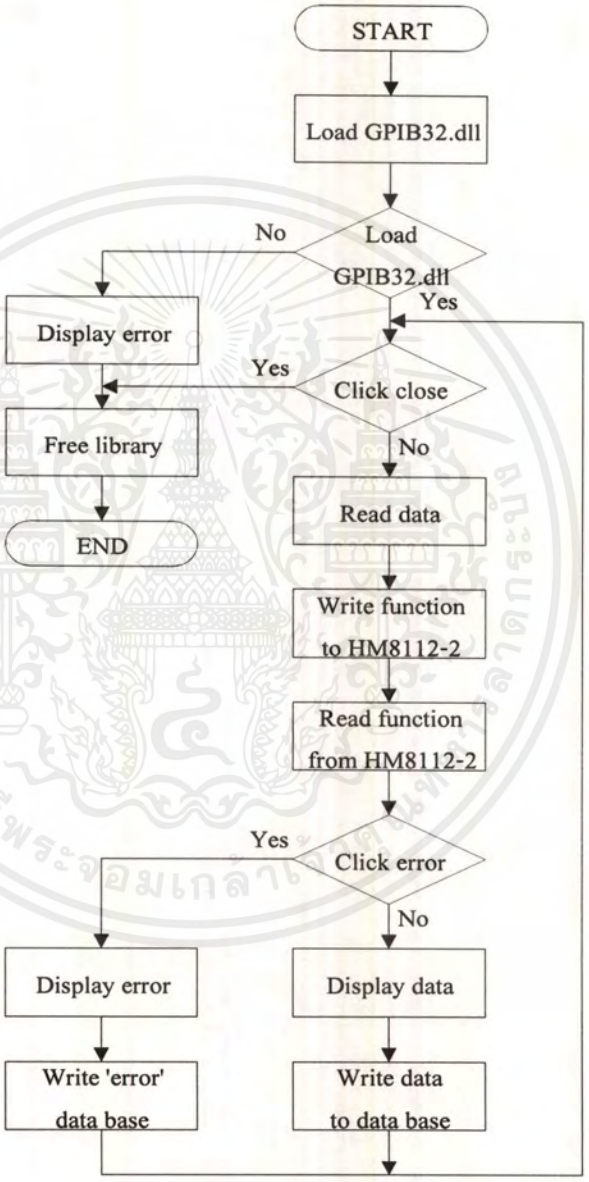
สำหรับการทำงานของโปรแกรมแสดง ดังรูปที่ 5.17



รูปที่ 5.17 ผังการทำงานของ โปรแกรม User Multimeter

5.4.2 โปรแกรม Master Programable Multimeter

โปรแกรมส่วนนี้มีหน้าที่รับคำสั่ง จากตาราง Dbo.Meter ในฐานข้อมูลมาสั่งงานตัว Multimeter และอ่านข้อมูลจากบัฟเฟอร์ ของ Multimeter ส่งไปให้ฐานข้อมูล การทำงานของโปรแกรมแสดง ดังรูปที่ 5.18



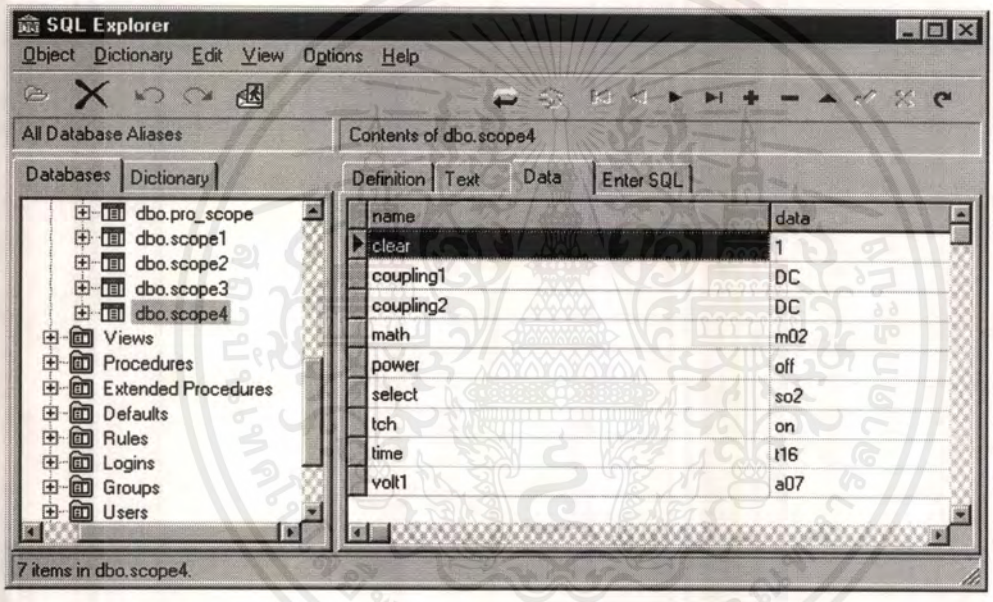
รูปที่ 5.18 ผังการทำงานของ Master Multimeter

5.5 โปรแกรมควบคุม programmable Oscilloscope

โปรแกรมในส่วนของ Programmable Oscilloscope ประกอบด้วย User Oscilloscope ซึ่งเป็นโปรแกรมด้านผู้ใช้ และ Master Programmable Oscilloscope ซึ่งเป็นโปรแกรมที่อยู่ที่เครื่องคอมพิวเตอร์ที่เป็นตัวอินเตอร์เฟซกับ Programmable Oscilloscope

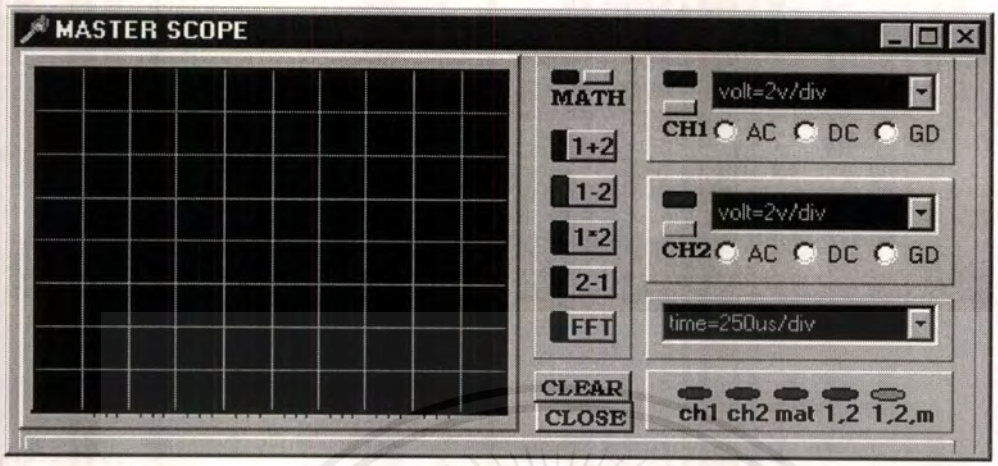
5.5.1 โปรแกรม User Osilloscope

โปรแกรม User Oscilloscope มีหน้าที่ส่งค่าฟังก์ชั่นสั่งงาน Oscilloscope ไปยังดาต้าเบส และรับข้อมูลจากดาต้าเบสซึ่งเป็นข้อมูลที่ตอบสนองมาจาก Osilloscope มาแสดงผลรูปสัญญาณที่หน้าจอของผู้ใช้



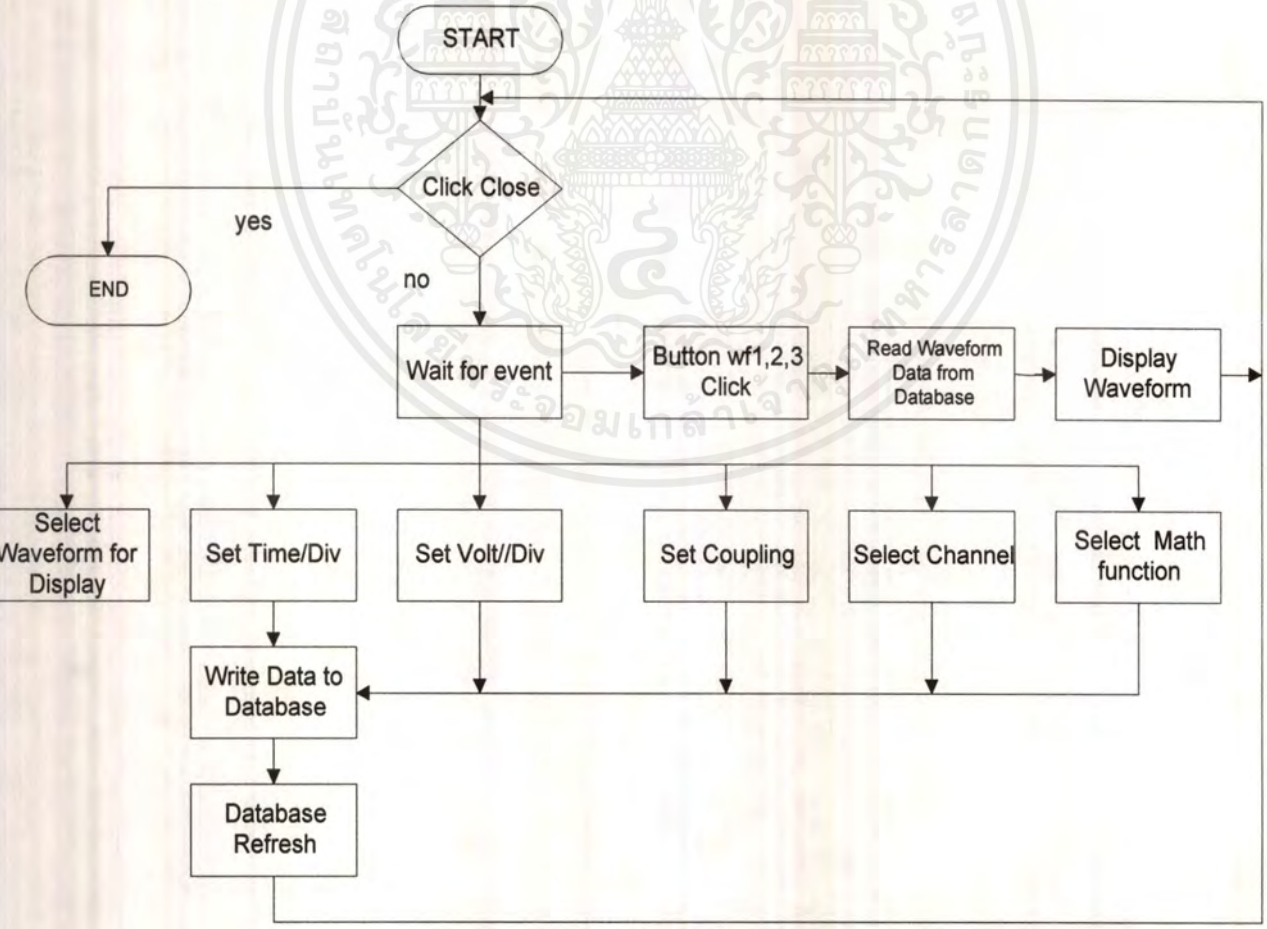
รูปที่ 5.19 ข้อมูลในตาราง dbo.Osilloscope4

รูปร่างของแบบจำลองหน้าปัดของ Programmable Oscilloscope แสดงในรูปที่ 5.20



รูปที่ 5.20 แบบจำลองของ Programmable Oscilloscope

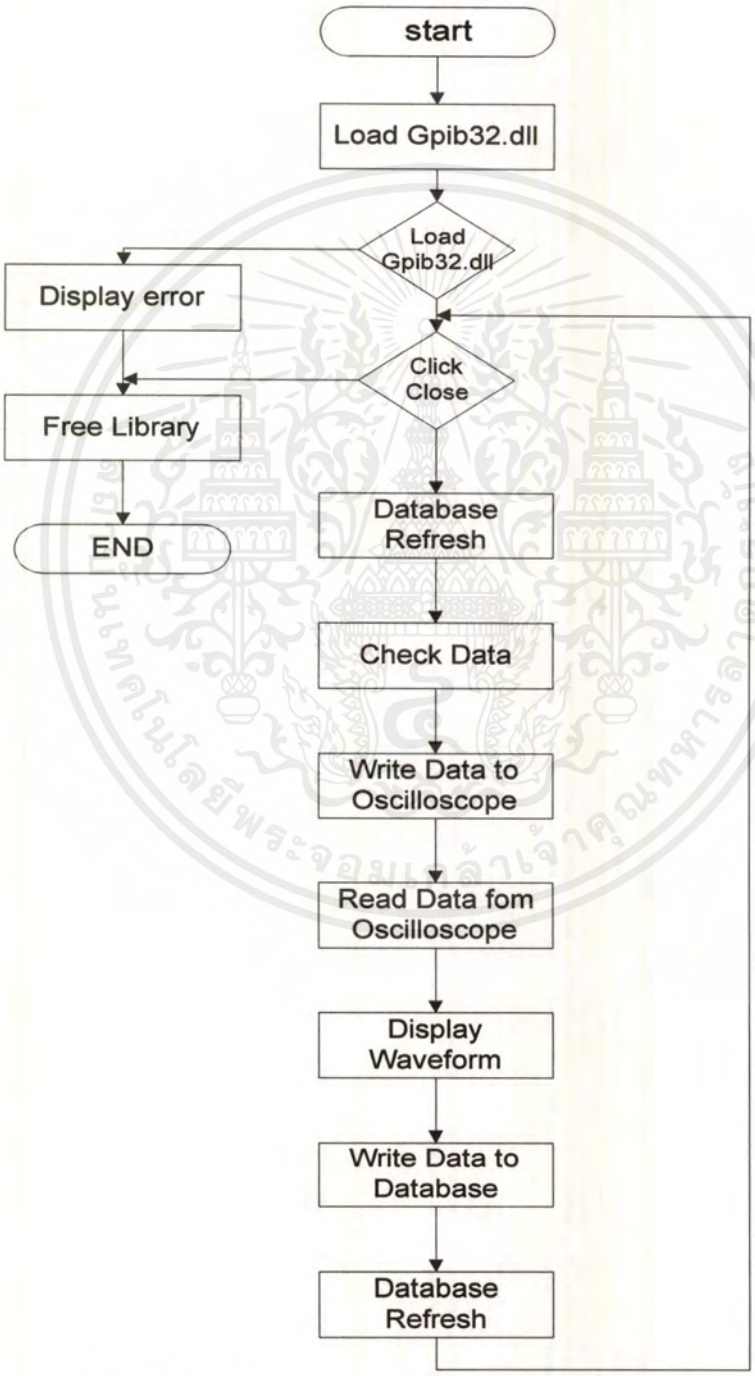
สำหรับการทำงานของโปรแกรมแสดงดังรูปที่ 5.21



รูปที่ 5.21 ผังการทำงานของโปรแกรม User Oscilloscope

5.5.2 โปรแกรม Master Pragrammable Oscilloscope

โปรแกรมส่วนนี้จะมีหน้าที่รับค่าฟังก์ชั่นจากดาต้าเบสมาสั่งงาน Programmable Oscilloscope และอ่านข้อมูลจาก Programmable Oscilloscope ส่งมาเก็บที่ดาต้าเบส การทำงานของโปรแกรมแสดงดังรูปที่ 5.22



รูปที่ 5.22 ผังการทำงานของ Master Programmable Oscilloscope

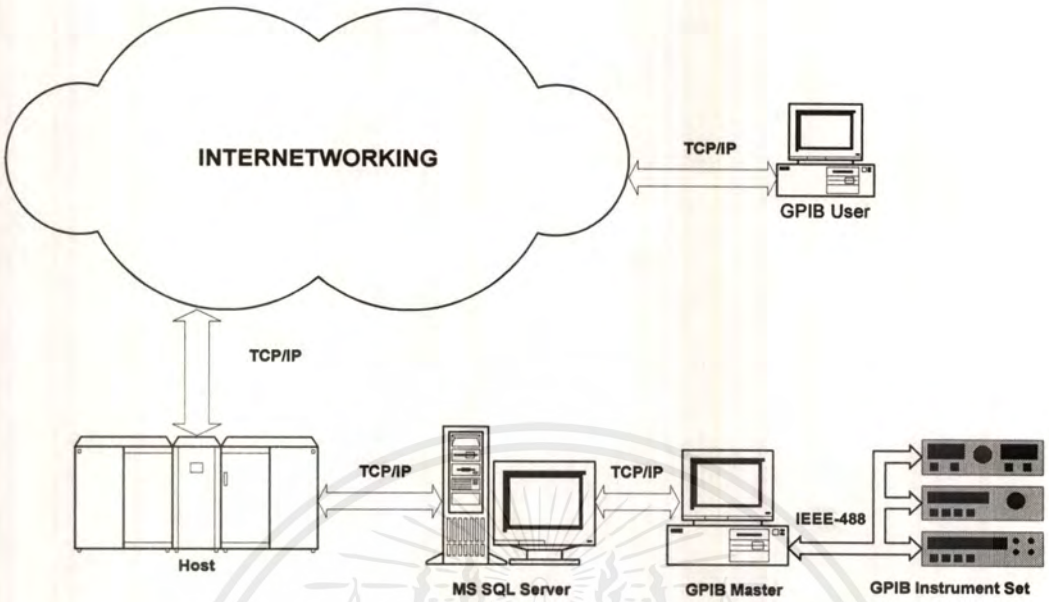
5.6 การควบคุม GPIB ในสถาวะ REMOTE

การควบคุม GPIB ในสถาวะ REMOTE เป็นการควบคุม GPIB ที่อยู่ที่ user และ user จะควบคุมได้ทั้งหมดทั้งการเปิด และ ปิด ของโปรแกรม และการควบคุมอุปกรณ์ GPIB ซึ่งจะมีโปรแกรมที่มีหน้าตาดังรูป

รูปที่ 5.23 หน้าจอของโปรแกรมการควบคุมอุปกรณ์ GPIB ผ่านเครือข่ายอินเตอร์เนตทั้ง user และ master

5.6.1 โครงสร้าง และ การทำงานของการควบคุม GPIB ในสถาวะ REMOTE

การทำงานจะเป็นการส่งข้อมูลจาก user ไปถึง master อย่างต่อเนื่อง โดยเมื่อ user ถูกเปิดโดยการเรียกจากโปรแกรม MUNU ของ user เองแล้ว ก่อนที่ user ตัวนั้นจะทำงาน user ตัวนั้นจะส่งรหัส “open” ไปเก็บที่ Table ของ user นั้นๆ เมื่อส่งไปแล้ว user ตัวนั้นก็จะทำงานตามปรกติ



รูปที่ 5.24 โครงสร้างของการทำงานในสถาวะ REMOTE

และในขณะเดียวกันตัวโปรแกรม MENU ของ master จะคอยขอบริการ server เพื่อคอยตรวจเช็คว่า user ตัวใดถูกเรียกใช้งานบาง โดยการตรวจที่ Table ของ user นั้นๆ ถ้าพบว่า user ตัวใดถูกเปิด โปรแกรม MENU ของ master จะเปิดโปรแกรมของ user นั้นๆ และทำการส่งรหัสเข้าไปที่ Table ของ user นั้นๆเพื่อเป็นการบอก ตัว user ว่าตอนนี้ตัว master ได้เปิดใช้งานแล้ว

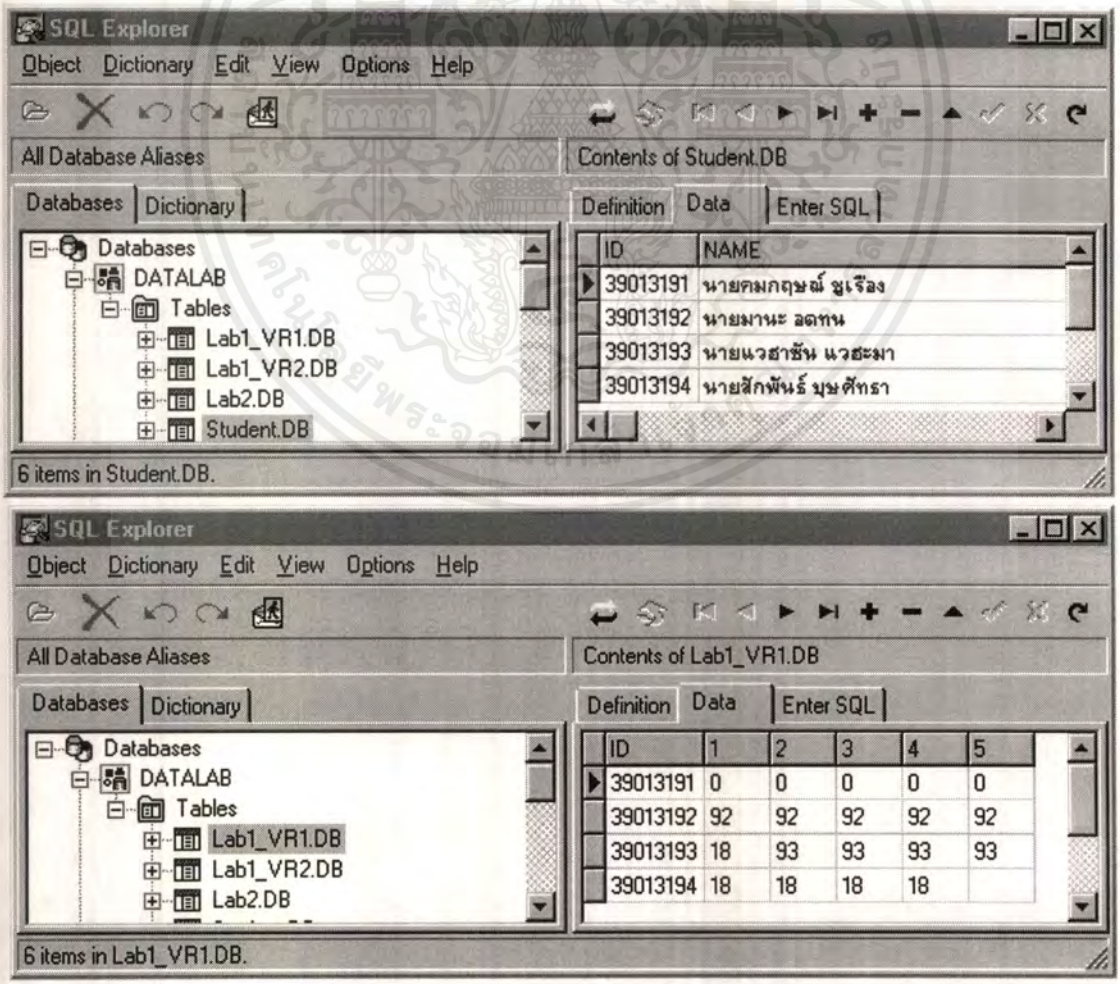
ในทำนองเดียวกันการปิดของโปรแกรมจะมีการทำงานเหมือนกันกับการเปิดโปรแกรมนั้นเอง

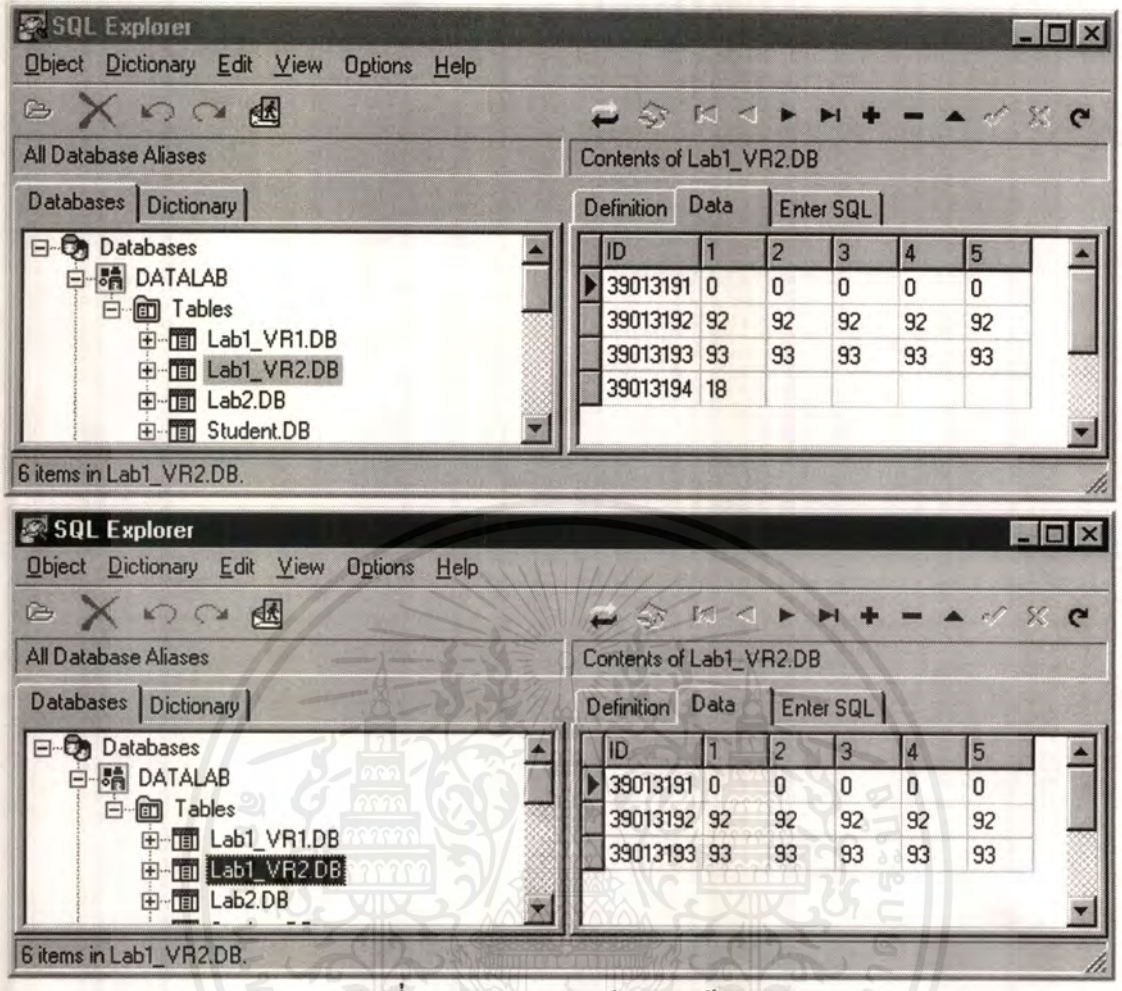
บทที่ 6
ตัวอย่างการประยุกต์ใช้งาน

ในตัวอย่างการประยุกต์ใช้งานนี้ ได้ยกตัวอย่างเป็นการสร้างการทดลองทางอิเล็กทรอนิกส์ขึ้นมา เพื่อให้สามารถทำการทดลองตามการทดลองและเก็บรายละเอียดของแต่ละคนที่ทำการทดลองไว้ เมื่อเข้ามาทดลองใช้ภายหลังก็สามารถนำข้อมูลที่ได้ทำการทดลองไว้แล้วมาใช้ได้อีก

การออกแบบ

ในส่วนของการออกแบบโปรแกรมนั้น ก็จะต้องสร้างฐานข้อมูลขึ้นมาเพื่อเก็บรายละเอียดต่างๆ เช่น รายละเอียดของผลการทดลองและรายละเอียดของนักศึกษาที่ทำการทดลอง รายละเอียดของฐานข้อมูลที่ใช้แสดงดังรูป

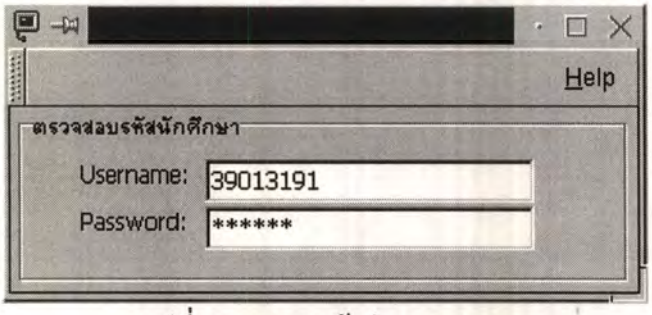




รูปที่ 6.1 แสดงรายละเอียดฐานข้อมูล

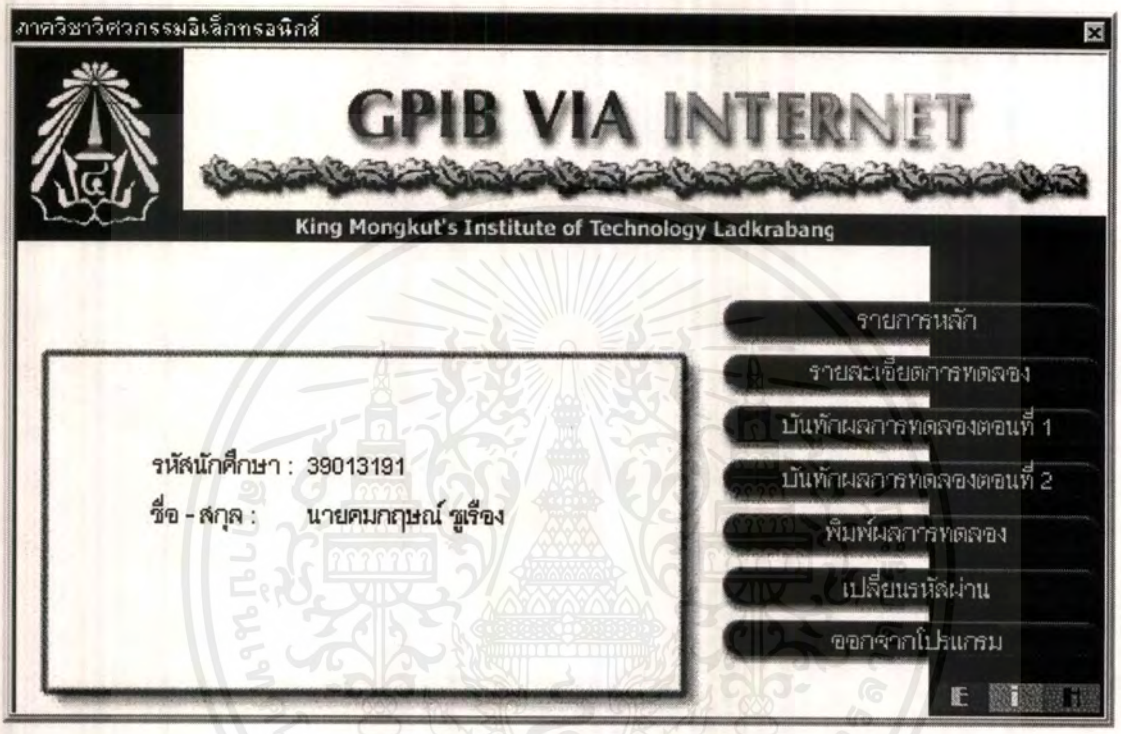
การใช้โปรแกรม

เริ่มจากผู้ที่ดูแลการทดลองจะต้องทำการเพิ่ม Account ให้กับนักศึกษาเพื่อจะได้นำไปใช้ในการ login ก่อนการทดลอง ดังรูปแสดงส่วนการ login ก่อนการทดลอง

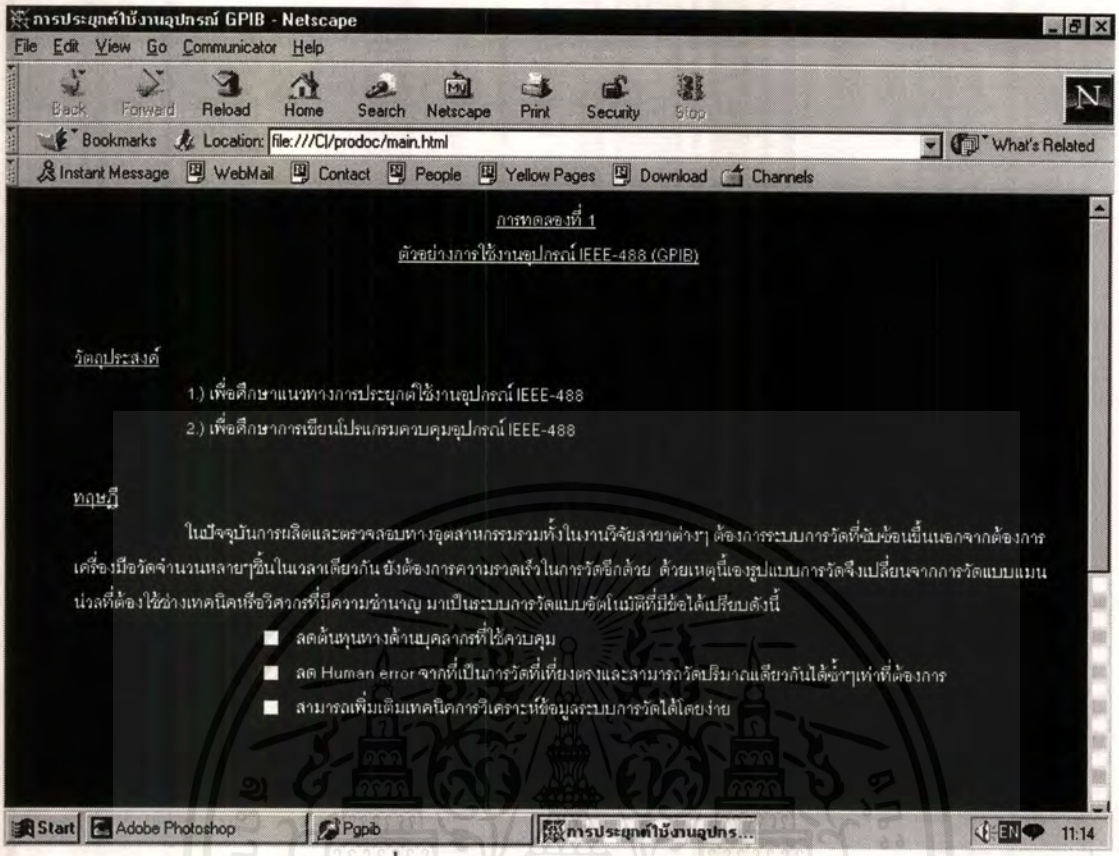


รูปที่ 6.2 แสดงหน้าต่างการ login

เมื่อนักศึกษาทำการ login ผ่านก็จะสามารถเข้าสู่หน้าต่างหลักของโปรแกรม นักศึกษาก็จะเลือกรายการที่จะแสดงรายละเอียดของการทดลองจากนั้นก็จะเลือกรายการเพื่อบันทึกผลการทดลอง การทดลองก็ใช้ควบคู่ไปกับโปรแกรมการควบคุมอุปกรณ์ GPIB เพื่อที่จะรับผลการการวัดมาบันทึกลงฐานข้อมูล

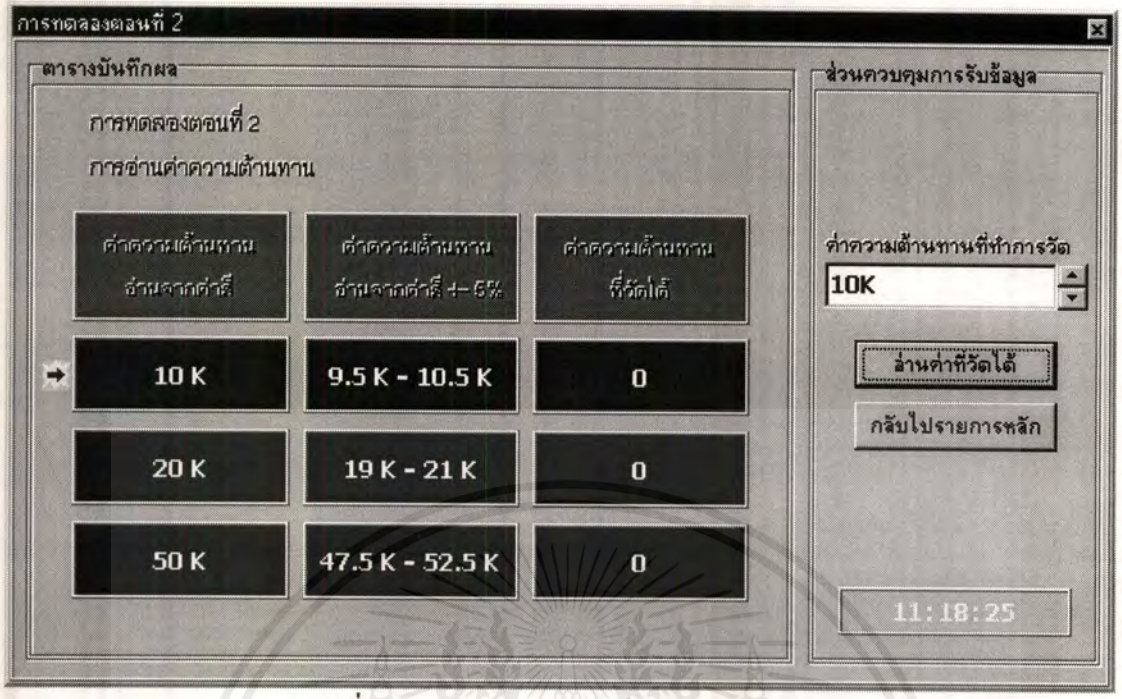


รูปที่ 6.3 แสดงหน้าต่างหลักของโปรแกรม



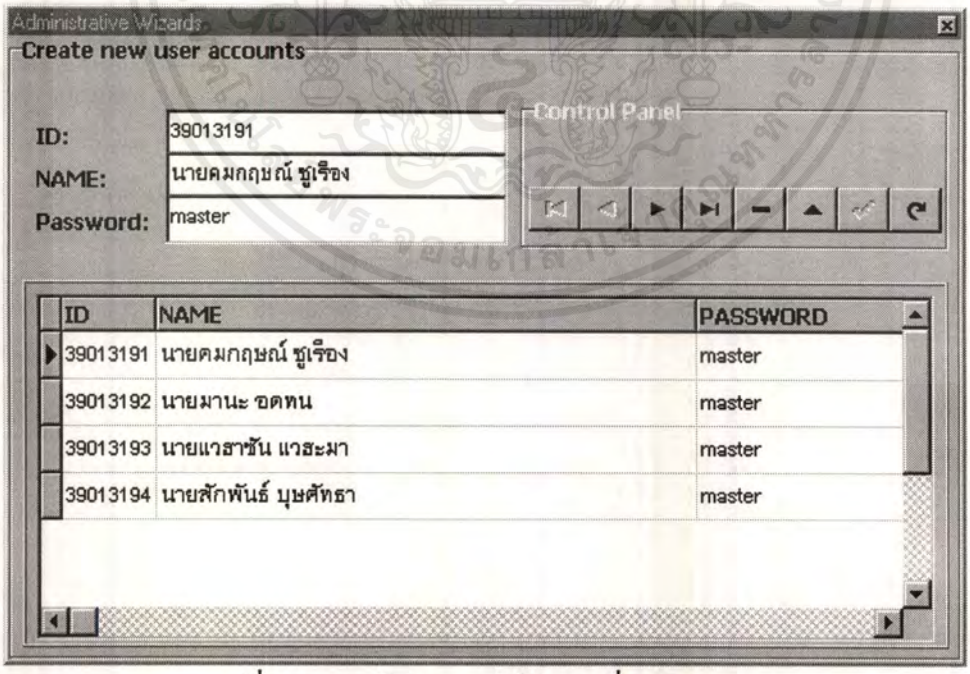
รูปที่ 6.4 แสดงรายละเอียดการทดลอง

รูปที่ 6.5 แสดงส่วนการบันทึกผลการทดลอง



รูปที่ 6.6 แสดงส่วนการบันทึกผลการทดลอง

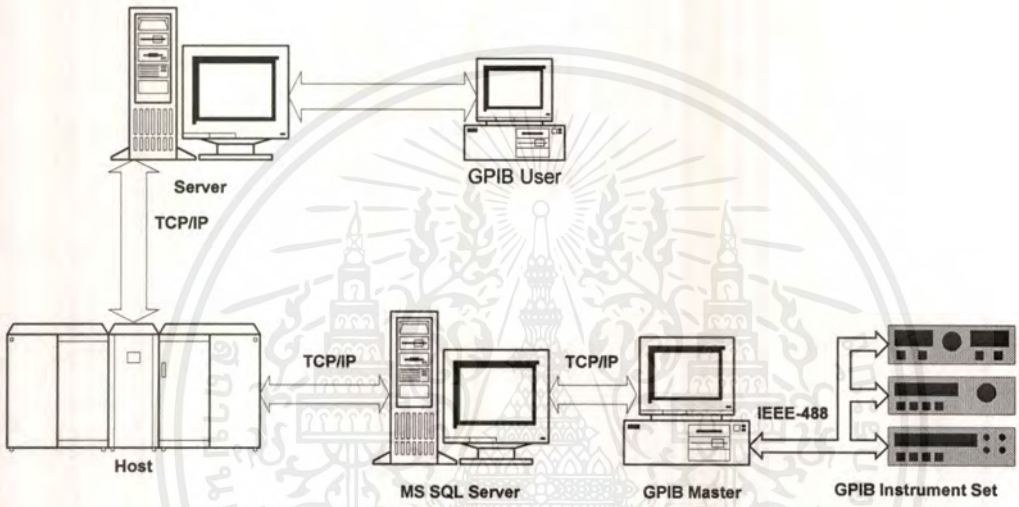
ในส่วนของผู้ดูแลว่าใครมีสิทธิในการทดลองก็จะใช้โปรแกรมดังรูปที่ 6.7 เป็นตัวคอยเพิ่มและลบ account ของนักศึกษา



รูปที่ 6.7 แสดงโปรแกรมใช้จัดการเกี่ยวกับ Account

บทที่ 7
การทดลองและผลการทดลอง

ในการทดลองบทนี้เป็นการทดลองเพื่อทดสอบว่า สามารถควบคุมชุดเครื่องมือวัด GPIB ได้จากเครื่องคอมพิวเตอร์ที่ต่อกับระบบอินเตอร์เน็ต ซึ่งสำหรับระบบที่ใช้ในการทดลองจะเป็นดังรูปที่ 7.1



รูปที่ 7.1 แสดงระบบคอมพิวเตอร์ที่ใช้ในการทดลอง

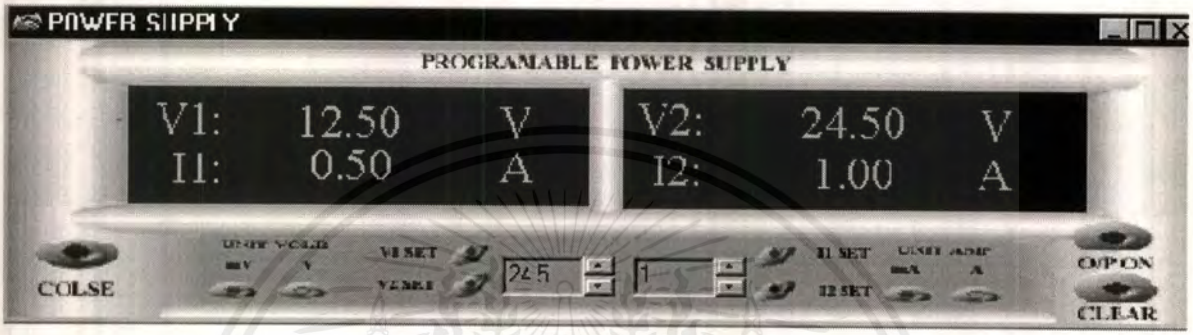
โดยการทดลองจะเป็นการทดสอบการทำงานผ่าน LAN 2 Network ซึ่งแต่ละ network ต่างก็เป็นเนทเวิร์คย่อยของเครือข่ายอินเตอร์เน็ต (subNetwork)โดยเครื่องคอมพิวเตอร์ของ SQL Sever จะอยู่ที่ IP 161.246.2.134 และเครื่อง user อยู่ที่ 161.246.2.134 การทดลองจะเป็นการทดลองสั่งงานอุปกรณ์ทีละตัว

7.1 การทดลองสั่งงาน Programmable Power Supply (HM8142)

ในการทดลอง Programmable Power Supply นี้จะเป็นการทดลองว่า ค่าแรงดัน และค่ากระแสที่ตั้งไว้โดยคอมพิวเตอร์ของ user จะมีค่าของข้อมูลที่ปรากฏในดาตาเบสของ MS SQL Sever ใน Database name ชื่อ gpib ใน Table ที่ชื่อ dbo.power เป็นอย่างไร และผลที่ได้ที่ตัว programmable Power Supply มีค่าเท่าไรเป็นไปตามที่ user ได้ตั้งไว้หรือไม่

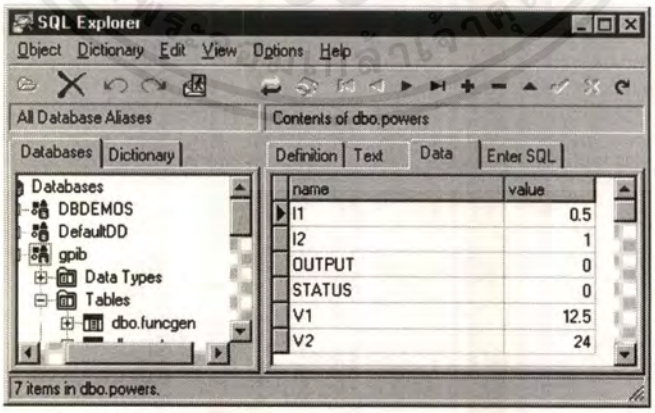
7.1.1 การทดลองตั้งค่าแรงดัน V1 และ V2, I1 และ I2 โดย user

การตั้งค่าแรงดันที่ User สามารถทำได้ 2 แบบ คือ ป้อนค่าโดยการพิมพ์ค่าแรงดันที่ Edit box หรือ Click ที่ลูกศรที่อยู่ติดกับ Edit box โดยลูกศรชี้ขึ้นจะเป็นการเพิ่มค่า ลูกศรชี้ลงจะเป็นการลดค่า จากนั้นถ้าต้องการตั้งค่าแรงดันที่ V1 หรือ V2 ก็ทำได้โดยการ Click ที่ V1 Set หรือ V2 Set ตามลำดับจากการทดลอง Set ค่า V1=12.5 V และ V2=24 V



รูปที่ 7.2 แสดงการตั้งค่า V1 และ V2, I1 และ I2

จากการทดลองจะทดลองป้อนค่า V1=12.5 V และ V2=24 V, I1=0.5 A และ I2=1 A ซึ่งแสดงดังรูปที่ 7.2 เป็นโปรแกรมในส่วนของผู้ใช้ (GPIB User)
จากนั้นจะเช็คว่าค่าที่อยู่ใน Table ของดาตาเบสเมื่อผู้ใช้ป้อนค่าแล้วจะมีค่าอย่างไรซึ่งดูได้จาก Table ชื่อ dbo.powers ในดาตาเบสชื่อ gpib ของ MS SQL Sever จะได้ค่าที่ปรากฏดังนี้

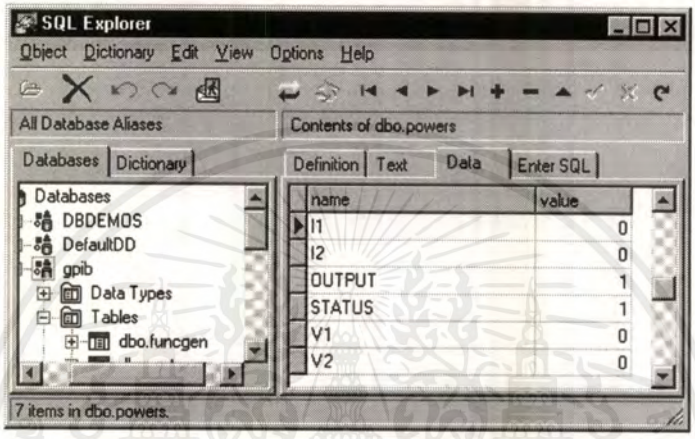


รูปที่ 7.3 ค่าใน Table dbo.powers

7.1.2 การทดลอง Set สถานะของเอาท์พุทและสถานะเคลียร์

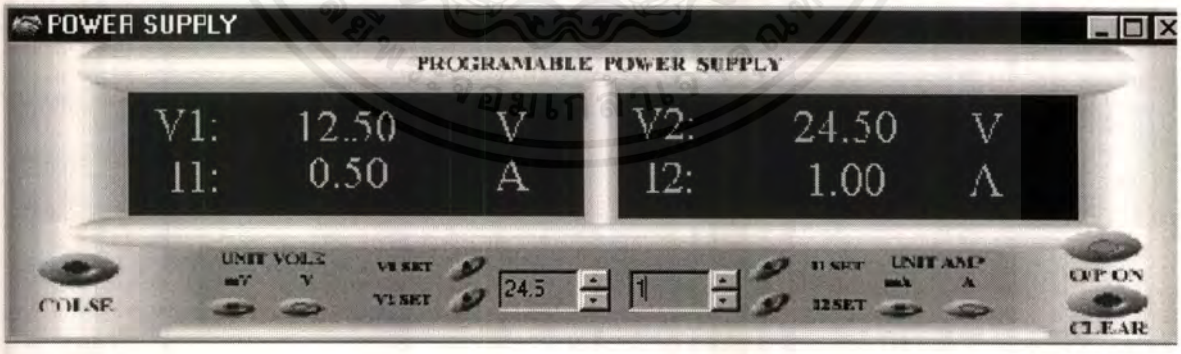
- เมื่อผู้ใช้ Click ที่ปุ่ม clear จะใช้ค่าที่ฟิลด์ Value ของเรคอร์ด status=1
- เมื่อผู้ใช้ Click ที่ปุ่ม output on จะใช้ค่าที่ฟิลด์ Value ของเรคอร์ด output=1

ซึ่งค่าที่ปรากฏที่ตาราง Table dbo.powers เป็นดังรูปที่ 7.4 เป็นสถานะที่ผู้ใช้กดปุ่ม output on=1



รูปที่ 7.4 ค่าใน Table dbo.powers ที่สถานะ clear/output on

ค่าที่ปรากฏที่หน้าปัดของโปรแกรม Programmable Power Supply จะเป็นดังรูปที่ 7.5



รูปที่ 7.5 แสดงค่าที่ปรากฏที่โปรแกรม Power Supply ด้าน Sever

7.2 การทดลองสั่งงาน Programmable Function Generator (HM8130)

ในการทดลอง Programmable Function Generator นี้จะเป็นการทดลองว่าค่า ความถี่, แอมปลิจูด และ สัญญาณ ชนิดต่าง ๆ ที่ตั้งโดยผู้ใช้คอมพิวเตอร์ของผู้ใช้ (user) จะมีค่าของข้อมูลที่

ปรากฏในดาตาเบสของ MS SQL Sever ใน Database name ชื่อ gpib ใน Table ที่ชื่อ dbo.funcgen มีค่าเป็นอย่างไร และผลที่เครื่องทำงานตามคำสั่งเป็นจริงอย่างไร ตรงตามที่ตั้งไว้หรือไม่

7.2.1 การทดลองตั้งค่า ความถี่, แอมปลิจูด และ ชนิดของสัญญาณ

7.2.1.1 การตั้งค่าโดย User

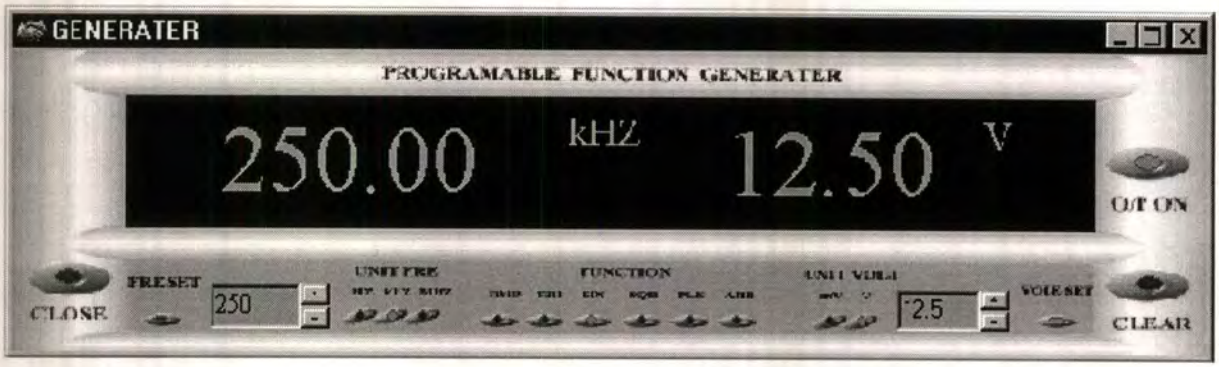
ในขั้นตอนนี้ User ทำการตั้งค่า ความถี่, แอมปลิจูด และ ชนิดของสัญญาณ ตามต้องการ แต่เนื่องจากความสามารถของ Programmable Function Generator (HM8130) ในการกำเนิดสัญญาณต่าง ๆ มีค่าจำกัดเมื่อ User ตั้งค่าเกิน Range ที่สูงสุดที่จะสามารถกำเนิดได้ ก็จะได้ค่าเท่ากับค่าสูงสุดที่สามารถกำเนิดได้ โดยที่แต่ละชนิดของสัญญาณก็จะมีขีดจำกัดของค่าความถี่ และ แอมปลิจูดสูงสุดต่างกัน ซึ่งได้แสดงไว้เป็น Flow Hint ที่ปุ่มตั้งค่านั้น ๆ ดังรูที่ 7.6

รูปที่ 7.6 แสดงค่าสูงสุดของสัญญาณที่ HM8130 สามารถกำเนิดได้

ในการทดลองด้าน User ได้ Set ค่าต่าง ๆ ไว้ดังนี้

- Frequency = 250 kHz
- Amplitude = 12.5 Volt
- Signal Type = sin
- Output = ON

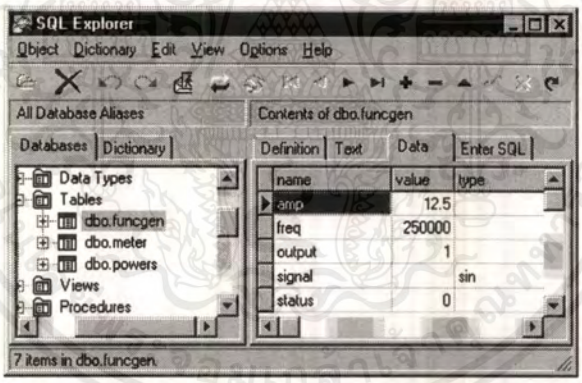
จะได้ค่าที่ปรากฏที่ Monitor ของ User ดังรูที่ 7.7



รูปที่ 7.7 แสดงค่าที่ปรากฏที่หน้าปัดของโปรแกรมด้าน User

7.2.1.2 ข้อมูลในดาตาเบสของ Function Generator

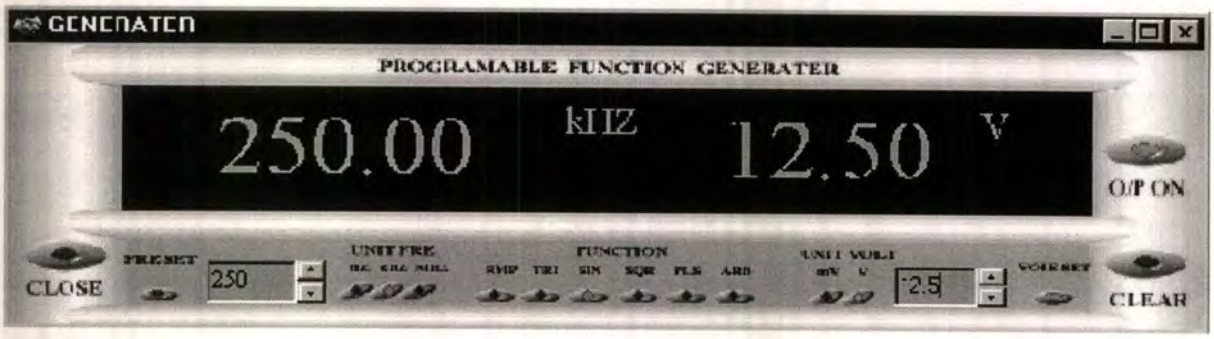
ข้อมูลที่ User สั่งงานจะถูกนำมาเก็บไว้ที่ Database name ชื่อ gpib ใน Table ชื่อ dbo.funcgen ซึ่งค่าที่ได้จะเป็นดังรูปที่ 7.8



รูปที่ 7.8 แสดงข้อมูลใน dbo.funcgen

7.2.1.3 ค่าที่ปรากฏจริงที่ Programmable Function Generator

คอมพิวเตอร์ที่ติดต่ออยู่กับ Programmable Function Generator จะนำค่าจากดาตาเบสมาสั่งงาน HM8130 ซึ่งค่าที่ได้แสดงดังรูปที่ 7.9



รูปที่ 7.9 ค่าที่ปรากฏที่หน้าปัดของโปรแกรมด้าน Sever

ซึ่งค่าที่แสดงเป็นค่าของโปรแกรมด้าน Master GPIB ซึ่งจะมีค่าเท่ากับค่าที่ HM8130 จริง เพราะเป็นค่าที่อ่านมาจาก Buffer ของ HM8130

7.3 การทดลองสั่งงาน Programmable Multimeter (HM8112-2)

ในการทดลอง Programmable Multimeter นี้จะทดลองว่า Multimeter สามารถรับคำสั่งจาก User และส่งค่าให้ User ได้อย่างถูกต้องหรือไม่ ซึ่งฟังก์ชันของ Multimeter ได้แก่ เลือกชนิดของการวัด และเลือกย่านของการวัด

7.3.1 การทดลองตั้งค่าโดย User

ในขั้นตอนนี้ User จะทำการตั้งชนิดของสัญญาณ และอ่านค่าของสัญญาณที่วัดได้ ย่านของ Multimeter ที่ได้เขียนโปรแกรมไว้มีดังนี้

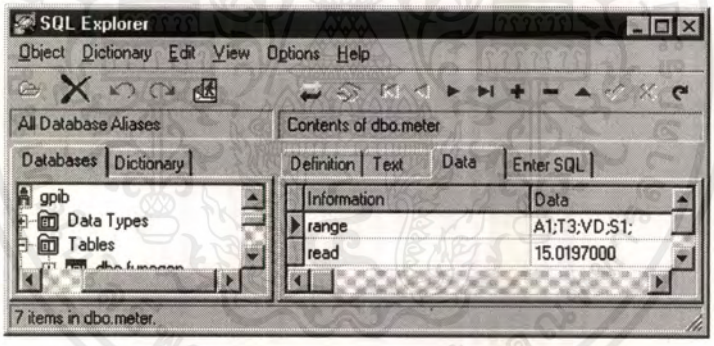
1. VDC
2. VAC
3. ADC
4. AAC
5. OHM
6. Auto Range

ในการทดลองนี้จะแสดงถึงการตั้งค่าวัด VDC และอ่านค่าจาก Power Supply ซึ่งต่ออยู่ด้าน Master GPIB โดยจะมี Range เป็น Auto Range ค่าที่ปรากฏที่หน้าปัดของ Multimeter จะเป็นค่าที่อ่านมาจากดาตาเบส และค่าในดาตาเบสจะได้รับจาก Buffer ของ Multimeter ดังแสดงในรูปที่ 7.10

รูปที่ 7.10 แสดงหน้าปัดของโปรแกรมด้าน GPIB User

7.3.2 ข้อมูลในดาตาเบสของ Multimeter

จะประกอบไปด้วยชุดคำสั่งที่ได้รับจาก User เมื่อ User ทำการตั้งค่าการวัด และอีกส่วนหนึ่งเป็นข้อมูลที่อ่านมาจาก Buffer ของ Multimeter ซึ่งค่าที่ได้จากการตั้งค่าวัด VDC และวัดค่าจาก Power Supply แสดงได้ดังรูปที่ 7.11



รูปที่ 7.11 ข้อมูลในดาตาเบสของ Multimeter

คอมพิวเตอร์ที่เป็นตัวควบคุม Multimeter จะนำคำสั่งจากดาตาเบสมาสั่งการที่ Multimeter ซึ่งจะแสดงผลพร้อมกับส่งค่าที่อ่านได้ไปยังดาตาเบส ซึ่งค่าที่ปรากฏที่หน้าปัดของ Multimeter เป็นดังรูปที่ 7.12 ซึ่งเป็นค่าที่หน้าปัดของโปรแกรมด้าน Sever ซึ่งจะเป็นค่าเท่ากันกับค่าที่ปรากฏที่ Multimeter

รูปที่ 7.12 แสดงหน้าปัดของโปรแกรมด้าน Sever

7.4 การทดลองสั่งงาน Programmable Oscilloscope (Tektronix TDS 360)

ในการทดลอง Programmable Oscilloscope จะทดลองว่า Oscilloscope สามารถรับค่าฟังก์ชั่นที่สั่งงานโดย User และ โปรแกรม User Oscilloscope สามารถนำข้อมูลจาก Programmable Oscilloscope มาแสดงผลได้ถูกต้องหรือไม

7.4.1 การทดลองตั้งค่าโดย User

ในขั้นตอนนี้ User จะสามารถตั้งค่าฟังก์ชั่นต่างๆ ได้ดังนี้

1.Time per Division

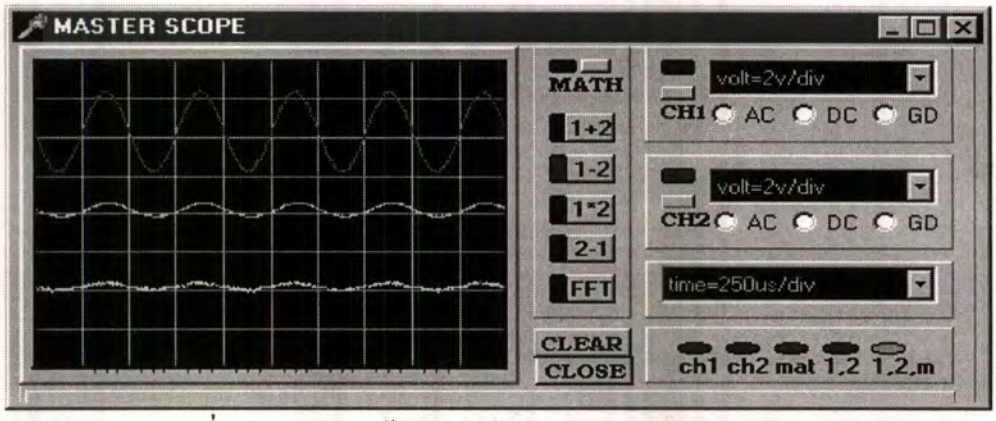
2.Volt per Division

3.Channel Coupling

4.Select Channel

5.ฟังก์ชั่นคณิตศาสตร์

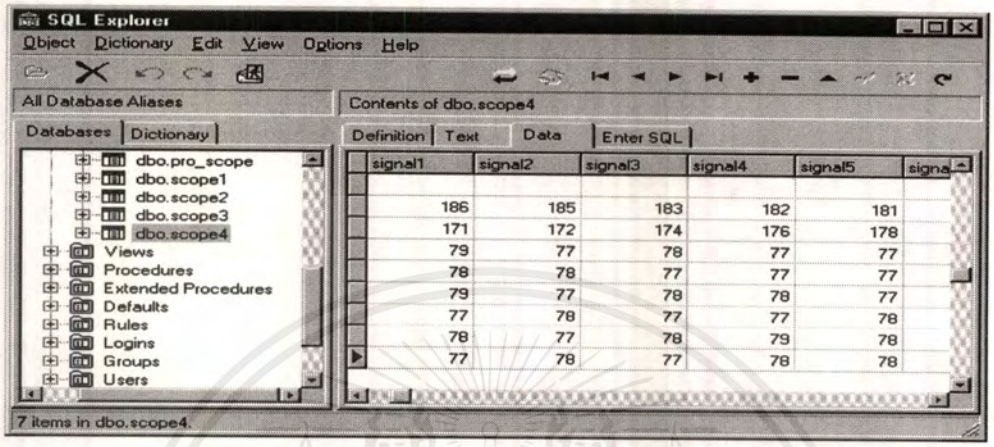
ค่าที่ปรากฏที่หน้าปัดของ Oscilloscope จะเป็นค่าที่ได้จากดาต้าเบส โดยค่าในดาต้าเบสได้รับมาจาก Oscilloscope แสดงดังรูปที่ 7.13



รูปที่ 7.13 แสดงหน้าปัดของโปรแกรมขณะวัดสัญญาณ

7.4.2 ข้อมูลในดาต้าเบสของ Oscilloscope

จะประกอบด้วยค่าฟังก์ชั่นที่ได้รับจากโปรแกรม User Oscilloscope และข้อมูลที่ได้จาก Oscilloscope



รูปที่ 7.14 ข้อมูลในดาต้าเบสของ Oscilloscope

บทที่ 8
สรุปและวิจารณ์

จากการทดสอบการทำงานของโปรแกรม พบว่าสามารถทำการควบคุมชุดเครื่องมือวัดมาตรฐาน IEEE-488 ผ่านระบบเครือข่ายอินเตอร์เน็ตได้จริง โดยค่าความผิดพลาดมีค่าน้อย แต่ยังมีข้อบกพร่องเรื่องความเร็วในการทำงานของระบบ ซึ่งจะต้องมีการหน่วง (Delay) ของข้อมูล โดยอุปกรณ์ที่มีการ Delay มากที่สุดคือ Programmable oscilloscope ซึ่งค่าหน่วงเวลานี้ขึ้นอยู่กับความเร็วในการทำงานของอุปกรณ์,ความเร็วของการทำงานของคอมพิวเตอร์ในระบบ และอัตราความเร็วในการส่งข้อมูลรหัส ASCII จำนวนมากผ่านตัวกลางของ Network

สำหรับแนวทางในการพัฒนาโครงงานนี้ สามารถทำได้โดยการปรับปรุงการทำงานของโปรแกรม ให้มีประสิทธิภาพมากยิ่งขึ้นและให้เร็วมากขึ้น ซึ่งอุปกรณ์ที่ควรพัฒนาเพิ่มเข้ามาได้แก่ หรือเครื่องมืออื่น ๆ ที่ใช้ในระบบการวัดและทดสอบทางอิเล็กทรอนิกส์ หรือการพัฒนาระบบ ให้สามารถติดต่อหรือ ควบคุม ได้โดยที่ ไม่ต้องใช้โปรแกรม ฐานข้อมูล เช่น การควบคุมผ่านทาง web site โดยอาศัยโปรแกรม web browser

หนังสืออ้างอิง

1. NI-488.2M Function Reference Manual for Win32, National Instrument Corporation, 1996
2. NI-488.2M Function Reference Manual for Windows95 and Windows NT, National Instrument Corporation, 1996
3. Todd Miller, David Powell, Etal, USING DELPHI3, Que Corpoation, 1997
4. L.Rantanen, "Using NI-488.2M Software with 32-bit Delphi Application," APPlication Note 085, National Instrument.

ภาคผนวก

****

program client multimeter

****

```
unit clientmeterV4;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls, ComCtrls, Buttons, StdCtrls,multimeter1,multimeter2,multimeter3,
  multimeter4,multimeter5, Grids, DBGrids, Db, DBTables;

type
  TMULTIMETER = class(TForm)
    DataSource1: TDataSource;
    Table1: TTable;
    Timer1: TTimer;
    panel1: TPanel;
    Shape1: TShape;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Panel2: TPanel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
```

Label7: TLabel;

Label8: TLabel;

Label9: TLabel;

VDC: TSpeedButton;

VAC: TSpeedButton;

ADC: TSpeedButton;

AAC: TSpeedButton;

OHM: TSpeedButton;

Shape2: TShape;

Shape3: TShape;

Shape4: TShape;

Shape5: TShape;

Shape6: TShape;

Panel3: TPanel;

Label12: TLabel;

AUTO: TSpeedButton;

Shape8: TShape;

Button1: TButton;

Panel4: TPanel;

Image1: TImage;

Label10: TLabel;

Label11: TLabel;

Shape7: TShape;

Shape9: TShape;

Shape10: TShape;

Shape11: TShape;

Shape12: TShape;

Shape13: TShape;

```
Shape14: TShape;
Shape15: TShape;
Shape16: TShape;
Table1information: TStringField;
Table1data: TStringField;
Database1: TDatabase;

procedure AUTOClick(Sender: TObject);
procedure VDCClick(Sender: TObject);
procedure VACClick(Sender: TObject);
procedure ADCClick(Sender: TObject);
procedure AACClick(Sender: TObject);
procedure OHMClick(Sender: TObject);
procedure FormClose(Sender: TObject; var Action: TCloseAction);
procedure Button1Click(Sender: TObject);
procedure Timer1Timer(Sender: TObject);
procedure FormShow(Sender: TObject);
procedure Shape7MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Shape16MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Shape9MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Shape10MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Shape12MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Shape14MouseDown(Sender: TObject; Button: TMouseButton;
```

```
    Shift: TShiftState; X, Y: Integer);
  procedure Shape15MouseDown(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);

 private

  { Private declarations }
 public
  { Public declarations }
 end;
var
  MULTIMETER : TMULTIMETER;
 implementation
{$R *.DFM}

procedure sadang(index:integer);
begin
  case  index of
    1: form3.Showmodal; //VDC
    2: form2.Showmodal; //VAC
    3: form4.Showmodal; //OHM
    4: form5.Showmodal; //ADC
    5: form6.Showmodal;  //AAC
  end;
 end;

procedure autorange(sett:integer);
var  buf:packed array[0..32]of char;
```

```
    msg:string;
begin
    case sett of
    1:begin
        buf:='A1;T3;VD;S1;';            //VDC
        msg:=buf;
        MULTIMETER.Table1.FindNearest(['range']);
        MULTIMETER.Table1.Edit;
        MULTIMETER.table1Data.AsVariant:=msg;
        MULTIMETER.Table1.Post;
        MULTIMETER.label11.caption:='V';
          end ;
    2:begin
        buf:='A1;T3;VA;S1;';          // VAC
        msg:=buf;
        MULTIMETER.Table1.FindNearest(['range']);
        MULTIMETER.Table1.Edit;
        MULTIMETER.table1Data.AsVariant:=msg;
        MULTIMETER.Table1.Post;
        MULTIMETER.Label11.caption:='V';
          end;
    3:begin
        buf:='A1;T3;ID;S1;';        //Idc
        msg:=buf;
        MULTIMETER.Table1.FindNearest(['range']);
        MULTIMETER.Table1.Edit;
        MULTIMETER.table1Data.AsVariant:=msg;
        MULTIMETER.Table1.Post;
```

```
        MULTIMETER.label11.caption:='mA';
          end;
    4:begin
        buf:='A1;T3;IA;S1;';     // Iac
        msg:=buf;
        MULTIMETER.Table1.FindNearest(['range']);
        MULTIMETER.Table1.Edit;
        MULTIMETER.table1Data.AsVariant:=msg;
        MULTIMETER.Table1.Post;
        MULTIMETER.label11.caption:='mA';
          end ;
    5:begin
        buf:='A1;T3;O2;S1;';    //OHM
        msg:=buf;
        MULTIMETER.Table1.FindNearest(['range']);
        MULTIMETER.Table1.Edit;
        MULTIMETER.table1Data.AsVariant:=msg;
        MULTIMETER.Table1.Post;
        MULTIMETER.label11.caption:='Kohm';
          end ;
     end;
end;
procedure TMULTIMETER.AUTOClick(Sender: TObject);
begin
  if shape16.brush.color =clGreen then
    begin
        shape16.Brush.color:=clLime;
     if  shape9.Brush.color=clLime then      //VDC
```

```
        autorange(1);
    if  shape10.Brush.color=clLime then     //VAC
        autorange(2);
    if  shape12.Brush.color=clLime then     //ADC
        autorange(3);
    if  shape14.Brush.color=clLime then     //AAC
        autorange(4);
    if  shape15.Brush.color=clLime then     //OHM
        autorange(5);
  end
  else
    begin shape16.Brush.Color:= clGreen;
    end;
end;
procedure TMULTIMETER.VDCClick(Sender: TObject);
var  buf :packed array [0..32] of char;
  msg :string; i:integer;
begin
  shape9.brush.color:=clLime;
  shape10.brush.color:=clGreen;
  shape12.brush.color:=clGreen;
  shape14.brush.color:=clGreen;
  shape15.brush.color:=clGreen;
    if shape16.brush.color <> clLime  then
    begin
        buf:='A0;T3;VD;S1;';
        sadang(1);
    for i:=1 to length(form3.Label1.Caption) do
```

```
        buf[i+11]:=form3.Label1.Caption[i];
        msg:=buf;
        MULTIMETER.Table1.FindNearest(['range']);
        MULTIMETER.Table1.Edit;
        MULTIMETER.table1Data.AsVariant:=msg;
        MULTIMETER.Table1.Post;
        label11.caption:='V';
       end
    else
      begin
       autorange(1);
      end ;
end;
procedure TMULTIMETER.VACClick(Sender: TObject);
var  buf:packed array[0..32]of char;
    i:integer; msg:string;
begin
     shape9.Brush.color:=clGreen;
     shape10.brush.color:=clLime;
     shape12.brush.color:=clGreen;
     shape14.brush.color:=clGreen;
     shape15.brush.color:=clGreen;

    if shape16.brush.color <> clLime  then
     begin
       sadang(2);
        buf:='A0;T3;VA;S1;';
     for i:=1 to length(form2.Label1.Caption) do
```

```
        buf[i+11]:=form2.Label1.Caption[i];
        msg:=buf;
        MULTIMETER.Table1.FindNearest(['range']);
        MULTIMETER.Table1.Edit;
        MULTIMETER.table1Data.AsVariant:=msg;
        MULTIMETER.Table1.Post;
        label11.caption:='V';
       end
     else
        begin
        autorange(2);
        end;
end;

procedure TMULTIMETER.ADCClick(Sender: TObject);
var  buf:packed array[0..32]of char;
     i:integer; msg:string;
begin
      shape9.brush.color:=clGreen;
      shape10.brush.color:=clGreen;
      shape12.brush.color:=clLime;
      shape14.brush.color:=clGreen;
      shape15.brush.color:=clGreen;
     if shape16.brush.color <> clLime  then
      begin
        sadang(4);
        buf:='A0;T3;ID;S1;';
      for i:=1 to length(form5.Label1.Caption) do
```

```
        buf[i+11]:=form5.Label1.Caption[i];
        msg:=buf;
        MULTIMETER.Table1.FindNearest(['range']);
        MULTIMETER.Table1.Edit;
        MULTIMETER.table1Data.AsVariant:=msg;
        MULTIMETER.Table1.Post;
        label11.Caption:='mA';
      end
       else
        begin
          autorange(3);
        end;
end;



procedure TMULTIMETER.AACClick(Sender: TObject);
var  buf:packed array[0..32]of char;
     i:integer; msg:string;
begin
      shape9.brush.color:=clGreen;
      shape10.brush.color:=clGreen;
      shape12.brush.color:=clGreen;
      shape14.brush.color:=clLime;
      shape15.brush.color:=clGreen;
     if shape16.brush.color <> clLime  then
     begin
         sadang(5);
```

```
        buf:='A0;T3;IA;S1;';
      for i:=1 to length(form6.Label1.Caption) do
        buf[i+11]:=form6.Label1.Caption[i];
        msg:=buf;
        MULTIMETER.Table1.FindNearest(['range']);
        MULTIMETER.Table1.Edit;
        MULTIMETER.table1Data.AsVariant:=msg;
        MULTIMETER.Table1.Post;
        label11.caption:='mA';
     end
      else
       begin
       autorange(4);
      end;
end;



procedure TMULTIMETER.OHMClick(Sender: TObject);
var  buf:packed array[0..32]of char;
    i:integer; msg:string;
begin
    shape9.brush.color:=clGreen;
    shape10.brush.color:=clGreen;
    shape12.brush.color:=clGreen;
    shape14.brush.color:=clGreen;
    shape15.brush.color:=clLime;
   if shape16.brush.color <> clLime  then
```

```
      begin
        sadang(3);
        buf:='A0;T3;O2;S1;';
      for i:=1 to length(form4.Label1.Caption) do
        buf[i+11]:=form4.Label1.Caption[i];
        msg:=buf;
        MULTIMETER.Table1.FindNearest(['range']);
        MULTIMETER.Table1.Edit;
        MULTIMETER.table1Data.AsVariant:=msg;
        MULTIMETER.Table1.Post;
        label11.caption:='Kohm';
      end
    else
      begin
      autorange(5);
      end;
end;



procedure TMULTIMETER.FormClose(Sender: TObject; var Action: TCloseAction);
begin
      ACTION:=CAFREE;
end;
procedure TMULTIMETER.Button1Click(Sender: TObject);
begin
      Close;
end;
```

```
procedure TMULTIMETER.Timer1Timer(Sender: TObject);
begin
      table1.Refresh;
      table1.FindNearest(['read']);
      label10.caption:= table1Data.AsString;
end;
procedure TMULTIMETER.FormShow(Sender: TObject);
begin
Table1.Active:=true;
 Table1.FindNearest(['power']);
    Table1.Edit ;
    table1data.AsVariant:='on';
    Table1.Post;
     timer1.Enabled := true;
     shape9.Brush.color:=clLime;
     AUTO.Click;
end;
```

```
procedure TMULTIMETER.Shape7MouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
 Table1.FindNearest(['power']);
    Table1.Edit ;
    table1data.AsVariant:='off';
```

```
    Table1.Post;

Button1.click;
end;

procedure TMULTIMETER.Shape16MouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
  AUTO.Click
end;

procedure TMULTIMETER.Shape9MouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
VDC.Click;
end;

procedure TMULTIMETER.Shape10MouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
VAC.Click;
end;


procedure TMULTIMETER.Shape12MouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
 ADC.Click;
```

```
end;

procedure TMULTIMETER.Shape14MouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
AAC.Click;
end;

procedure TMULTIMETER.Shape15MouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
OHM.Click;
end;

end.
```

****

program client powersupply

****

```
unit client_power;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls, ComCtrls, StdCtrls, Db, DBTables;

type
  TForm1 = class(TForm)
    Image1: TImage;
    Shape1: TShape;
    Shape2: TShape;
    Shape3: TShape;
    Shape4: TShape;
    Shape5: TShape;
    Shape6: TShape;
    Shape7: TShape;
    Shape8: TShape;
    Shape9: TShape;
    Shape10: TShape;
    Shape11: TShape;
```

```
    Edit1: TEdit;
    UpDown1: TUpDown;
    Edit2: TEdit;
    UpDown2: TUpDown;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Label5: TLabel;
    Label6: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    Label9: TLabel;
    Label10: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    DataSource1: TDataSource;
    Table1: TTable;
    Table1name: TStringField;
    Table1value: TStringField;
    Database1: TDatabase;
    procedure Shape1MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure Shape3MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure Edit1KeyPress(Sender: TObject; var Key: Char);
    procedure UpDown1MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
```

```
    procedure UpDown2MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure Edit2KeyPress(Sender: TObject; var Key: Char);
    procedure Shape2MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure Shape4MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure Shape5MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure Shape6MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure Shape7MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure Shape8MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure Shape9MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure Shape10MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure Shape11MouseDown(Sender: TObject; Button: TMouseButton;
      Shift: TShiftState; X, Y: Integer);
    procedure FormShow(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
var
```

```
  Form1: TForm1;
implementation

{$R *.DFM}


procedure TForm1.Shape1MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
 Table1.FindNearest(['power']);
    Table1.Edit ;
    table1Value.AsVariant:='off';
    Table1.Post;
close;
end;

procedure TForm1.Shape3MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
    shape4.Brush.color:=clGreen;
    shape5.Brush.color:=clGreen;
    shape6.Brush.color:=clGreen;
    shape7.Brush.color:=clGreen;
    shape8.Brush.color:=clGreen;
    shape9.Brush.color:=clGreen;
    shape10.Brush.color:=clGreen;
    shape11.Brush.color:=clGreen;
    shape3.Brush.color:=clLime;
```

```
shape5.Brush.color:=clLime;
shape7.Brush.color:=clLime;
Table1.FindNearest(['status']);
Table1.Edit;
Table1Value.AsVariant:= 1;
Table1.Post;

Table1.FindNearest(['I1']);
Table1.Edit;
table1Value.AsVariant:= 0;
Table1.Post;

 Table1.FindNearest(['I2']);
Table1.Edit ;
table1Value.AsVariant:=0;
Table1.Post;

 Table1.FindNearest(['V1']);
Table1.Edit ;
table1Value.AsVariant:=0;
Table1.Post;

 Table1.FindNearest(['V2']);
Table1.Edit ;
table1Value.AsVariant:=0;
Table1.Post;

Table1.FindNearest(['unit_v']);
```

```
    Table1.Edit ;
    table1Value.AsVariant:=2;
    Table1.Post;

    Table1.FindNearest(['unit_I']);
    Table1.Edit ;
    table1Value.AsVariant:=2;
    Table1.Post;

    Label9.Caption := 'V';
    Label10.Caption := 'V';
    Label5.Caption := '0.00';
    Label6.Caption := '0.00';

    Label11.Caption := 'A';
    Label12.Caption := 'A';
    Label7.Caption := '0.00';
    Label8.Caption := '0.00';

    edit1.text:=IntTostr(0);
    edit2.text:=IntTostr(0);

end;

procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);

 var num:real;
   code:integer;
```

```
begin
      Shape8.Brush.color:= clGreen;
      Shape9.Brush.color:= clGreen;
     if((key < '0')or(key >'9'))and(key<>'.')then
      begin
         MessageBeep(0);
         key:=#0;
      end;
      val(edit1.text,num,code);
     if ( num >1000) then
     begin
     MessageBeep(0);
         edit1.text:=IntTostr(999);
     end;

end;

procedure TForm1.UpDown1MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
      Shape8.Brush.color:= clGreen;
      Shape9.Brush.color:= clGreen;
end;

procedure TForm1.UpDown2MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
      Shape10.Brush.color:= clGreen;
```

```
      Shape11.Brush.color:= clGreen;
end;

procedure TForm1.Edit2KeyPress(Sender: TObject; var Key: Char);

 var num:real;
   code:integer;
begin
      Shape10.Brush.color:= clGreen;
      Shape11.Brush.color:= clGreen;
     if((key < '0')or(key >'9'))and(key<>'.')then
      begin
         MessageBeep(0);
         key:=#0;
      end;
      val(edit1.text,num,code);
     if ( num >1000) then
     begin
     MessageBeep(0);
         edit1.text:=IntTostr(999);
     end;

end;

procedure TForm1.Shape2MouseDown(Sender: TObject; Button: TMouseButton;
 Shift: TShiftState; X, Y: Integer);
begin
 if shape2.Brush.color = clGreen then
```

```
      begin
        shape2.Brush.color:=clLime;
        Table1.FindNearest(['output']);
        Table1.Edit;
        Table1value.AsVariant := 1;
        Table1.Post;

      end
    else
      begin
        shape2.brush.color:=clGreen;
        Table1.FindNearest(['output']);
        Table1.Edit;
        Table1value.AsVariant :=0;
        Table1.Post;
      end;
  end;

procedure TForm1.Shape4MouseDown(Sender: TObject; Button: TMouseButton;
 Shift: TShiftState; X, Y: Integer);

Var
   signal:string;
begin
    shape4.Brush.color:=clLime;
   shape5.Brush.color:=clGreen;
   shape3.Brush.color:=clGreen;
   shape8.Brush.color:=clGreen;
```

```
    shape9.Brush.color:=clGreen;

    signal:= '1';
    Table1.FindNearest(['unit_v']);
    Table1.Edit;
    table1value.AsVariant:= signal;
    Table1.Post;
     Table1.FindNearest(['status']);
    Table1.Edit;
    table1value.AsVariant:= 0;
    Table1.Post;


end;



procedure TForm1.Shape5MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);

Var
    signal:string;
begin
     shape5.Brush.color:=clLime;
    shape4.Brush.color:=clGreen;
    shape3.Brush.color:=clGreen;
    shape8.Brush.color:=clGreen;
```

```
    shape9.Brush.color:=clGreen;

    signal:= '2';
    Table1.FindNearest(['unit_v']);
    Table1.Edit;
    table1value.AsVariant:= signal;
    Table1.Post;
     Table1.FindNearest(['status']);
    Table1.Edit;
    table1value.AsVariant:= 0;
    Table1.Post;

end;



procedure TForm1.Shape6MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);

Var
    signal:string;
begin
     shape6.Brush.color:=clLime;
    shape7.Brush.color:=clGreen;
```

```
    shape10.Brush.color:=clGreen;
    shape11.Brush.color:=clGreen;
    shape3.Brush.color:=clGreen;

    signal:= '1';
    Table1.FindNearest(['unit_I']);
    Table1.Edit;
    table1value.AsVariant:= signal;
    Table1.Post;
     Table1.FindNearest(['status']);
    Table1.Edit;
    table1value.AsVariant:= 0;
    Table1.Post;

end;



procedure TForm1.Shape7MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);

Var
    signal:string;
begin
```

```
    shape7.Brush.color:=clLime;
    shape6.Brush.color:=clGreen;
    shape10.Brush.color:=clGreen;
    shape11.Brush.color:=clGreen;
    shape3.Brush.color:=clGreen;

    signal:= '2';
    Table1.FindNearest(['unit_I']);
    Table1.Edit;
    table1value.AsVariant:= signal;
    Table1.Post;
    Table1.FindNearest(['status']);
    Table1.Edit;
    table1value.AsVariant:= 0;
    Table1.Post;
end;


procedure TForm1.Shape8MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
var
  ch1:string;
  ch2:char;
 num:real;
 num1:real;
 bum:real;
    code:integer;
    digit:integer;
```

```
  value:char;
  freq :string;
begin
  Shape8.Brush.color:= clLime;
  Shape9.Brush.color:= clGreen;
   Shape10.Brush.color:= clGreen;
   Shape11.Brush.color:= clGreen;
   Shape3.Brush.color:= clGreen;
   val(edit1.text,num,code);
    if ( num >=1000) then
     begin
     MessageBeep(0);
         edit1.text:=IntTostr(999);
     end;

     val(edit1.text,num,code);

      Table1.FindNearest(['UNIT_V']);
       ch1:=table1Value.asString;

      if ch1='3' then
        label9.Caption:='V'
      else if ch1='1'then begin
           label9.Caption:='mV';
           num1 := num*0.001;
           end
      else if ch1='2' then begin
           label9.Caption:='V';
```

```
          num1:=num;
        if ( num > 30  ) then
          begin
          MessageBeep(0);
          edit1.text:=IntTostr(30);
           num1:=30;
           num:=30;
          end;

        end;

    Table1.FindNearest(['v1']);
    Table1.Edit ;
    table1Value.AsVariant:=num1;
    Table1.Post;
    Table1.FindNearest(['status']);
    Table1.Edit;
    table1value.AsVariant:= 0;
    Table1.Post;

    str(num:5:2,freq);
    label5.caption:=freq;

end;
```

```
procedure TForm1.Shape9MouseDown(Sender: TObject; Button: TMouseButton;
 Shift: TShiftState; X, Y: Integer);

 var
 ch1:string;
 ch2:char;
num:real;
num1:real;
bum:real;
  code:integer;
  digit:integer;
  value:char;
  freq :string;
begin
  Shape9.Brush.color:= clLime;
  Shape8.Brush.color:= clGreen;
   Shape10.Brush.color:= clGreen;
   Shape11.Brush.color:= clGreen;
   Shape3.Brush.color:= clGreen;
   val(edit1.text,num,code);
    if ( num >=1000) then
     begin
     MessageBeep(0);
         edit1.text:=IntTostr(999);
     end;

     val(edit1.text,num,code);
```

```
Table1.FindNearest(['UNIT_V']);
 ch1:=table1Value.asString;

 if ch1='3' then
   label9.Caption:='V'
 else if ch1='1'then begin
     label10.Caption:='mV';
     num1 := num*0.001;
     end
 else if ch1='2' then begin
     label10.Caption:='V';
      num1:=num;
     if ( num > 30  ) then
       begin
       MessageBeep(0);
       edit1.text:=IntTostr(30);
        num1:=30;
        num:=30;
       end;

     end;

Table1.FindNearest(['v2']);
Table1.Edit ;
table1Value.AsVariant:=num1;
Table1.Post;
Table1.FindNearest(['status']);
Table1.Edit;
```

```
    table1value.AsVariant:= 0;
    Table1.Post;

    str(num:5:2,freq);
    label6.caption:=freq;

end;


procedure TForm1.Shape10MouseDown(Sender: TObject; Button: TMouseButton;
 Shift: TShiftState; X, Y: Integer);

 var
  ch1:string;
  ch2:char;
 num:real;
 num1:real;
 bum:real;
   code:integer;
   digit:integer;
   value:char;
   freq :string;
 begin
   Shape10.Brush.color:= clLime;
   Shape8.Brush.color:= clGreen;
    Shape9.Brush.color:= clGreen;
    Shape11.Brush.color:= clGreen;
    Shape3.Brush.color:= clGreen;
```

```
val(edit2.text,num,code);
 if ( num >=1000) then
  begin
  MessageBeep(0);
       edit2.text:=IntTostr(999);
  end;

  val(edit2.text,num,code);

  Table1.FindNearest(['UNIT_I']);
   ch1:=table1Value.asString;

   if ch1='3' then
     label9.Caption:='V'
   else if ch1='1'then begin
        label11.Caption:='mA';
        num1 := num*0.001;
        end
   else if ch1='2' then begin
        label11.Caption:='A';
         num1:=num;
        if ( num > 1  ) then
          begin
          MessageBeep(0);
          edit2.text:=IntTostr(1);
           num1:=1;
           num:=1;
          end;
```

```
        end;

    Table1.FindNearest(['I1']);
    Table1.Edit ;
    table1Value.AsVariant:=num1;
    Table1.Post;
    Table1.FindNearest(['status']);
    Table1.Edit;
    table1value.AsVariant:= 0;
    Table1.Post;

    str(num:5:2,freq);
    label7.caption:=freq;

end;



procedure TForm1.Shape11MouseDown(Sender: TObject; Button: TMouseButton;
 Shift: TShiftState; X, Y: Integer);

 var
 ch1:string;
 ch2:char;
num:real;
num1:real;
bum:real;
```

```
  code:integer;
  digit:integer;
  value:char;
  freq :string;
begin
  Shape11.Brush.color:= clLime;
  Shape10.Brush.color:= clGreen;
   Shape9.Brush.color:= clGreen;
   Shape8.Brush.color:= clGreen;
   Shape3.Brush.color:= clGreen;
   val(edit2.text,num,code);
    if ( num >=1000) then
     begin
     MessageBeep(0);
         edit2.text:=IntTostr(999);
     end;

     val(edit2.text,num,code);

      Table1.FindNearest(['UNIT_I']);
      ch1:=table1Value.asString;

      if ch1='3' then
        label9.Caption:='V'
      else if ch1='1'then begin
           label12.Caption:='mA';
           num1 := num*0.001;
           end
```

```
    else if ch1='2' then begin
        label12.Caption:='A';
         num1:=num;
        if ( num > 1  ) then
          begin
          MessageBeep(0);
          edit2.text:=IntTostr(1);
           num1:=1;
           num:=1;
          end;

       end;

  Table1.FindNearest(['I2']);
  Table1.Edit ;
  table1Value.AsVariant:=num1;
  Table1.Post;
  Table1.FindNearest(['status']);
  Table1.Edit;
  table1value.AsVariant:= 0;
  Table1.Post;

  str(num:5:2,freq);
  label8.caption:=freq;

end;
```

```
procedure TForm1.FormShow(Sender: TObject);
begin
Table1.Active:=true;
 Table1.FindNearest(['power']);
    Table1.Edit ;
    table1Value.AsVariant:='on';
    Table1.Post;
end;

end.
```

****

program client generater

****

```
unit client_gen1;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls, ComCtrls, StdCtrls, Db, DBTables;

type
  TForm1 = class(TForm)
    Image1: TImage;
    Shape1: TShape;
    Shape2: TShape;
    Shape3: TShape;
    Shape4: TShape;
    Shape5: TShape;
    Shape6: TShape;
    Shape7: TShape;
    Shape8: TShape;
    Shape9: TShape;
    Shape10: TShape;
    Shape11: TShape;
```

```
Shape12: TShape;
Shape13: TShape;
Shape14: TShape;
Shape15: TShape;
Shape16: TShape;
Edit1: TEdit;
UpDown1: TUpDown;
Edit2: TEdit;
UpDown2: TUpDown;
Label1: TLabel;
Label2: TLabel;
Label3: TLabel;
Label4: TLabel;
DataSource1: TDataSource;
Table1: TTable;
Table1name: TStringField;
Table1value: TStringField;
Table1type: TStringField;
Database1: TDatabase;
procedure Edit1KeyPress(Sender: TObject; var Key: Char);
procedure Shape4MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Shape5MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Shape3MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Shape1MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
```

```
procedure Shape2MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Shape11MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Shape12MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Shape13MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Shape14MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Shape15MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Shape16MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Shape6MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Shape7MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Shape8MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Shape9MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Shape10MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Edit2KeyPress(Sender: TObject; var Key: Char);
procedure UpDown1MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure UpDown2MouseDown(Sender: TObject; Button: TMouseButton;
```

```
    Shift: TShiftState; X, Y: Integer);
  procedure FormShow(Sender: TObject);

 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 Form1: TForm1;

implementation

{$R *.DFM}

procedure TForm1.Edit1KeyPress(Sender: TObject; var Key: Char);
var num:real;
  code:integer;
begin
     Shape4.Brush.color:= clGreen;
    if((key < '0')or(key >'9'))and(key<>'.')then
     begin
        MessageBeep(0);
        key:=#0;
     end;
     val(edit1.text,num,code);
    if ( num >=1000) then
```

```
      begin
      MessageBeep(0);
          edit1.text:=IntTostr(999);
      end;

end;

procedure TForm1.Shape4MouseDown(Sender: TObject; Button: TMouseButton;
 Shift: TShiftState; X, Y: Integer);
 var
  ch1:string;
  ch2:char;
 num:real;
 num1:real;
 bum:real;
    code:integer;
    digit:integer;
    value:char;
    freq :string;
 begin
    Shape4.Brush.color:= clLime;
    Shape5.Brush.color:= clGreen;
     Shape3.Brush.color:= clGreen;
     val(edit1.text,num,code);
      if ( num >=1000) then
       begin
       MessageBeep(0);
           edit1.text:=IntTostr(999);
```

```
end;

val(edit1.text,num,code);

 Table1.FindNearest(['UNIT_F']);
 ch1:=table1Value.asString;

 if ch1='1' then
 begin
   label2.Caption:='HZ';
   num1 := num ;
   end
 else if ch1='2'then begin
      label2.Caption:='kHZ';
      num1 := num*1000;
      end
 else if ch1='3' then begin
      label2.Caption:='MHZ';
       num1:=num*1000000;
      if ( num > 10  ) then
        begin
        MessageBeep(0);
        edit1.text:=IntTostr(10);
         num1:=10000000;
         num:=10;
        end;

      end;
```

```
            bum := num;
Table1.FindNearest(['freq']);
Table1.Edit ;
table1Value.AsVariant:=num1;
Table1.Post;
Table1.FindNearest(['status']);
Table1.Edit;
table1value.AsVariant:= 0;
Table1.Post;

 if (num1<0.01) then value:= '1'
 else if (num1>=0.01) and (num1<1e3) then value:= '2'
      else if  (num1>=1e3) and (num1<1e6) then value:= '3'
           else if (num1>=1e6) and (num1<10e6) then value:= '4'
                else if (num1>=10e6) then value:= '5'
                     else value:='0';
case value of
  '1': begin
        digit:=2;
       end;
  '2': begin
         digit:=2;
       end;
  '3': begin

       if (num1>=1e3) and (num1<10e3) then digit:=4
         else if (num1>=10e3) and (num1<100e3) then digit := 3
              else if (num1>=100e3) and (num1<1e6) then digit := 2
```

```
                else digit :=2;

            end;
        '4': begin
                digit := 4;
            end;
        '5': begin
                digit:=4;
            end;

    end;
    str(num:7:digit,freq);
    label1.caption:=freq;

end;

procedure TForm1.Shape5MouseDown(Sender: TObject; Button: TMouseButton;
 Shift: TShiftState; X, Y: Integer);
 var
 ch1:string;
 ch2:char;
num:real;
num1:real;
bum:real;
   code:integer;
   digit:integer;
   value:char;
```

```
  freq :string;
begin
 Shape4.Brush.color:= clGreen;
  Shape5.Brush.color:= clLime;
   Shape3.Brush.color:= clGreen;
   val(edit2.text,num,code);
    val(edit2.text,num,code);
     Table1.FindNearest(['UNIT_V']);
      ch1:=table1Value.asString;

       if ch1='1'then begin
          label4.Caption:='mV';
            if num>=100 then
             num1 := num*0.001
             else

             num1 := num*0.001;
        end

      else if ch1='2' then begin
          label4.Caption:='V';
          num1:=num;
           if ( num >20) then
            begin
             MessageBeep(0);
             edit2.text:=IntTostr(20);
             num1 := 20;
             num := 20;
```

```
            end;
          end;

            bum := num;
    Table1.FindNearest(['amp']);
    Table1.Edit ;
    table1Value.AsVariant:=num1;
    Table1.Post;
    Table1.FindNearest(['status']);
    Table1.Edit;
    table1value.AsVariant:= 0;
    Table1.Post;
    str(num:6:2,freq);
    label3.caption:=freq;
end;


procedure TForm1.Shape3MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
    shape13.Brush.color:=clLime;
    shape11.Brush.color:=clGreen;
    shape12.Brush.color:=clGreen;
    shape14.Brush.color:=clGreen;
    shape15.Brush.color:=clGreen;
    shape16.Brush.color:=clGreen;
    shape7.Brush.color:=clLime;
    shape6.Brush.color:=clGreen;
```

```
shape8.Brush.color:=clGreen;
shape10.Brush.color:=clLime;
shape9.Brush.color:=clGreen;
shape3.Brush.color:=clLime;
 shape4.Brush.color:=clGreen;
  shape5.Brush.color:=clGreen;


Table1.FindNearest(['status']);
Table1.Edit;
table1value.AsVariant:= 1;
Table1.Post;

Table1.FindNearest(['signal']);
Table1.Edit;
table1type.AsVariant:= 'sin';
Table1.Post;


Table1.FindNearest(['Unit_f']);
Table1.Edit ;
table1Value.AsVariant:=2;
Table1.Post;

 Table1.FindNearest(['Unit_v']);
Table1.Edit ;
table1Value.AsVariant:=2;
Table1.Post;
```

```
    Label2.Caption := 'kHZ';
    Label4.Caption := 'V';
    Label1.Caption := '1.0000';
    Label3.Caption := '10.0';

     Table1.FindNearest(['freq']);
    Table1.Edit ;
    table1Value.AsVariant:=1000;
    Table1.Post;

    Table1.FindNearest(['amp']);
    Table1.Edit;
    table1Value.AsVariant:=10.0;
    Table1.Post;

end;


procedure TForm1.Shape1MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
    Table1.FindNearest(['power']);
    Table1.Edit ;
    table1type.AsVariant:='off';
    Table1.Post;
```

```
Close;
end;

procedure TForm1.Shape2MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
  if shape2.Brush.color = clGreen then
        begin
          shape2.Brush.color:=clLime;
          Table1.FindNearest(['output']);
          Table1.Edit;
          Table1value.AsVariant := 1;
          Table1.Post;

        end
      else
        begin
          shape2.brush.color:=clGreen;
          Table1.FindNearest(['output']);
          Table1.Edit;
          Table1value.AsVariant :=0;
          Table1.Post;
        end;
      end;

procedure TForm1.Shape11MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
```

```
Var
    signal:string;
begin
     shape11.Brush.color:=clLime;
    shape12.Brush.color:=clGreen;
    shape13.Brush.color:=clGreen;
    shape14.Brush.color:=clGreen;
    shape15.Brush.color:=clGreen;
    shape16.Brush.color:=clGreen;
    shape4.Brush.color:=clGreen;
    shape5.Brush.color:=clGreen;
     Shape3.Brush.color:= clGreen;

    signal:= 'rmp';
    Table1.FindNearest(['signal']);
    Table1.Edit;
    table1type.AsVariant:= signal;
    Table1.Post;
     Table1.FindNearest(['status']);
    Table1.Edit;
    table1value.AsVariant:= 0;
    Table1.Post;


end;

procedure TForm1.Shape12MouseDown(Sender: TObject; Button: TMouseButton;
```

```
  Shift: TShiftState; X, Y: Integer);

Var
    signal:string;
begin
     shape12.Brush.color:=clLime;
    shape11.Brush.color:=clGreen;
    shape13.Brush.color:=clGreen;
    shape14.Brush.color:=clGreen;
    shape15.Brush.color:=clGreen;
    shape16.Brush.color:=clGreen;
     shape4.Brush.color:=clGreen;
    shape5.Brush.color:=clGreen;
     Shape3.Brush.color:= clGreen;

    signal:= 'tri';
    Table1.FindNearest(['signal']);
    Table1.Edit;
    table1type.AsVariant:= signal;
    Table1.Post;
     Table1.FindNearest(['status']);
    Table1.Edit;
    table1value.AsVariant:= 0;
    Table1.Post;
end;

procedure TForm1.Shape13MouseDown(Sender: TObject; Button: TMouseButton;
```

```
  Shift: TShiftState; X, Y: Integer);

Var
    signal:string;
begin
     shape13.Brush.color:=clLime;
    shape12.Brush.color:=clGreen;
    shape11.Brush.color:=clGreen;
    shape14.Brush.color:=clGreen;
    shape15.Brush.color:=clGreen;
    shape16.Brush.color:=clGreen;
    shape4.Brush.color:=clGreen;
    shape5.Brush.color:=clGreen;
     Shape3.Brush.color:= clGreen;

    signal:= 'sin';
    Table1.FindNearest(['signal']);
    Table1.Edit;
    table1type.AsVariant:= signal;
    Table1.Post;
     Table1.FindNearest(['status']);
    Table1.Edit;
    table1value.AsVariant:= 0;
    Table1.Post;


end;
```

```
procedure TForm1.Shape14MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);

 Var
    signal:string;
begin
     shape14.Brush.color:=clLime;
    shape12.Brush.color:=clGreen;
    shape13.Brush.color:=clGreen;
    shape11.Brush.color:=clGreen;
    shape15.Brush.color:=clGreen;
    shape16.Brush.color:=clGreen;
    shape4.Brush.color:=clGreen;
    shape5.Brush.color:=clGreen;
     Shape3.Brush.color:= clGreen;

    signal:= 'sqr';
    Table1.FindNearest(['signal']);
    Table1.Edit;
    table1type.AsVariant:= signal;
    Table1.Post;
     Table1.FindNearest(['status']);
    Table1.Edit;
    table1value.AsVariant:= 0;
    Table1.Post;
```

```
end;


procedure TForm1.Shape15MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);

Var
    signal:string;
begin
     shape15.Brush.color:=clLime;
    shape12.Brush.color:=clGreen;
    shape13.Brush.color:=clGreen;
    shape14.Brush.color:=clGreen;
    shape11.Brush.color:=clGreen;
    shape16.Brush.color:=clGreen;
    shape4.Brush.color:=clGreen;
    shape5.Brush.color:=clGreen;
     Shape3.Brush.color:= clGreen;

    signal:= 'pls';
    Table1.FindNearest(['signal']);
    Table1.Edit;
    table1type.AsVariant:= signal;
    Table1.Post;
     Table1.FindNearest(['status']);
    Table1.Edit;
    table1value.AsVariant:= 0;
```

```
    Table1.Post;


end;


procedure TForm1.Shape16MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);

Var
    signal:string;
begin
     shape16.Brush.color:=clLime;
    shape12.Brush.color:=clGreen;
    shape13.Brush.color:=clGreen;
    shape14.Brush.color:=clGreen;
    shape15.Brush.color:=clGreen;
    shape11.Brush.color:=clGreen;
    shape4.Brush.color:=clGreen;
    shape5.Brush.color:=clGreen;
     Shape3.Brush.color:= clGreen;

    signal:= 'arb';
    Table1.FindNearest(['signal']);
    Table1.Edit;
    table1type.AsVariant:= signal;
    Table1.Post;
```

```
    Table1.FindNearest(['status']);
    Table1.Edit;
    table1value.AsVariant:= 0;
    Table1.Post;


end;
procedure TForm1.Shape6MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);

Var
    signal:string;
begin
    shape6.Brush.color:=clLime;
    shape7.Brush.color:=clGreen;
    shape8.Brush.color:=clGreen;
    shape4.Brush.color:=clGreen;
    shape5.Brush.color:=clGreen;
    Shape3.Brush.color:= clGreen;

    signal:= '1';
    Table1.FindNearest(['unit_f']);
    Table1.Edit;
    table1value.AsVariant:= signal;
    Table1.Post;
    Table1.FindNearest(['status']);
    Table1.Edit;
    table1value.AsVariant:= 0;
```

```
    Table1.Post;



end;



procedure TForm1.Shape7MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);

Var
    signal:string;
begin
     shape7.Brush.color:=clLime;
    shape6.Brush.color:=clGreen;
    shape8.Brush.color:=clGreen;
    shape4.Brush.color:=clGreen;
    shape5.Brush.color:=clGreen;
     Shape3.Brush.color:= clGreen;

    signal:= '2';
    Table1.FindNearest(['unit_f']);
    Table1.Edit;
    table1value.AsVariant:= signal;
    Table1.Post;
     Table1.FindNearest(['status']);
```

```
    Table1.Edit;
    table1value.AsVariant:= 0;
    Table1.Post;


end;



procedure TForm1.Shape8MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);

Var
    signal:string;
begin
     shape8.Brush.color:=clLime;
    shape7.Brush.color:=clGreen;
    shape6.Brush.color:=clGreen;
    shape4.Brush.color:=clGreen;
    shape5.Brush.color:=clGreen;
     Shape3.Brush.color:= clGreen;

    signal:= '3';
    Table1.FindNearest(['unit_f']);
    Table1.Edit;
    table1value.AsVariant:= signal;
```

```
    Table1.Post;
     Table1.FindNearest(['status']);
    Table1.Edit;
    table1value.AsVariant:= 0;
    Table1.Post;


end;



procedure TForm1.Shape9MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);

Var
    signal:string;
begin
     shape9.Brush.color:=clLime;
    shape10.Brush.color:=clGreen;
    shape4.Brush.color:=clGreen;
    shape5.Brush.color:=clGreen;
     Shape3.Brush.color:= clGreen;


    signal:= '1';
    Table1.FindNearest(['unit_v']);
    Table1.Edit;
```

```
    table1value.AsVariant:= signal;
    Table1.Post;
     Table1.FindNearest(['status']);
    Table1.Edit;
    table1value.AsVariant:= 0;
    Table1.Post;


end;



procedure TForm1.Shape10MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);

Var
    signal:string;
begin
     shape10.Brush.color:=clLime;
    shape9.Brush.color:=clGreen;
    shape4.Brush.color:=clGreen;
    shape5.Brush.color:=clGreen;
     Shape3.Brush.color:= clGreen;
```

```
    signal:= '2';
    Table1.FindNearest(['unit_v']);
    Table1.Edit;
    table1value.AsVariant:= signal;
    Table1.Post;
     Table1.FindNearest(['status']);
    Table1.Edit;
    table1value.AsVariant:= 0;
    Table1.Post;


end;



procedure TForm1.Edit2KeyPress(Sender: TObject; var Key: Char);
 var num:real;
   code:integer;
begin
      Shape5.Brush.color:= clGreen;
     if((key < '0')or(key >'9'))and(key<>'.')then
      begin
         MessageBeep(0);
         key:=#0;
      end;
      val(edit1.text,num,code);
```

```
    if ( num >=1000) then
    begin
    MessageBeep(0);
        edit1.text:=IntTostr(999);
    end;

end;


procedure TForm1.UpDown1MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
 Shape4.Brush.color:= clGreen;

end;



procedure TForm1.UpDown2MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
Shape5.Brush.color:= clGreen;

end;

procedure TForm1.FormShow(Sender: TObject);
begin
```

```
Table1.Active:=true;
    Table1.FindNearest(['power']);
    Table1.Edit ;
    table1type.AsVariant:='on';
    Table1.Post;
end;

end.
```

****

program client munu

****

```
unit remoteGPIB;
interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls, StdCtrls, Db, DBTables;

type
  TremoteGPIB1 = class(TForm)
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    Image1: TImage;
```

Table1: TTable;

DataSource1: TDataSource;

Table2: TTable;

DataSource2: TDataSource;

Table3: TTable;

DataSource3: TDataSource;

Table1name: TStringField;

Table1value: TStringField;

Table1type: TStringField;

Table2name: TStringField;

Table2value: TStringField;

Table3information: TStringField;

Table3data: TStringField;

Button5: TButton;

Table4: TTable;

DataSource4: TDataSource;

Table4name: TStringField;

Table4data: TStringField;

Image2: TImage;

Shape1: TShape;

Shape2: TShape;

Shape3: TShape;

Shape4: TShape;

Shape5: TShape;

Shape6: TShape;

Shape7: TShape;

Shape8: TShape;

Shape9: TShape;

```
Shape10: TShape;
Shape11: TShape;
Shape12: TShape;
Database1: TDatabase;
procedure Button1Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Button3Click(Sender: TObject);
procedure Button4Click(Sender: TObject);
procedure Button5Click(Sender: TObject);
procedure Shape1MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Shape11MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Shape12MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Shape2MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Shape3MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Shape4MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Shape5MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Shape6MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Shape7MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
procedure Shape8MouseDown(Sender: TObject; Button: TMouseButton;
```

```
    Shift: TShiftState; X, Y: Integer);
  procedure FormShow(Sender: TObject);
 private
  { Private declarations }
 public
  { Public declarations }
 end;

var
 remoteGPIB1: TremoteGPIB1;

implementation

{$R *.DFM}

procedure TremoteGPIB1.Button1Click(Sender: TObject);
begin
 Table1.FindNearest(['power']);
   Table1.Edit ;
   table1type.AsVariant:='lnlon';
   Table1.Post;
    winexec('d:\project\GPIB via internet\My project\sourceuser\Pclient_generater',SW_Show);
 end;



procedure TremoteGPIB1.Button2Click(Sender: TObject);
```

```
begin
 Table2.FindNearest(['power']);
    Table2.Edit ;
    table2value.AsVariant:='onkk';
    Table2.Post;
     winexec('d:\project\GPIB via internet\My project\sourceuser\Pclient_powersupply',SW_Show);

 end;

procedure TremoteGPIB1.Button3Click(Sender: TObject);
begin
 Table3.FindNearest(['power']);
    Table3.Edit ;
    table3data.AsVariant:='onjjh';
    Table3.Post;
     winexec('d:\project\GPIB via internet\My project\sourceuser\PclientpmeterV4',SW_Show);

end;

procedure TremoteGPIB1.Button4Click(Sender: TObject);
begin
close;
end;

procedure TremoteGPIB1.Button5Click(Sender: TObject);
begin
Table4.FindNearest(['power']);
    Table4.Edit ;
```

```
    table4data.AsVariant:='onjjh';
    Table4.Post;
     winexec('d:\testscope\user6',SW_Show);

end;

procedure TremoteGPIB1.Shape1MouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
  Brush.color:=clred;
  button1.click;
end;

procedure TremoteGPIB1.Shape11MouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
 button4.click;
end;

procedure TremoteGPIB1.Shape12MouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
button4.click;
end;

procedure TremoteGPIB1.Shape2MouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
```

```
 button1.click;
end;

procedure TremoteGPIB1.Shape3MouseDown(Sender: TObject;
 Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
 button2.click;
end;

procedure TremoteGPIB1.Shape4MouseDown(Sender: TObject;
 Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
 button2.click;
end;

procedure TremoteGPIB1.Shape5MouseDown(Sender: TObject;
 Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
 button3.click;
end;

procedure TremoteGPIB1.Shape6MouseDown(Sender: TObject;
 Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
 button3.click;
end;

procedure TremoteGPIB1.Shape7MouseDown(Sender: TObject;
```

```
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
 button4.click;
end;

procedure TremoteGPIB1.Shape8MouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
 button4.click;
end;

procedure TremoteGPIB1.FormShow(Sender: TObject);
begin
Table1.Active:=true;
Table2.Active:=true;
Table3.Active:=true;
Table4.Active:=true;

end;

end.
```

```
**********************************************

program master menu

**********************************************

unit Umaster_remote;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ExtCtrls, Db, DBTables;

type
  Tremote = class(TForm)
    Image1: TImage;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    Button5: TButton;
    Timer1: TTimer;
    Table1: TTable;
    DataSource1: TDataSource;
    Table2: TTable;
    DataSource2: TDataSource;
    Table3: TTable;
    DataSource3: TDataSource;
```

```
  Table4: TTable;
  DataSource4: TDataSource;
  Table1name: TStringField;
  Table1value: TStringField;
  Table1type: TStringField;
  Table2name: TStringField;
  Table2value: TStringField;
  Table3information: TStringField;
  Table3data: TStringField;
  Table4name: TStringField;
  Table4data: TStringField;
  Table4signal: TMemoField;
  Database1: TDatabase;
  Image2: TImage;
  Shape1: TShape;
  Shape2: TShape;
  procedure Button5Click(Sender: TObject);
  procedure Timer1Timer(Sender: TObject);
  procedure FormShow(Sender: TObject);
  procedure Shape1MouseDown(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);
  procedure Shape2MouseDown(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);
private
  { Private declarations }
public
  { Public declarations }
end;
```

```
var
 remote: Tremote;

implementation

{$R *.DFM}

procedure Tremote.Button5Click(Sender: TObject);
begin
close;
end;
procedure clickgenerator;
var test:string[15];
begin
remote.Table1.FindNearest(['power']);
     test:= remote.Table1type.asString;



     remote.Table1.FindNearest(['power']);
     test:= remote.Table1type.asString;

     remote.Table1.FindNearest(['power']);
     test:= remote.Table1type.asString;

     if test='on'   then
     begin
 remote.Table1.FindNearest(['power']);
```

```
remote.Table1.Edit;
remote.Table1type.asString:='open';
remote.Table1.Post;



remote.Table1.FindNearest(['power']);
remote.Table1.Edit;
remote.Table1type.asString:='open';
remote.Table1.Post;


  remote.Table1.FindNearest(['power']);
remote.Table1.Edit;
remote.Table1type.asString:='open';
remote.Table1.Post;


  remote.Table1.FindNearest(['power']);
remote.Table1.Edit;
remote.Table1type.asString:='open';
remote.Table1.Post;

winexec('d:\Sun_client\dataproject\master\master_GPIB\P_master_gen',SW_Show);


    end;
```

```
end;

procedure clickpower;
var test:string[15];
begin
 remote.Table2.FindNearest(['power']);
      test:= remote.Table2value.asString;

 remote.Table2.FindNearest(['power']);
      test:= remote.Table2value.asString;


 remote.Table2.FindNearest(['power']);
      test:= remote.Table2value.asString;

      if test='on'   then
      begin
  remote.Table2.FindNearest(['power']);
  remote.Table2.Edit;
  remote.Table2value.asString:='open';
  remote.Table2.Post;


   remote.Table2.FindNearest(['power']);
  remote.Table2.Edit;
  remote.Table2value.asString:='open';
  remote.Table2.Post;
```

```
  remote.Table2.FindNearest(['power']);
  remote.Table2.Edit;
  remote.Table2value.asString:='open';
  remote.Table2.Post;


   remote.Table2.FindNearest(['power']);
  remote.Table2.Edit;
  remote.Table2value.asString:='open';
  remote.Table2.Post;

  winexec('D:\Sun_master\master\P_master_po',SW_Show);

      end;

end;


procedure clickmeter;
var test:string[15];
begin

 remote.Table3.FindNearest(['power']);
      test:= remote.Table3data.asString;

       remote.Table3.FindNearest(['power']);
      test:= remote.Table3data.asString;
```

```
 remote.Table3.FindNearest(['power']);
 test:= remote.Table3data.asString;

 if test='on'  then
 begin
remote.Table3.FindNearest(['power']);
remote.Table3.Edit;
remote.Table3data.asString:='open';
remote.Table3.Post;

 remote.Table3.FindNearest(['power']);
remote.Table3.Edit;
remote.Table3data.asString:='open';
remote.Table3.Post;

 remote.Table3.FindNearest(['power']);
remote.Table3.Edit;
remote.Table3data.asString:='open';
remote.Table3.Post;

 remote.Table3.FindNearest(['power']);
remote.Table3.Edit;
remote.Table3data.asString:='open';
remote.Table3.Post;
```

```
  winexec('d:\Sun_client\dataproject\master\master_GPIB\P_master_meter',SW_Show);

      end;

end;

procedure Tremote.Timer1Timer(Sender: TObject);
begin
remote.Table1.Refresh;
remote.Table1.Refresh;
remote.Table1.Refresh;
remote.Table1.Refresh;
remote.Table1.Refresh;
remote.Table1.Refresh;
remote.Table1.Refresh;
remote.Table1.Refresh;

clickgenerator;


remote.Table2.Refresh;
remote.Table2.Refresh;
remote.Table2.Refresh;
remote.Table2.Refresh;
remote.Table2.Refresh;
remote.Table2.Refresh;
```

```
remote.Table2.Refresh;
remote.Table2.Refresh;
remote.Table2.Refresh;
clickpower;
remote.Table3.Refresh;
remote.Table3.Refresh;
remote.Table3.Refresh;
remote.Table3.Refresh;
remote.Table3.Refresh;
remote.Table3.Refresh;
remote.Table3.Refresh;
remote.Table3.Refresh;
remote.Table3.Refresh;
remote.Table3.Refresh;
clickmeter;

end;

procedure Tremote.FormShow(Sender: TObject);
begin
Timer1.Enabled:=True;
Table1.Active:=true;
Table2.Active:=true;
Table3.Active:=true;
Table4.Active:=true;
end;

procedure Tremote.Shape1MouseDown(Sender: TObject; Button: TMouseButton;
```

```
  Shift: TShiftState; X, Y: Integer);
begin
    Button5.click;
end;

procedure Tremote.Shape2MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
 Button5.click;
end;

end.
```

****

program master generater

****

```
unit Pmaster_gen;

interface

uses
```

```
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls, ComCtrls, StdCtrls, Db, DBTables;

type
  TGENERATOR = class(TForm)
    Image1: TImage;
    Shape2: TShape;
    Shape1: TShape;
    Shape3: TShape;
    Shape4: TShape;
    Shape5: TShape;
    Shape6: TShape;
    Shape7: TShape;
    Shape8: TShape;
    Shape9: TShape;
    Shape10: TShape;
    Shape11: TShape;
    Shape12: TShape;
    Shape13: TShape;
    Shape14: TShape;
    Shape15: TShape;
    Shape16: TShape;
    Label1: TLabel;
    Label2: TLabel;
    Label3: TLabel;
    Label4: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
```

```
  UpDown1: TUpDown;
  UpDown2: TUpDown;
  Timer1: TTimer;
  DataSource1: TDataSource;
  Table1: TTable;
  Table1name: TStringField;
  Table1value: TStringField;
  Table1type: TStringField;
  Database1: TDatabase;
  procedure Shape1MouseDown(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);
  procedure FormShow(Sender: TObject);
  procedure Timer1Timer(Sender: TObject);

private
  { Private declarations }
public
  { Public declarations }
end;

type
TReceive = procedure (boardID:    integer;
                      address:    word;
                      var buffer;
                      cnt:        longint;
                      termination: integer); stdcall;
 TSend = procedure (boardID: integer;
                    address: word;
```

```
                var buffer;
                datacnt: longint;
                eotmode: integer); stdcall;

 TSendIFC = procedure (boardID: integer); stdcall;
 Tibclr   = function (ud:integer):integer;stdcall;
 Tibonl   = function(ud:integer;v:integer):integer;stdcall;
 Tibwrt   = function(ud:integer;var wrtbuf;
              cnt:longint):integer;stdcall;
 Tibdev   = function(ud:integer;
              pad:integer;
              sad:integer;
              tmo:integer;
              eot:integer;
              eos:integer):integer;stdcall;
 Tibrd    = function(ud:integer;
              var rdbuf;
              cnt:Longint):integer;stdcall;


var
 GENERATOR: TGENERATOR;
 Gpib32lib:Thandle;
 Receive: TReceive;
 Send: TSend;
 SendIFC: TSendIFC;
 ibclr:Tibclr;
 ibwrt:Tibwrt;
```

```
  ibonl:Tibonl;
  ibdev:Tibdev;
  ibrd:Tibrd;
  dvm:integer;
  num:integer;
  buf:packed array[0..100] of char;

implementation

{$R *.DFM}

procedure loadDLL;
VAR
  str:string;
  begin
  Gpib32Lib:=LoadLibrary('GPIB-32.DLL');
  If Gpib32Lib = 0 then
    begin
     str:='LoadLibary FAILED!';
     MessageDlg(str,mtError,[mbOK],0);
     halt;
    end;
  @Receive       := GetProcAddress(Gpib32Lib, 'Receive');
  @Send          := GetProcAddress(Gpib32Lib, 'Send');
  @SendIFC       := GetProcAddress(Gpib32Lib, 'SendIFC');
  @ibclr       :=getProcAddress(Gpib32Lib,'ibclr');
  @ibonl       :=GetProcAddress(Gpib32Lib,'ibonl');
  @ibwrt       :=GetProcAddress(Gpib32Lib,'ibwrt');
```

```
  @ibdev       :=GetProcAddress(Gpib32Lib,'ibdev');
  @ibrd        :=GetProcAddress(Gpib32Lib,'ibrd');
  If (@Receive       = NIL) Or
    (@Send           = NIL) Or
    (@SendIFC        = NIL) Or
    (@ibclr        = NIL) or
    (@ibonl        = NIL) or
    (@ibdev        = NIL) or
    (@ibrd         = NIL) or
    (@ibwrt        = NIL) Then
    begin
     str:='GetProcAddress FAIL!';
     MessageDlg(str,mtError,[mbOK],0);
     FreeLibrary(Gpib32Lib);
    end;
  end;


procedure TGENERATOR.Shape1MouseDown(Sender: TObject; Button: TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
ibonl(0,0);
FreeLibrary(Gpib32Lib);
close;
end;


procedure TGENERATOR.FormShow(Sender: TObject);
```

```
begin
 GENERATOR.Table1.Active:=true;
  GENERATOR.Table1.FindNearest(['power']);
  GENERATOR.Table1.Edit;
  GENERATOR.Table1type.asString:='open';
  GENERATOR.Table1.Post;

  GENERATOR.shape2.brush.color := clGreen;
  GENERATOR.shape3.brush.color := clGreen;
Timer1.Enabled:=True;
LoadDLL;
sendifc(0);
end;

procedure rmp;

begin

    GENERATOR.shape11.Brush.color:=clLime;
    GENERATOR.shape12.Brush.color:=clGreen;
    GENERATOR.shape13.Brush.color:=clGreen;
    GENERATOR.shape14.Brush.color:=clGreen;
    GENERATOR.shape15.Brush.color:=clGreen;
    GENERATOR.shape16.Brush.color:=clGreen;


  dvm:=ibdev(0,12,0,15,1,0);
  buf:= 'rmp ';
```

```
  ibwrt(dvm,buf,4);
end;

procedure tri;

begin

    GENERATOR.shape12.Brush.color:=clLime;
    GENERATOR.shape11.Brush.color:=clGreen;
    GENERATOR.shape13.Brush.color:=clGreen;
    GENERATOR.shape14.Brush.color:=clGreen;
    GENERATOR.shape15.Brush.color:=clGreen;
    GENERATOR.shape16.Brush.color:=clGreen;


  dvm:=ibdev(0,12,0,15,1,0);
  buf:= 'tri ';
  ibwrt(dvm,buf,4);

end;

procedure sin;

begin

    GENERATOR.shape13.Brush.color:=clLime;
    GENERATOR.shape12.Brush.color:=clGreen;
    GENERATOR.shape11.Brush.color:=clGreen;
```

```
    GENERATOR.shape14.Brush.color:=clGreen;
    GENERATOR.shape15.Brush.color:=clGreen;
    GENERATOR.shape16.Brush.color:=clGreen;
  dvm:=ibdev(0,12,0,15,1,0);
  buf:= 'sin ';
  ibwrt(dvm,buf,4);

end;

procedure sqr;

begin

    GENERATOR.shape14.Brush.color:=clLime;
    GENERATOR.shape12.Brush.color:=clGreen;
    GENERATOR.shape13.Brush.color:=clGreen;
    GENERATOR.shape11.Brush.color:=clGreen;
    GENERATOR.shape15.Brush.color:=clGreen;
    GENERATOR.shape16.Brush.color:=clGreen;


  dvm:=ibdev(0,12,0,15,1,0);
  buf:= 'sqr ';
  ibwrt(dvm,buf,4);

end;
```

```
procedure pls;
begin
    GENERATOR.shape15.Brush.color:=clLime;
    GENERATOR.shape12.Brush.color:=clGreen;
    GENERATOR.shape13.Brush.color:=clGreen;
    GENERATOR.shape11.Brush.color:=clGreen;
    GENERATOR.shape14.Brush.color:=clGreen;
    GENERATOR.shape16.Brush.color:=clGreen;
  dvm:=ibdev(0,12,0,15,1,0);
  buf:= 'pls ';
  ibwrt(dvm,buf,4);
end;

procedure arb;
begin
    GENERATOR.shape16.Brush.color:=clLime;
    GENERATOR.shape12.Brush.color:=clGreen;
    GENERATOR.shape13.Brush.color:=clGreen;
    GENERATOR.shape14.Brush.color:=clGreen;
    GENERATOR.shape15.Brush.color:=clGreen;
    GENERATOR.shape11.Brush.color:=clGreen;
  dvm:=ibdev(0,12,0,15,1,0);
  buf:= 'arb ';
```

```
  ibwrt(dvm,buf,4);
end;



procedure ReadFreq;
var
  str:string;
begin
    dvm := ibdev(0,12,0,15,1,0);
    buf := 'frq? ';
    send(0,12,buf,20,1);
    receive(0,12,buf,20,1);
    case  buf[13] of
        '0' : GENERATOR.Label2.Caption :='Hz';
        '3' :  if buf[12]='-' then
                 begin
                  GENERATOR.Label2.caption :='mHz';
                 end
               else
                 begin
                  GENERATOR.Label2.caption :='KHz';
                 end;
        '6' : GENERATOR.Label2.caption :='MHz';
    end;
    str := buf;
```

```
    delete(str,1,4);
    delete(str,8,4);
    GENERATOR.label1.caption:= str;
end;
procedure ReadAmp;
var
  str1:string;
  num:real;
  rps:string;
  code:integer;
begin
    dvm := ibdev(0,12,0,15,1,0);
    buf:='amp? ';
    send(0,12,buf,20,1);
    receive(0,12,buf,20,1);
    case buf[11] of
       '0' : GENERATOR.label4.caption := 'V';
       '3' : GENERATOR.label4.caption := 'mV';
    end;
    str1 := buf;
     delete(str1,1,4);
     delete(str1,6,4);
       GENERATOR.Table1.FindNearest(['signal']);
        rps := GENERATOR.table1Type.asString;
            if rps = 'pls' then
            begin
              val(str1,num,code);
              num := num/2;
```

```
            str(num:6:2,str1);
          end;
    GENERATOR.label3.caption:= str1;
end;

Procedure CheckSignal;
var
  index:char;
  SignalType:string;
begin
  GENERATOR.Table1.FindNearest(['signal']);
  SignalType:= GENERATOR.Table1type.asString;
 If SignalType = 'rmp' then index := '1'
    else if SignalType = 'tri' then index := '2'
    else if SignalType = 'sin' then index := '3'
    else if SignalType = 'sqr' then index := '4'
    else if SignalType = 'pls' then index := '5'
    else if SignalType = 'arb' then index := '6'
    else index:= '0';
 case index of
    '1': rmp;
    '2': tri;
    '3': sin;
    '4': sqr;
    '5': pls;
    '6': arb;
 end;
END;
```

```
Procedure CheckUnit;
var
Output:string;
begin
   dvm:=ibdev(0,12,0,15,1,0);
   GENERATOR.Table1.FindNearest(['unit_f']);
   Output := GENERATOR.Table1value.asString;
   if Output = '1' then
     begin
        GENERATOR.shape6.Brush.color:=clLime;
        GENERATOR.shape7.Brush.color:=clGreen;
        GENERATOR.shape8.Brush.color:=clGreen;
     end
   else  if Output = '2' then
     begin
       GENERATOR.shape7.Brush.color:=clLime;
        GENERATOR.shape6.Brush.color:=clGreen;
        GENERATOR.shape8.Brush.color:=clGreen;
    end
    else  if Output = '3' then
     begin
       GENERATOR.shape8.Brush.color:=clLime;
        GENERATOR.shape6.Brush.color:=clGreen;
        GENERATOR.shape7.Brush.color:=clGreen;
    end;
     dvm:=ibdev(0,12,0,15,1,0);
   GENERATOR.Table1.FindNearest(['unit_v']);
```

```
  Output := GENERATOR.Table1value.asString;
  if Output = '1' then
    begin
       GENERATOR.shape9.Brush.color:=clLime;
       GENERATOR.shape10.Brush.color:=clGreen;
    end
  else  if Output = '2' then
    begin
      GENERATOR.shape10.Brush.color:=clLime;
       GENERATOR.shape9.Brush.color:=clGreen;
   end
   else  if Output = '3' then
    begin
      GENERATOR.shape8.Brush.color:=clLime;
       GENERATOR.shape6.Brush.color:=clGreen;
       GENERATOR.shape7.Brush.color:=clGreen;
   end;
END;

Procedure CheckOutput;
var
Output:string;
begin

  dvm:=ibdev(0,12,0,15,1,0);
  GENERATOR.Table1.FindNearest(['output']);
  Output := GENERATOR.Table1value.asString;
  if Output = '1' then
```

```
    begin
        GENERATOR.shape2.Brush.color:=clLime;
        buf:='ot1 ';
        ibwrt(dvm,buf,4);
    end
  else  if Output = '0' then
    begin
        GENERATOR.shape2.brush.color:=clGreen;
        buf:='ot0 ';
        ibwrt(dvm,buf,4);
   end;
END;
Procedure CheckStatus;
var
Status:string;
begin

  dvm:=ibdev(0,12,0,15,1,0);
  GENERATOR.Table1.FindNearest(['status']);
  Status := GENERATOR.Table1value.asString;
  if Status = '1' then
    begin
        ibclr(dvm);
        GENERATOR.shape7.brush.color := clLime;
         GENERATOR.shape13.brush.color := clLime;
          GENERATOR.shape10.brush.color := clLime;
          GENERATOR.shape3.brush.color := clLime;
           GENERATOR.shape5.brush.color := clGreen;
```

```
    end;
END;

procedure FreSet;
var
  freq:string[15];
  i: integer;
  freq1:string[15];
begin
  if GENERATOR.shape4.brush.color = clGreen then
      begin
        GENERATOR.shape4.Brush.color:=clLime;
        GENERATOR.shape5.Brush.color:=clGreen;
      end;
  GENERATOR.Table1.FindNearest(['freq']);
  GENERATOR.edit1.text := GENERATOR.table1Value.asString;
  dvm := ibdev(0,12,0,15,1,0);


 if GENERATOR.edit1.Text <> '' then
   begin
      freq:=GENERATOR.edit1.Text;
      buf := 'frq:';
      for i := 1 to Length(freq) do
      buf[3 + i ] :=freq[i];
      buf[4 + Length(freq)] := ';';
      send(0,12,buf,20,1);
      buf :=  '';
```

```
        ReadFreq;
    end;
end;

procedure voleSet;
var
    rps:string;
  amp:string[15];
  i: integer;
  amp1:string[15];
  amp2:string[15];
   num:real;
   num1:real;
   num3:real;
  code:integer;
begin
  if GENERATOR.shape5.brush.color = clGreen then
      begin
         GENERATOR.shape5.Brush.color:=clLime;
         GENERATOR.shape4.Brush.color:=clGreen;
      end;
  GENERATOR.Table1.FindNearest(['amp']);
  GENERATOR.edit2.text := GENERATOR.table1Value.asString;
  dvm := ibdev(0,12,0,15,1,0);
  if GENERATOR.edit2.Text <> '' then
    begin
       amp:=GENERATOR.edit2.Text;
       val(amp,num3,code);
```

```
      if  num3 < 1 then
       begin
         str(num3:5:2,amp2);
         for i:=1 to 5 do
         amp[i]:=amp2[i];
       end;

        GENERATOR.Table1.FindNearest(['signal']);
        rps := GENERATOR.table1Type.asString;

            if rps = 'pls' then
            begin
              val(amp,num,code);
              num := num*2;
              str(num:5:2,amp);
                        end;

        buf := 'amp:';
        for i := 1 to Length(amp) do
        buf[3 + i ] :=amp[i];
        buf[4 + Length(amp)] := ';';
        send(0,12,buf,20,1);
        buf :=  '';
        ReadAmp;
    end;
end;
```

```
procedure TGENERATOR.Timer1Timer(Sender: TObject);
var test:string;
begin
    Table1.Refresh;
    GENERATOR.Table1.FindNearest(['power']);
     test:= GENERATOR.Table1type.asString;
     if test='off'  then
   begin
    // ibonl(0,0);
    //   GENERATOR.shape6.Brush.color:=clLime;
     //    GENERATOR.shape7.Brush.color:=clGreen;
     //    GENERATOR.shape8.Brush.color:=clGreen;

  //FreeLibrary(Gpib32Lib);
   // GENERATOR.shape6.Brush.color:=clLime;
    //     GENERATOR.shape7.Brush.color:=clGreen;
     //    GENERATOR.shape8.Brush.color:=clGreen;

  close;
   end;
 Table1.Refresh;
  Table1.FindNearest(['freq']);
  GENERATOR.edit1.Text := GENERATOR.table1value.asString;
  FreSet;

    Table1.Refresh;
    Table1.FindNearest(['amp']);
    GENERATOR.edit2.Text := GENERATOR.table1value.asString;
```

```
    voleSet;
    CheckSignal;
    CheckOutput;
    CheckStatus;
    CheckUnit;
end;
end.
```

****

program master multimeter

****

```
unit Pmaster_meter;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ExtCtrls, Buttons, Db, DBTables;

type
  TMULTIMETER = class(TForm)
    Image1: TImage;
    Shape1: TShape;
    Shape2: TShape;
    Shape3: TShape;
    Shape4: TShape;
    Shape5: TShape;
    Shape6: TShape;
    Shape7: TShape;
    Label1: TLabel;
    Label2: TLabel;
    SpeedButton1: TSpeedButton;
    SpeedButton2: TSpeedButton;
```

```
  SpeedButton3: TSpeedButton;
  SpeedButton4: TSpeedButton;
  SpeedButton5: TSpeedButton;
  SpeedButton6: TSpeedButton;
  Timer1: TTimer;
  DataSource1: TDataSource;
  Table1: TTable;
  Table1information: TStringField;
  Table1data: TStringField;
  Database1: TDatabase;
  procedure SpeedButton1Click(Sender: TObject);
  procedure Timer1Timer(Sender: TObject);
  procedure FormShow(Sender: TObject);
  procedure Shape1MouseDown(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);
  procedure SpeedButton2Click(Sender: TObject);
  procedure SpeedButton3Click(Sender: TObject);
  procedure SpeedButton4Click(Sender: TObject);
  procedure SpeedButton5Click(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;

const
ATN    =  $10;        (* Attention asserted                *)
TACS   =  $8;         (* Talker active                  const
```

```
(* GPIB status bit definitions.                    *)
ERR    = $8000;      (* Error detected             *)
TIMO   = $4000;      (* Timeout                    *)
ENDgpib = $2000;     (* EOI or EOS detected        *)
SRQI   = $1000;      (* SRQ detected by CIC        *)
RQS    = $800;       (* Device needs service       *)
SPOLL  = $400;       (* Board has been serially polled *)
EVENT  = $200;       (* An event has occurred      *)
CMPL   = $100;       (* I/O completed              *)
LOK    = $80;        (* Local lockout state        *)
REM    = $40;        (* Remote state               *)
CIC    = $20;        (* Controller-in-charge       *)
LACS   = $4;         (* Listener active            *)
DTAS   = $2;         (* Device trigger state       *)
DCAS   = $1;         (* Device clear state         *)

(* Error messages returned in global variable IBERR:  *)
EDVR = 0;            (* System error               *)
ECIC = 1;            (* Function requires GPIB board to be CIC *)
ENOL = 2;            (* Write function detected no Listeners *)
EADR = 3;            (* Interface board not addressed correctly *)
EARG = 4;            (* Invalid argument to function call *)
ESAC = 5;            (* Function requires GPIB board to be SAC *)
EABO = 6;            (* I/O operation aborted      *)
ENEB = 7;            (* Non-existent interface board *)
EDMA = 8;            (* Error performing DMA       *)
EOIP = 10;           (* I/O operation started before previous *)
                     (* operation completed        *)
```

```
ECAP = 11;          (* No capability for intended operation    *)
EFSO = 12;          (* File system operation error              *)
EBUS = 14;          (* Command error during device call        *)
ESTB = 15;          (* Serial poll status byte lost            *)
ESRQ = 16;          (* SRQ remains asserted                    *)
ETAB = 20;          (* The return buffer is full               *)

(* Values for the 488.2 Send command.                          *)
NULLend = $00;      (* Do nothing at the end of a transfer.    *)
NLend   = $01;      (* Send NL with EOI after a transfer.      *)
DABend  = $02;      (* Send EOI with the last DAB.             *)
STOPend = $100;     (* Value used by the 488.2 Receive command. *)
MAVbit  = $10;      (* Position of the Message Available bit.  *)

(* Timeout values and meanings:                                *)
TNONE  = 0;         (* Infinite timeout (disabled)             *)
T10us  = 1;         (* Timeout of 10 microseconds (ideal)      *)
T30us  = 2;         (* Timeout of 30 microseconds (ideal)      *)
T100us = 3;         (* Timeout of 100 microseconds (ideal)     *)
T300us = 4;         (* Timeout of 300 microseconds (ideal)     *)
T1ms   = 5;         (* Timeout of 1 milliseconds (ideal)       *)
T3ms   = 6;         (* Timeout of 3 milliseconds (ideal)       *)
T10ms  = 7;         (* Timeout of 10 milliseconds (ideal)      *)
T30ms  = 8;         (* Timeout of 30 milliseconds (ideal)      *)
T100ms = 9;         (* Timeout of 100 milliseconds (ideal)     *)
T300ms = 10;        (* Timeout of 300 milliseconds (ideal)     *)
T1s    = 11;        (* Timeout of 1 second (ideal)             *)
T3s    = 12;        (* Timeout of 3 seconds (ideal)            *)
```

```
T10s   = 13;          (* Timeout of 10 seconds (ideal)        *)
T30s   = 14;          (* Timeout of 30 seconds (ideal)        *)
T100s  = 15;          (* Timeout of 100 seconds (ideal)       *)
T300s  = 16;          (* Timeout of 300 seconds (ideal)       *)
T1000s = 17;          (* Timeout of 1000 seconds (ideal)      *)

type
(* Type declarations for exported NI-488.2 Global Variables.          *)
Tibsta  = function : integer; stdcall;
Tiberr  = function : integer; stdcall;
Tibcntl = function : Longint; stdcall;

(* Type declarations for exported NI-488.2 routines.                  *)
Tibclr = function(ud : integer): integer; stdcall;

Tibdev = function(ud : integer;
                  pad : integer;
                  sad : integer;
                  tmo : integer;
                  eot : integer;
                  eos : integer): integer;stdcall;

Tibrd = function(ud : integer;
                 var rdbuf;
                 cnt  : longint) : integer; stdcall;

Tibtrg = function(ud : integer) : integer; stdcall;
```

```
Tibwrt = function(ud : integer;
            var wrtbuf;
            cnt : longint): integer;stdcall;

Tibwait = function(ud : integer;
            mask: integer): integer; stdcall;
TDevClear = procedure (boardID: integer;
                        address: word); stdcall;

Tibonl = function (ud: integer;
            v:  integer) : integer; stdcall;

TRcvRespMsg = procedure ( boardID: integer;
                var buffer;
                cnt:  longint;
                termination : integer); stdcall;
TReceiveSetup = procedure (boardID : integer;
                address : word ); stdcall;

TReceive = procedure (boardID:    integer;
                address:    word;
                var buffer;
                cnt:        longint;
                termination: integer); stdcall;

TSendSetup = procedure (boardID: integer;
                address:  word); stdcall;
```

```
  TSendDataBytes = procedure (boardID: integer;
                              var buffer;
                              datacount: longint;
                              eotmode: integer); stdcall;

  TSend = procedure (boardID: integer;
                     address: word;
                     var buffer;
                     datacnt: longint;
                     eotmode: integer); stdcall;

  TSendIFC = procedure (boardID: integer); stdcall;

var

  MULTIMETER:TMULTIMETER;
   (* NI-488.2 GPIB global status variables                    *)
  AddrIbsta  : Tibsta;
  AddrIberr  : Tiberr;
  AddrIbcntl : Tibcntl;

(* Pointers to the NI-488.2 GPIB global status variables      *)
  Pibsta : ^integer;
  Piberr : ^integer;
  Pibcntl : ^Longint;

(* Declaration for the Handle for the GPIB library            *)
```

```
  Gpib32Lib: THandle;

(* Declarations for the NI-488.2 GPIB calls                    *)
  ibclr : Tibclr;
  ibdev : Tibdev;
  ibrd  : Tibrd;
  ibtrg : Tibtrg;
  ibwait: Tibwait;
  ibwrt : Tibwrt;
  DevClear: TDevClear;
  Receive: TReceive;
  ReceiveSetup: TReceiveSetup;
  RcvRespMsg: TRcvRespMsg;
  SendDataBytes: TSendDataBytes;
  SendSetup: TSendSetup;
  Send: TSend;
  SendIFC: TSendIFC;

(* Declaration for the NI-488 GPIB call                        *)
  ibonl: Tibonl;
  dvm : integer;  {* device Number *}
  index :integer;
  bufn: string;
implementation

{$R *.DFM}

  procedure loadDLL;
```

```
var
 str : string;

 begin
 (* Load the GPIB-32.DLL library using the LoadLibrary function.       *)
  Gpib32Lib := LoadLibrary('GPIB-32.DLL');

   If Gpib32Lib = 0 Then
    Begin
     str := 'LoadLibrary FAILED!';
     MessageDlg(str, mtError, [mbOK], 0);
     halt;
    End;
 (* Get the addresses of the GPIB Global Variables.                    *)
  @AddrIbsta  := GetProcAddress(Gpib32Lib, 'user_ibsta');
  @AddrIberr  := GetProcAddress(Gpib32Lib, 'user_iberr');
  @AddrIbcntl := GetProcAddress(Gpib32Lib, 'user_ibcnt');

 (* Get the addresses of the functions needed for this application.    *)
  @ibclr       := GetProcAddress(Gpib32Lib, 'ibclr');
  @ibdev       := GetProcAddress(Gpib32Lib, 'ibdev');
  @ibrd        := GetProcAddress(Gpib32Lib, 'ibrd');
  @ibtrg       := GetProcAddress(Gpib32Lib, 'ibtrg');
  @ibwait      := GetProcAddress(Gpib32Lib, 'ibwait');
  @ibwrt       := GetProcAddress(Gpib32Lib, 'ibwrt');
  @DevClear    := GetProcAddress(Gpib32Lib, 'DevClear');
  @ibonl       := GetProcAddress(Gpib32Lib, 'ibonl');
  @RcvRespMsg  := GetProcAddress(Gpib32Lib, 'RcvRespMsg');
```

```
@ReceiveSetup   := GetProcAddress(Gpib32Lib, 'ReceiveSetup');
@Receive        := GetProcAddress(Gpib32Lib, 'Receive');
@SendSetup      := GetProcAddress(Gpib32Lib, 'SendSetup');
@SendDataBytes  := GetProcAddress(Gpib32Lib, 'SendDataBytes');
@Send           := GetProcAddress(Gpib32Lib, 'Send');
@SendIFC        := GetProcAddress(Gpib32Lib, 'SendIFC');

if (@ibdev        = NIL) or
  (@ibclr         = NIL) or
  (@ibrd          = NIL) or
  (@ibtrg         = NIL) or
  (@ibwait        = NIL) or
  (@ibwrt         = NIL) or
  (@AddrIbsta     = NIL) Or
  (@AddrIberr     = NIL) Or
  (@AddrIbcntl    = NIL) Or
  (@DevClear      = NIL) Or
  (@ibonl         = NIL) Or
  (@RcvRespMsg    = NIL) or
  (@ReceiveSetup  = NIL) or
  (@Receive       = NIL) Or
  (@SendDataBytes = NIL) or
  (@SendSetup     = NIL) or
  (@Send          = NIL) Or
  (@SendIFC       = NIL) Then
  Begin
   str := 'GetProcAddresses FAILED!';
   MessageDlg(str, mtError, [mbOK], 0);
```

```
        FreeLibrary(Gpib32Lib);
        halt;
      End;

 (* Initialize GPIB global pointers to point to address location      *)
      Pibsta := @AddrIbsta;
      Piberr := @AddrIberr;
      Pibcntl := @AddrIbcntl;
  end;
function result_v(msg:string;n:integer):string;
var  dat:real;i:integer;
begin    if msg[1]<>'E' then begin
           val(msg,dat,i);
           str(dat:7:n,msg);end else begin msg:='ERROR 01';END;
           multimeter.table1.FindNearest(['read']);
           multimeter.table1.Edit;
           multimeter.table1data.AsString:=msg;
           multimeter.table1.Post;
           result_v:=msg;
end;
function result_i(msg:string;n:integer):string;
var  dat:real;i:integer;
begin
       if msg[1]<>'E' then begin
          val(msg,dat,i);
          dat:=dat*1000;
          str(dat:7:n,msg);end else begin msg:='ERROR 01';END;
          multimeter.table1.FindNearest(['read']);
```

```
        multimeter.table1.Edit;
        multimeter.table1data.AsString:=msg;
        multimeter.table1.Post;
        result_i:=msg;
end;
function result_ohm(msg:string;n:integer):string;
var  DAT:REAL;I:INTEGER;
begin
      if msg[1]<>'E' then begin
        val(msg,dat,i);
        dat:=dat*0.001;
        str(dat:7:n,msg);end else begin msg:='ERROR 01';END;
        multimeter.table1.FindNearest(['read']);
        multimeter.table1.Edit;
        multimeter.table1data.AsString:=msg;
        multimeter.table1.Post;
        result_ohm:=msg;
end;



procedure autorange(sett:integer);
var  buf:packed array[0..32]of char;
   msg:string;
begin
    case sett of
    1:begin
```

```
        multimeter.shape2.Brush.color:=clLime;
        multimeter.shape3.brush.color:=clGreen;
        multimeter.shape4.brush.color:=clGreen;
        multimeter.shape5.brush.color:=clGreen;
        multimeter.shape6.brush.color:=clGreen;
        buf:='A1;VD;';      //VDC
        ibwrt(dvm,buf,10);
        buf:='T3;S1;';
        ibwrt(dvm,buf,10);
        ibrd(dvm,buf,12);
        msg:=buf;
        MULTIMETER.label1.caption:=result_v(msg,7);
        MULTIMETER.label2.caption:='V';
        end ;
2:begin
        multimeter.shape2.Brush.color:=clGreen;
        multimeter.shape3.brush.color:=clLime;
        multimeter.shape4.brush.color:=clGreen;
        multimeter.shape5.brush.color:=clGreen;
        multimeter.shape6.brush.color:=clGreen;
        buf:='A1;VA;';          // VAC
        ibwrt(dvm,buf,12);
        buf:='T3;S1;';
        ibwrt(dvm,buf,10);
        ibrd(dvm,buf,12);
        msg:=buf;
        MULTIMETER.label1.caption:=result_v(msg,7);
        MULTIMETER.Label2.caption:='V';
```

```
        end;
3:begin
        multimeter.shape2.Brush.color:=clGreen;
        multimeter.shape3.brush.color:=clGreen;
        multimeter.shape4.brush.color:=clLime;
        multimeter.shape5.brush.color:=clGreen;
        multimeter.shape6.brush.color:=clGreen;
        buf:='A1;ID;';          //Idc
        ibwrt(dvm,buf,10);
        buf:='T3;S1;';
        ibwrt(dvm,buf,10);
        ibrd(dvm,buf,12);
        msg:=buf;
        MULTIMETER.label1.caption:=result_i(msg,7);
        MULTIMETER.label2.caption:='mA';
        end;
4:begin
        multimeter.shape2.Brush.color:=clGreen;
        multimeter.shape3.brush.color:=clGreen;
        multimeter.shape4.brush.color:=clGreen;
        multimeter.shape5.brush.color:=clLime;
        multimeter.shape6.brush.color:=clGreen;
        buf:='A1;IA;';      // Iac
        ibwrt(dvm,buf,12);
        buf:='T3;S1;';
        ibwrt(dvm,buf,10);
        ibrd(dvm,buf,12);
        msg:=buf;
```

```
            MULTIMETER.label1.caption:=result_i(msg,7);
            MULTIMETER.label2.caption:='mA';
            end ;
      5:begin
            multimeter.shape2.Brush.color:=clGreen;
            multimeter.shape3.brush.color:=clGreen;
            multimeter.shape4.brush.color:=clGreen;
            multimeter.shape5.brush.color:=clGreen;
            multimeter.shape6.brush.color:=clLime;
            buf:='A1;O2;';//OHM
            ibwrt(dvm,buf,15);
            buf:='T3;S1;';
            ibwrt(dvm,buf,10);
            ibrd(dvm,buf,12);
            msg:=buf;
            MULTIMETER.label1.caption:=result_ohm(msg,7);
            MULTIMETER.label2.caption:='Kohm';
            end ;

   end;
end;
procedure checktable ;
var data :string;
begin
    multimeter.Table1.FindNearest(['range']);
    data:=multimeter.Table1data.AsString;
  //  multimeter.Table1.FindNearest(['range']);
  //  bufn:=multimeter.Table1data.AsString;
```

```
 bufn := data;
if bufn[1] <> ' ' then begin
if ((bufn[1]='A')and(bufn[2]='1')) THEN
 begin multimeter.shape7.Brush.color:=clLime;
     if ((bufn[7]='V')and(bufn[8]='D')) then
       autorange(1);
     if ((bufn[7]='V')and(bufn[8]='A')) then
       autorange(2);
     if ((bufn[7]='I')and(bufn[8]='D'))then
       autorange(3);
     if ((bufn[7]='I')and(bufn[8]='A'))then
       autorange(4);
     if ((bufn[7]='O')and(bufn[8]='2'))then
       autorange(5);
   end
else begin multimeter.shape7.Brush.color:=clGreen;
     if ((bufn[7]='V')and(bufn[8]='D')) then
       multimeter.SpeedButton1.Click;
     if ((bufn[7]='V')and(bufn[8]='A')) then
       multimeter.SpeedButton1.Click;
     if ((bufn[7]='I')and(bufn[8]='D')) then
       multimeter.SpeedButton1.Click;
     if ((bufn[7]='I')and(bufn[8]='A')) then
       multimeter.SpeedButton1.Click;
     if ((bufn[7]='O')and(bufn[8]='2')) then
       multimeter.SpeedButton1.Click;
   end;
 end
```

```
    else
end;
procedure vdcRange(ra:integer);
var buf:packed array[0..15]of char; msg:string;
begin  case ra of
   1: begin buf:='A0;T3;VD;';  // vdc
         ibwrt(dvm,buf,10);
         buf:='S1;R1;';
         ibwrt(dvm,buf,10);
         ibrd(dvm,buf,12);
         msg:=buf;
         multimeter.label1.caption:=result_v(msg,7);
         multimeter.label2.caption:='V';
      end;
     2:begin buf:='A0;T3;VD;';  // vdc
         ibwrt(dvm,buf,10);
         buf:='S1;R2;';
         ibwrt(dvm,buf,10);
         ibrd(dvm,buf,12);
         msg:=buf;
         multimeter.label1.caption:=result_v(msg,6);
         multimeter.label2.caption:='V';
       end;
   3: begin buf:='A0;T3;VD;';  // vdc
         ibwrt(dvm,buf,10);
         buf:='S1;R3;';
         ibwrt(dvm,buf,10);
         ibrd(dvm,buf,12);
```

```
        msg:=buf;
        multimeter.label1.caption:=result_v(msg,5);
        multimeter.label2.caption:='V';
    end;
    4: begin buf:='A0;T3;VD;';  // vdc
        ibwrt(dvm,buf,10);
        buf:='S1;R4;';
        ibwrt(dvm,buf,10);
        ibrd(dvm,buf,12);
        msg:=buf;
        multimeter.label1.caption:=result_v(msg,4);
        multimeter.label2.caption:='V';
     end;
    5: begin buf:='A0;T3;VD;';  // vdc
        ibwrt(dvm,buf,10);
        buf:='S1;R5;';
        ibwrt(dvm,buf,10);
        ibrd(dvm,buf,12);
        msg:=buf;
        multimeter.label1.caption:=result_v(msg,3);
        multimeter.label2.caption:='V';
      end;
     end;
end;
procedure vacRange(ra:integer);
var buf:packed array[0..15]of char; msg:string;
begin  case ra of
   1: begin buf:='A0;T3;VA;';  // VAC
```

```
        ibwrt(dvm,buf,10);
        buf:='S1;R1;';
        ibwrt(dvm,buf,10);
        ibrd(dvm,buf,12);
        msg:=buf;
        multimeter.label1.caption:=result_v(msg,7);
        multimeter.label2.caption:='V';
    end;
   2:begin buf:='A0;T3;VA;';  // VAC
        ibwrt(dvm,buf,10);
        buf:='S1;R2;';
        ibwrt(dvm,buf,10);
        ibrd(dvm,buf,12);
        msg:=buf;
        multimeter.label1.caption:=result_v(msg,6);
        multimeter.label2.caption:='V';
    end;
  3: begin buf:='A0;T3;VA;';  // VAC
        ibwrt(dvm,buf,10);
        buf:='S1;R3;';
        ibwrt(dvm,buf,10);
        ibrd(dvm,buf,12);
        msg:=buf;
        multimeter.label1.caption:=result_v(msg,5);
        multimeter.label2.caption:='V';
    end;
   4: begin buf:='A0;T3;VA;';  // VAC
        ibwrt(dvm,buf,10);
```

```
        buf:='S1;R4;';
        ibwrt(dvm,buf,10);
        ibrd(dvm,buf,12);
        msg:=buf;
        multimeter.label1.caption:=result_v(msg,4);
        multimeter.label2.caption:='V';
    end;

5: begin buf:='A0;T3;VA;';  // VAC
        ibwrt(dvm,buf,10);
        buf:='S1;R5;';
        ibwrt(dvm,buf,10);
        ibrd(dvm,buf,12);
        msg:=buf;
        multimeter.label1.caption:=result_v(msg,3);
        multimeter.label2.caption:='V';
     end;
    end;
end;
procedure idcRange(ra:integer);
var buf:packed array[0..15]of char; msg:string;
begin  case ra of
   1: begin buf:='A0;T3;ID;';  // IDC
        ibwrt(dvm,buf,10);
        buf:='S1;R2;';
        ibwrt(dvm,buf,10);
        ibrd(dvm,buf,12);
        msg:=buf;
```

```
        multimeter.label1.caption:=result_i(msg,5);
        multimeter.label2.caption:='mA';
     end;
    2:begin buf:='A0;T3;ID;';  // IDC
        ibwrt(dvm,buf,10);
        buf:='S1;R5;';
        ibwrt(dvm,buf,10);
        ibrd(dvm,buf,12);
        msg:=buf;
        multimeter.label1.caption:=result_i(msg,2);
        multimeter.label2.caption:='mA';
     end;
  end;
end;
procedure iacRange(ra:integer);
var buf:packed array[0..15]of char; msg:string;
begin  case ra of
   1: begin buf:='A0;T3;IA;';  // IAC
        ibwrt(dvm,buf,10);
        buf:='S1;R2;';
        ibwrt(dvm,buf,10);
        ibrd(dvm,buf,12);
        msg:=buf;
        multimeter.label1.caption:=result_i(msg,5);
        multimeter.label2.caption:='A';
     end;
    2:begin buf:='A0;T3;IA;';  // IAC
        ibwrt(dvm,buf,10);
```

```
        msg:=buf;
        multimeter.label1.caption:=result_ohm(msg,3);
        multimeter.label2.caption:='Kohm';
    end;
     6: begin buf:='A0;T3;02;';  // ohm
        ibwrt(dvm,buf,10);
        buf:='S1;R6;';
        ibwrt(dvm,buf,10);
        ibrd(dvm,buf,12);
        msg:=buf;
        multimeter.label1.caption:=result_ohm(msg,2);
        multimeter.label2.caption:='Kohm';
      end;

    end;
end;
procedure TMULTIMETER.Timer1Timer(Sender: TObject);
var test:string;
begin
  MULTIMETER.Table1.Refresh;
    MULTIMETER.Table1.FindNearest(['power']);
     test:= MULTIMETER.Table1data.asString;

     if test='off'  then
   begin

    // ibonl(0,0);
    //   GENERATOR.shape6.Brush.color:=clLime;
```

```
    //    GENERATOR.shape7.Brush.color:=clGreen;
    //    GENERATOR.shape8.Brush.color:=clGreen;

  //FreeLibrary(Gpib32Lib);
   // GENERATOR.shape6.Brush.color:=clLime;
    //    GENERATOR.shape7.Brush.color:=clGreen;
     //    GENERATOR.shape8.Brush.color:=clGreen;

  close;
   end;

     table1.Refresh;
      checktable;
end;



procedure TMULTIMETER.FormShow(Sender: TObject);
begin
MULTIMETER.Table1.Active:=true;
 MULTIMETER.Table1.FindNearest(['power']);
  MULTIMETER.Table1.Edit;
  MULTIMETER.Table1data.asString:='open';
  MULTIMETER.Table1.Post;

     loaddll;
     dvm:=ibdev(0,7,0,15,1,0);
```

```
    timer1.Enabled:=true;
end;

procedure TMULTIMETER.Shape1MouseDown(Sender: TObject;
  Button: TMouseButton; Shift: TShiftState; X, Y: Integer);
begin
ibonl(0,0);
FreeLibrary(Gpib32Lib);
close;
end;

procedure TMULTIMETER.SpeedButton1Click(Sender: TObject);
begin
    shape2.brush.color:=clLime;
    shape3.brush.color:=clGreen;
    shape4.brush.color:=clGreen;
    shape5.brush.color:=clGreen;
    shape6.brush.color:=clGreen;
      if bufn[14]='1' then
        vdcRange(1);
      if bufn[14]='2' then
        vdcRange(2);
      if bufn[14]='3' then
        vdcRange(3);
      if bufn[14]='4' then
        vdcRange(4);
      if bufn[14]='5' then
        vdcRange(5);
```

```
end;


procedure TMULTIMETER.SpeedButton2Click(Sender: TObject);
begin
     shape2.Brush.color:=clGreen;
     shape3.brush.color:=clLime;
     shape4.brush.color:=clGreen;
     shape5.brush.color:=clGreen;
     shape6.brush.color:=clGreen;
       if bufn[14]='1' then
         vacRange(1);
       if bufn[14]='2' then
         vacRange(2);
       if bufn[14]='3' then
         vacRange(3);
       if bufn[14]='4' then
         vacRange(4);
       if bufn[14]='5' then
         vacRange(5);

end;

procedure TMULTIMETER.SpeedButton3Click(Sender: TObject);
begin
  shape2.brush.color:=clGreen;
     shape3.brush.color:=clGreen;
     shape4.brush.color:=clGreen;
```

```
    shape5.brush.color:=clLime;
    shape6.brush.color:=clGreen;
     if bufn[14]='2' then
       iacRange(1);
     if bufn[14]='5' then
       iacRange(2);
end;

procedure TMULTIMETER.SpeedButton4Click(Sender: TObject);
begin
    shape2.brush.color:=clGreen;
    shape3.brush.color:=clGreen;
    shape4.brush.color:=clGreen;
    shape5.brush.color:=clLime;
    shape6.brush.color:=clGreen;
     if bufn[14]='2' then
       iacRange(1);
     if bufn[14]='5' then
       iacRange(2);
end;


procedure TMULTIMETER.SpeedButton5Click(Sender: TObject);
begin
    shape2.brush.color:=clGreen;
    shape3.brush.color:=clGreen;
```

```
    shape4.brush.color:=clGreen;
    shape5.brush.color:=clGreen;
    shape6.brush.color:=clLime;
     if bufn[14]='1' then
       ohmRange(1);
     if bufn[14]='2' then
       ohmRange(2);
     if bufn[14]='3' then
       ohmRange(3);
     if bufn[14]='4' then
       ohmRange(4);
     if bufn[14]='5' then
       ohmRange(5);
     if bufn[14]='6' then
       ohmRange(6);
end;

end.
```

****

program master powersupply

****

```
unit Pmaster_power;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls, ComCtrls, StdCtrls, Db, DBTables;

type
  TPOWERSUPPLY = class(TForm)
    Image1: TImage;
    Shape1: TShape;
    Shape2: TShape;
    Shape3: TShape;
    Shape4: TShape;
    Shape5: TShape;
    Shape6: TShape;
    Shape7: TShape;
    Shape8: TShape;
    Shape9: TShape;
    Shape10: TShape;
    Shape11: TShape;
```

```
  Edit1: TEdit;
  Edit2: TEdit;
  UpDown1: TUpDown;
  UpDown2: TUpDown;
  Timer1: TTimer;
  DataSource1: TDataSource;
  Label1: TLabel;
  Label2: TLabel;
  Label3: TLabel;
  Label4: TLabel;
  Label5: TLabel;
  Label6: TLabel;
  Label7: TLabel;
  Label8: TLabel;
  Label9: TLabel;
  Label10: TLabel;
  Label11: TLabel;
  Label12: TLabel;
  Table1: TTable;
  Table1name: TStringField;
  Table1value: TStringField;
  procedure Shape1MouseDown(Sender: TObject; Button: TMouseButton;
    Shift: TShiftState; X, Y: Integer);
  procedure FormShow(Sender: TObject);
  procedure Timer1Timer(Sender: TObject);
private
  { Private declarations }
public
```

```
  { Public declarations }
end;

type

  TReceive = procedure (boardID:    integer;
                        address:    word;
                        var buffer;
                        cnt:        longint;
                        termination: integer); stdcall;
 TSend = procedure (boardID: integer;
                    address: word;
                    var buffer;
                    datacnt: longint;
                    eotmode: integer); stdcall;

 TSendIFC = procedure (boardID: integer); stdcall;
Tibclr   = function (ud:integer):integer;stdcall;
Tibonl   = function(ud:integer;v:integer):integer;stdcall;
Tibwrt   = function(ud:integer;var wrtbuf;
              cnt:longint):integer;stdcall;
Tibdev   = function(ud:integer;
              pad:integer;
              sad:integer;
              tmo:integer;
              eot:integer;
              eos:integer):integer;stdcall;
Tibrd    = function(ud:integer;
```

```
                var rdbuf;
                cnt:Longint):integer;stdcall;
var
  POWERSUPPLY: TPOWERSUPPLY;
   Gpib32lib:Thandle;
  Receive: TReceive;
  Send: TSend;
  SendIFC: TSendIFC;
  ibclr:Tibclr;
  ibwrt:Tibwrt;
  ibonl:Tibonl;
  ibdev:Tibdev;
  ibrd:Tibrd;
  dvm:integer;
  num:integer;
  buf:packed array[0..100] of char;

implementation

{$R *.DFM}

 procedure loadDLL;
VAR
  str:string;
  begin
  Gpib32Lib:=LoadLibrary('GPIB-32.DLL');
  If Gpib32Lib = 0 then
     begin
```

```
   str:='LoadLibary FAILED!';
   MessageDlg(str,mtError,[mbOK],0);
   halt;
  end;
@Receive      := GetProcAddress(Gpib32Lib, 'Receive');
@Send         := GetProcAddress(Gpib32Lib, 'Send');
@SendIFC      := GetProcAddress(Gpib32Lib, 'SendIFC');
@ibclr       :=getProcAddress(Gpib32Lib,'ibclr');
@ibonl       :=GetProcAddress(Gpib32Lib,'ibonl');
@ibwrt       :=GetProcAddress(Gpib32Lib,'ibwrt');
@ibdev       :=GetProcAddress(Gpib32Lib,'ibdev');
@ibrd        :=GetProcAddress(Gpib32Lib,'ibrd');
If (@Receive      = NIL) Or
  (@Send         = NIL) Or
  (@SendIFC      = NIL) Or
  (@ibclr       = NIL) or
  (@ibonl       = NIL) or
  (@ibdev       = NIL) or
  (@ibrd        = NIL) or
  (@ibwrt       = NIL) Then
  begin
    str:='GetProcAddress FAIL!';
    MessageDlg(str,mtError,[mbOK],0);
    FreeLibrary(Gpib32Lib);
  end;
end;

procedure TPOWERSUPPLY.Shape1MouseDown(Sender: TObject; Button: TMouseButton;
```

```
  Shift: TShiftState; X, Y: Integer);
begin
ibonl(0,0);
FreeLibrary(Gpib32Lib);
close;
end;



procedure TPOWERSUPPLY.FormShow(Sender: TObject);
begin
        POWERSUPPLY.shape2.Brush.color:=clGreen;
        POWERSUPPLY.shape3.Brush.color:=clGreen;
        Timer1.Enabled := True;
        Loaddll;
end;

procedure ReadV1;
   var
    buffer : string[30];
    buf    : array[0..40] of char;
    msg    : string[20];
     i:integer;
    ii:real;
    code:integer;
   begin
   // LoadDLL;
```

```
    dvm:=ibdev(0,8,0,15,1,0);
    buf := 'mu1 ' ;
    send(0,8,buf,32,1);
    receive(0,8,buf,32,1);
   { ibwrt(dvm,buf,8);
    ibrd(dvm,buf,20); }
    buffer := buf;
    msg := buffer;
     delete(msg,1,3);
     delete(msg,6,3);
     val(msg,ii,code);
      if ii < 10 then
        str(ii:5:2,msg);
    POWERSUPPLY.label5.caption := msg;
   end;

procedure ReadI1;
 Var
    buffer : string[30];
    buf    : array[0..40] of char;
      msg : string[20];
     i:integer;
    ii:real;
    code:integer;
  begin
    //LoadDLL;
    dvm:=ibdev(0,8,0,15,1,0);
    buf := 'mi1 ' ;
```

```
  send(0,8,buf,32,1);
  receive(0,8,buf,32,1);
 {  ibwrt(dvm,buf,8);
  ibrd(dvm,buf,20);   }
  buffer := buf;
  msg := buffer;
  delete(msg,1,3);
  delete(msg,6,3);
  val(msg,ii,code);
   if ii < 10 then
    str(ii:5:2,msg);
  POWERSUPPLY.label7.caption :=msg;
 end;

procedure ReadV2;
 Var
  buffer : string[20];
  buf    : array[0..20] of char;
  msg : string[20];
  i:integer;
  ii:real;
  code:integer;
 begin
  //LoadDLL;
  dvm:=ibdev(0,8,0,15,1,0);
  buf := 'mu2 ' ;
  send(0,8,buf,32,1);
  receive(0,8,buf,32,1);
```

```
  { ibwrt(dvm,buf,8);
    ibrd(dvm,buf,20);   }
    buffer := buf;
    buffer := buf;
    msg := buffer;
    delete(msg,1,3);
    delete(msg,6,3);
    val(msg,ii,code);
     if ii < 10 then
       str(ii:5:2,msg);
   POWERSUPPLY.label6.caption := msg;
 end;



procedure ReadI2;
 Var
   buffer ; string[30];
   buf   : array[0..40] of char;
     msg : string[20];
    i:integer;
   ii:real;
   code:integer;
 begin
   //LoadDLL;
   dvm:=ibdev(0,8,0,15,1,0);
   buf := 'mi2 ' ;
```

```
  send(0,8,buf,32,1);
  receive(0,8,buf,32,1);
 {  ibwrt(dvm,buf,8);
  ibrd(dvm,buf,20);  }
  buffer := buf;
   msg := buffer;
   delete(msg,1,3);
   delete(msg,6,3);
   val(msg,ii,code);
    if ii < 10 then
      str(ii:5:2,msg);
  POWERSUPPLY.label8.caption := msg;

 end;



procedure CheckOutput;
  var
    buf : array[0..30] of char;
    i:integer;
    Output:String;
  begin
    POWERSUPPLY.Table1.FindNearest(['output']);
    Output := POWERSUPPLY.Table1Value.AsString;
    //LoadDLL;
    dvm := ibdev(0,8,0,15,1,0);
```

```
if Output = IntToStr(1) then
  begin
    POWERSUPPLY.shape2.Brush.Color:= clLime ;
    buf := 'op1;';
    send(0,8,buf,32,1);
    receive(0,8,buf,32,1);
 {   ibwrt(dvm,buf,5);
    ibrd(dvm,buf,35);    }
    for i := 1 to length(buf) do
    buf[i]:=' ';
    send(0,8,buf,32,1);
 {   ibwrt(dvm,buf,5);    }
  end
 else  if Output = IntToStr(0) then
  begin
    POWERSUPPLY.shape2.brush.color:= clGreen;
    buf:='op0 ';
    send(0,8,buf,32,1);
{     ibwrt(dvm,buf,5);   }
  //   ibonl(0,0);
    for i := 1 to length(buf) do
    buf[i]:=' ';
    send(0,8,buf,32,1);
 {   ibwrt(dvm,buf,5); }


   end;
end;
```

```
procedure CheckStatus;
Var
  Clear:String;
  begin
   POWERSUPPLY.Table1.FindNearest(['status']);
   Clear:= POWERSUPPLY.Table1Value.AsString;
   If Clear = IntToStr(1) Then
   Begin
   //LoadDLL;
   dvm:=ibdev(0,8,0,15,1,0);
   ibclr(dvm);

   //ibonl(0,0);
   //FreeLibrary(Gpib32Lib);
   end;
  end;



Procedure CheckUnit;
var
Output:string;
begin

   dvm:=ibdev(0,12,0,15,1,0);
```

```
POWERSUPPLY.Table1.FindNearest(['unit_V']);
Output := POWERSUPPLY.Table1value.asString;
if Output = '1' then
  begin
     POWERSUPPLY.shape4.Brush.color:=clLime;
     POWERSUPPLY.shape5.Brush.color:=clGreen;
     POWERSUPPLY.shape5.Brush.color:=clGreen;

  end
else  if Output = '2' then
  begin

    POWERSUPPLY.shape5.Brush.color:=clLime;
     POWERSUPPLY.shape4.Brush.color:=clGreen;
     POWERSUPPLY.shape4.Brush.color:=clGreen;
 end
 else  if Output = '3' then
  begin

    POWERSUPPLY.shape8.Brush.color:=clLime;
     POWERSUPPLY.shape6.Brush.color:=clGreen;
     POWERSUPPLY.shape7.Brush.color:=clGreen;

 end;
  dvm:=ibdev(0,12,0,15,1,0);
POWERSUPPLY.Table1.FindNearest(['unit_I']);
Output := POWERSUPPLY.Table1value.asString;
if Output = '1' then
```

```
      begin
         POWERSUPPLY.shape10.Brush.color:=clLime;
         POWERSUPPLY.shape11.Brush.color:=clGreen;
      end
    else  if Output = '2' then
      begin
        POWERSUPPLY.shape11.Brush.color:=clLime;
         POWERSUPPLY.shape10.Brush.color:=clGreen;
    end
    else  if Output = '3' then
      begin

        POWERSUPPLY.shape8.Brush.color:=clLime;
         POWERSUPPLY.shape6.Brush.color:=clGreen;
         POWERSUPPLY.shape7.Brush.color:=clGreen;
    end;
END;

procedure V1set;
var
    buf    :array[0..30] of char;
    V1     :string;
    i   :integer;
begin
  POWERSUPPLY.shape8.Brush.color:=ClGreen;
  if POWERSUPPLY.shape8.Brush.color= clLime then
    POWERSUPPLY.shape9.Brush.color:= clGreen;
    POWERSUPPLY.shape10.Brush.color:= clGreen;
```

```
  POWERSUPPLY.shape11.Brush.color:= clGreen;
 //LoadDLL;
 dvm:=ibdev(0,8,0,15,1,0);
 POWERSUPPLY.Table1.FindNearest(['V1']);
 POWERSUPPLY.Edit1.Text:= POWERSUPPLY.Table1Value.AsString;
  if POWERSUPPLY.edit1.Text <> '' then
  begin
    V1 := POWERSUPPLY.edit1.text;
    buf := 'su1 ';
    for i := 1 to Length(V1) do
      buf[3 + i ] :=V1[i];
      buf[4 + Length(V1)] := ';';
      send(0,8,buf,32,1);
     { ibwrt(dvm,buf,12);    }
       for i := 1 to length(buf) do
      buf[i] := ' ';
      send(0,8,buf,32,1);
     { ibwrt(dvm,buf,8);   }
    ReadV1;
    //ibonl(0,0);
    //FreeLibrary(Gpib32Lib);
   end;
end;
```

```
procedure  V2set;
var
  I1     : string;
  buf    : array[0..30] of char;
  i  : integer;
begin
   POWERSUPPLY.shape9.Brush.color := clGreen;
   if POWERSUPPLY.shape9.Brush.color = clLime then
     POWERSUPPLY.shape8.Brush.color := clGreen;
     POWERSUPPLY.shape10.Brush.color := clGreen;
     POWERSUPPLY.shape11.Brush.color := clGreen;
   //LoadDLL;
  dvm:=ibdev(0,8,0,15,1,0);
  POWERSUPPLY.Table1.FindNearest(['V2']);
  POWERSUPPLY.Edit1.Text:= POWERSUPPLY.Table1Value.AsString;
   if POWERSUPPLY.edit2.Text <> '' then
   begin
     I1 := POWERSUPPLY.edit1.text;
     buf := 'Su2 ';
     for i := 1 to Length(I1) do
       buf[3 + i ] :=I1[i];
       buf[4 + Length(I1)] := ';';
       send(0,8,buf,32,1);
     //  ibwrt(dvm,buf,12);
        for i := 1 to length(buf) do
       buf[i] := ' ';
       send(0,8,buf,32,1);
     //   ibwrt(dvm,buf,8);
```

```
    ReadV2;
    //ibonl(0,0);
    //FreeLibrary(Gpib32Lib);
   end;
end;

procedure I2set;
var
  I1     : string;
  buf    : array[0..30] of char;
  i  : integer;
begin
    POWERSUPPLY.shape11.Brush.color := clGreen;
    if POWERSUPPLY.shape11.Brush.color = clLime then
      POWERSUPPLY.shape8.Brush.color := clGreen;
      POWERSUPPLY.shape9.Brush.color := clGreen;
      POWERSUPPLY.shape10.Brush.color := clGreen;
    //LoadDLL;
  dvm:=ibdev(0,8,0,15,1,0);
  POWERSUPPLY.Table1.FindNearest(['I2']);
  POWERSUPPLY.Edit2.Text:= POWERSUPPLY.Table1Value.AsString;
   if POWERSUPPLY.edit2.Text <> '' then
   begin
     I1 := POWERSUPPLY.edit2.text;
     buf := 'SI2 ';
     for i := 1 to Length(I1) do
       buf[3 + i ] :=I1[i];
       buf[4 + Length(I1)] := ';';
```

```
        send(0,8,buf,32,1);
      //  ibwrt(dvm,buf,12);
        for i := 1 to length(buf) do
        buf[i] := ' ';
        send(0,8,buf,32,1);
     //   ibwrt(dvm,buf,8);
    ReadI2;
    //ibonl(0,0);
    //FreeLibrary(Gpib32Lib);
   end;
end;

procedure I1set;
var
  I1     : string;
  buf    : array[0..30] of char;
  i  : integer;
begin
    POWERSUPPLY.shape10.Brush.color := clGreen;
    if POWERSUPPLY.shape10.Brush.color = clLime then
      POWERSUPPLY.shape8.Brush.color := clGreen;
      POWERSUPPLY.shape9.Brush.color := clGreen;
      POWERSUPPLY.shape11.Brush.color := clGreen;
    //LoadDLL;
  dvm:=ibdev(0,8,0,15,1,0);
  POWERSUPPLY.Table1.FindNearest(['I1']);
  POWERSUPPLY.Edit2.Text:= POWERSUPPLY.Table1Value.AsString;
   if POWERSUPPLY.edit2.Text <> '' then
```

```
    begin
      I1 := POWERSUPPLY.edit2.text;
      buf := 'SI1 ';
      for i := 1 to Length(I1) do
        buf[3 + i ] :=I1[i];
        buf[4 + Length(I1)] := ';';
        send(0,8,buf,32,1);
      //   ibwrt(dvm,buf,12);
         for i := 1 to length(buf) do
        buf[i] := ' ';
        send(0,8,buf,32,1);
      //    ibwrt(dvm,buf,8);
    ReadI1;
    //ibonl(0,0);
    //FreeLibrary(Gpib32Lib);
   end;
end;

 procedure TPOWERSUPPLY.Timer1Timer(Sender: TObject);
begin
POWERSUPPLY.Table1.Refresh;
V1set;
V2set;
I1set;
I2set;
CheckOutput;
CheckStatus;
Checkunit; end; end.
```

```
*******************************************************************************
```

Source code ส่วน Database

```
*******************************************************************************
program PGPIB;
uses
  Forms,
  Umainform in 'Umainform.pas' {mainform},
  uPassword in 'uPassword.pas' {passform},
  ulab1 in 'ulab1.pas' {lab1form},
  ulab2 in 'ulab2.pas' {lab2form},
  Uchangpass in 'Uchangpass.pas' {changeform},
  Ulabprint in 'Ulabprint.pas' {Form1};
{$R *.RES}
begin
  Application.Initialize;
  Application.CreateForm(Tmainform, mainform);
  Application.Run;
end.
*******************************************************************************
```

Source code ส่วน Menu หลัก

```
*******************************************************************************
unit Umainform;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls, StdCtrls, Animate, GIFCtrl,shellAPI;

type
```

```
Tmainform = class(TForm)
  Panel1: TPanel;
  Image1: TImage;
  Label1: TLabel;
  Label2: TLabel;
  Label3: TLabel;
  Label4: TLabel;
  Label5: TLabel;
  Label6: TLabel;
  Label7: TLabel;
  RxGIFAnimator1: TRxGIFAnimator;
  Edit1: TEdit;
  Label8: TLabel;
  Label9: TLabel;
  Label10: TLabel;
  Label11: TLabel;
  procedure Image1MouseMove(Sender: TObject; Shift: TShiftState; X,
    Y: Integer);
  procedure Label2MouseMove(Sender: TObject; Shift: TShiftState; X,
    Y: Integer);
  procedure Label3MouseMove(Sender: TObject; Shift: TShiftState; X,
    Y: Integer);
  procedure Label4MouseMove(Sender: TObject; Shift: TShiftState; X,
    Y: Integer);
  procedure Label5MouseMove(Sender: TObject; Shift: TShiftState; X,
    Y: Integer);
  procedure Label6MouseMove(Sender: TObject; Shift: TShiftState; X,
    Y: Integer);
  procedure Label7MouseMove(Sender: TObject; Shift: TShiftState; X,
    Y: Integer);
```

```
    procedure Label7Click(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure Label2Click(Sender: TObject);
    procedure Label3Click(Sender: TObject);
    procedure Label4Click(Sender: TObject);
    procedure Label6Click(Sender: TObject);
    procedure Label5Click(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  mainform: Tmainform;

implementation

uses uPassword, ulab1, ulab2, Uchangpass, Ulabprint;

{$R *.DFM}
```

```
procedure Tmainform.Image1MouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
begin
    label1.Font.Color:=$000080FF;
    label2.Font.Color:=$000080FF;
    label3.Font.Color:=$000080FF;
    label4.Font.Color:=$000080FF;
    label5.Font.Color:=$000080FF;
    label6.Font.Color:=$000080FF;
    label7.Font.Color:=$000080FF;
end;

procedure Tmainform.Label2MouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
begin
    label1.Font.Color:=$000080FF;
    label3.Font.Color:=$000080FF;
    label4.Font.Color:=$000080FF;
    label5.Font.Color:=$000080FF;
    label6.Font.Color:=$000080FF;
    label7.Font.Color:=$000080FF;
    label2.Font.Color:=$0000FF80;
end;

procedure Tmainform.Label3MouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
begin
    label1.Font.Color:=$000080FF;
    label2.Font.Color:=$000080FF;
    label4.Font.Color:=$000080FF;
```

```
    label5.Font.Color:=$000080FF;
    label6.Font.Color:=$000080FF;
    label7.Font.Color:=$000080FF;
    label3.font.color:=$0000FF80;
end;

procedure Tmainform.Label4MouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
begin
    label1.Font.Color:=$000080FF;
    label2.Font.Color:=$000080FF;
    label3.Font.Color:=$000080FF;
    label5.Font.Color:=$000080FF;
    label6.Font.Color:=$000080FF;
    label7.Font.Color:=$000080FF;
    label4.font.color:=$0000FF80;
end;

procedure Tmainform.Label5MouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
begin
    label1.Font.Color:=$000080FF;
    label2.Font.Color:=$000080FF;
    label3.Font.Color:=$000080FF;
    label4.Font.Color:=$000080FF;
    label6.Font.Color:=$000080FF;
    label7.Font.Color:=$000080FF;
    label5.font.color:=$0000FF80;
end;
```

```
procedure Tmainform.Label6MouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
begin
    label1.Font.Color:=$000080FF;
    label2.Font.Color:=$000080FF;
    label3.Font.Color:=$000080FF;
    label4.Font.Color:=$000080FF;
    label5.Font.Color:=$000080FF;
    label7.Font.Color:=$000080FF;
    label6.font.color:=$0000FF80;
end;

procedure Tmainform.Label7MouseMove(Sender: TObject; Shift: TShiftState; X,
  Y: Integer);
begin
    label1.Font.Color:=$000080FF;
    label2.Font.Color:=$000080FF;
    label3.Font.Color:=$000080FF;
    label4.Font.Color:=$000080FF;
    label5.Font.Color:=$000080FF;
    label6.Font.Color:=$000080FF;
    label7.font.color:=$0000FF80;
end;

procedure Tmainform.Label7Click(Sender: TObject);
begin

    Close;
    halt;
```

```
end;

procedure Tmainform.Button1Click(Sender: TObject);
begin
    passform.Show;
end;

procedure Tmainform.FormCreate(Sender: TObject);
begin
    Application.CreateForm(Tpassform, passform);
    passform.Showmodal;
//    passform.Free;
end;

procedure Tmainform.Label2Click(Sender: TObject);
begin
    ShellExecute(Self.Handle,'open','c:\prodoc\main.html','','',SW_ShowDefault);
end;

procedure Tmainform.Label3Click(Sender: TObject);
begin
    mainform.hide;
    Application.CreateForm(Tlab1form, lab1form);
    lab1form.Showmodal;
//    lab1form.Free;

end;

procedure Tmainform.Label4Click(Sender: TObject);
begin
```

```
    mainform.hide;
    Application.CreateForm(Tlab2form, lab2form);
    lab2form.Showmodal;
//    lab2form.Free;
end;

procedure Tmainform.Label6Click(Sender: TObject);
begin

    Application.CreateForm(Tchangeform, changeform);
    changeform.Showmodal;
end;

procedure Tmainform.Label5Click(Sender: TObject);
begin
    mainform.Hide;
    Application.CreateForm(TForm1, Form1);
    form1.Showmodal;
end;

procedure Tmainform.FormClose(Sender: TObject; var Action: TCloseAction);
begin
   Action:=caFree;
end;

end.
```

Panel10: TPanel;

Panel11: TPanel;

Panel12: TPanel;

Panel13: TPanel;

Panel14: TPanel;

Panel15: TPanel;

Panel16: TPanel;

Panel17: TPanel;

Panel18: TPanel;

Panel19: TPanel;

RxLabel3: TRxLabel;

Image2: TImage;

Image3: TImage;

Image4: TImage;

Image5: TImage;

RadioGroup1: TRadioGroup;

RadioButton1: TRadioButton;

RadioButton2: TRadioButton;

Button2: TButton;

Image6: TImage;

Image7: TImage;

DataSource1: TDataSource;

Table1: TTable;

DataSource2: TDataSource;

Table2: TTable;

RxLabel2: TRxLabel;

RxLabel4: TRxLabel;

DataSource3: TDataSource;

Table3: TTable;

RxLabel5: TRxLabel;

```
  Table1ID: TStringField;
  Table1StringField1: TStringField;
  Table1StringField2: TStringField;
  Table1StringField3: TStringField;
  Table1StringField4: TStringField;
  Table1StringField5: TStringField;
  Table2ID: TStringField;
  Table2StringField1: TStringField;
  Table2StringField2: TStringField;
  Table2StringField3: TStringField;
  Table2StringField4: TStringField;
  Table2StringField5: TStringField;
  Table3information: TStringField;
  Table3data: TStringField;
  RxClock1: TRxClock;
  procedure UpDown1Click(Sender: TObject; Button: TUDBtnType);
  procedure RadioButton2Click(Sender: TObject);
  procedure RadioButton1Click(Sender: TObject);
  procedure FormCreate(Sender: TObject);
  procedure Button1Click(Sender: TObject);
  procedure Button2Click(Sender: TObject);
  procedure FormClose(Sender: TObject; var Action: TCloseAction);


private
  { Private declarations }
public

  { Public declarations }
end;
```

```
var
  lab1form: Tlab1form;

implementation

uses uPassword, Umainform;

{$R *.DFM}

procedure Tlab1form.UpDown1Click(Sender: TObject; Button: TUDBtnType);
begin
    if edit1.text='1' then
    begin
    image1.Visible:=true;
    image2.Visible:=false;
    image3.Visible:=false;
    image4.Visible:=false;
    image5.Visible:=false;
    end
    else if edit1.text='2' then
    begin
    image1.Visible:=false;
    image2.Visible:=true;
    image3.Visible:=false;
    image4.Visible:=false;
    image5.Visible:=false;
    end
    else if edit1.text='3' then
    begin
```

```
    image1.Visible:=false;
    image2.Visible:=false;
    image3.Visible:=true ;
    image4.Visible:=false;
    image5.Visible:=false;
    end
    else if edit1.text='4' then
    begin
    image1.Visible:=false;
    image2.Visible:=false;
    image3.Visible:=false;
    image4.Visible:=true ;
    image5.Visible:=false;
    end
    else if edit1.text='5' then
    begin
    image1.Visible:=false;
    image2.Visible:=false;
    image3.Visible:=false;
    image4.Visible:=false;
    image5.Visible:=True ;
    end
end;

procedure Tlab1form.RadioButton2Click(Sender: TObject);
begin
    image6.Visible:=false;
    image7.Visible:=true;
end;
```

```
procedure Tlab1form.RadioButton1Click(Sender: TObject);
begin
    image6.Visible:=true;
    image7.Visible:=false;

end;

procedure Tlab1form.FormCreate(Sender: TObject);
Var hSysMnu:THandle;
begin
    hSysMnu:=GetSystemMenu(Handle, False);
    AppendMenu(hSysMnu,mf_Separator,0,nil);
    AppendMenu(hSysMnu,mf_String,101,'โปรแกรมโดย คมกฤษณ์ ชูเรือง');
    radiobutton1.Checked:=true;

    with table1 do
    begin
    Setkey;
    fieldbyname('id').AsString:=mainform.Edit1.Text;
    GotoNearest;
    end;
    // VR1 Display
    Panel6.Caption:=Table1StringField1.AsString;
    Panel9.Caption:=Table1StringField2.AsString;
    Panel12.Caption:=Table1StringField3.AsString;
    Panel15.Caption:=Table1StringField4.AsString;
    Panel18.Caption:=Table1StringField5.AsString;

    with table2 do
    begin
```

```
    Setkey;
    fieldbyname('id').AsString:=mainform.Edit1.Text;
    GotoNearest;
    end;
    // VR2 Display
    Panel7.Caption:=Table2StringField1.AsString;
    Panel10.Caption:=Table2StringField2.AsString;
    Panel13.Caption:=Table2StringField3.AsString;
    Panel16.Caption:=Table2StringField4.AsString;
    Panel19.Caption:=Table2StringField5.AsString;

end;

procedure Tlab1form.Button1Click(Sender: TObject);
begin
    if Radiobutton1.Checked=true then
    begin
    with Table3 do
    begin
        Refresh;
        SetKey;
        fieldbyname('information').AsString:='read';
        GotoNearest;
    end;
    if edit1.text='1' then
    begin table1.Edit;
    Table1StringField1.AsString:=table3Data.AsString;
    table1.Post;
    Panel6.Caption:=Table1StringField1.AsString;
    end
```

```
else if edit1.Text='2' then
begin table1.Edit;
Table1StringField2.AsString:=table3Data.AsString;
table1.Post;
Panel9.Caption:=Table1StringField2.AsString;
end
else if edit1.Text='3' then
begin table1.Edit;
Table1StringField3.AsString:=table3Data.AsString;
table1.Post;
Panel12.Caption:=Table1StringField3.AsString;
end
else if edit1.Text='4' then
begin table1.Edit;
Table1StringField4.AsString:=table3Data.AsString;
table1.Post;
Panel15.Caption:=Table1StringField4.AsString;
end
else if edit1.Text='5' then
begin table1.Edit;
Table1StringField5.AsString:=table3Data.AsString;
table1.Post;
Panel18.Caption:=Table1StringField5.AsString;
end;
end

else if radiobutton2.Checked then
begin
with Table3 do
begin
```

```
    Refresh;
    SetKey;
    fieldbyname('information').AsString:='read';
    GotoNearest;
end;
if edit1.text='1' then
begin table2.Edit;
Table2StringField1.AsString:=table3Data.AsString;
table2.Post;
Panel7.Caption:=Table2StringField1.AsString;
end
else if edit1.Text='2' then
begin table2.Edit;
Table2StringField2.AsString:=table3Data.AsString;
table2.Post;
Panel10.Caption:=Table2StringField2.AsString;
end
else if edit1.Text='3' then
begin table2.Edit;
Table2StringField3.AsString:=table3Data.AsString;
table2.Post;
Panel13.Caption:=Table2StringField3.AsString;
end
else if edit1.Text='4' then
begin table2.Edit;
Table2StringField4.AsString:=table3Data.AsString;
table2.Post;
Panel16.Caption:=Table2StringField4.AsString;
end
else if edit1.Text='5' then
```

```
    begin table2.Edit;
    Table2StringField5.AsString:=table3Data.AsString;
    table2.Post;
    Panel19.Caption:=Table2StringField5.AsString;
    end;
    end;
end;

procedure Tlab1form.Button2Click(Sender: TObject);
begin
    lab1form.Close;
    mainform.show;
end;

procedure Tlab1form.FormClose(Sender: TObject; var Action: TCloseAction);
begin
    Action:=caFree
end;
end.
```

****

Source code ส่วน การทดลองที่ 2

****

```
unit ulab2;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, ExtCtrls, RXCtrls, Db, DBTables, ComCtrls, Animate, GIFCtrl,
  RXClock;
```

```
type
  Tlab2form = class(TForm)
    Panel1: TPanel;
    GroupBox1: TGroupBox;
    Panel2: TPanel;
    RxLabel1: TRxLabel;
    Panel3: TPanel;
    RxLabel2: TRxLabel;
    Panel4: TPanel;
    RxLabel3: TRxLabel;
    Panel5: TPanel;
    Panel6: TPanel;
    Panel7: TPanel;
    Panel8: TPanel;
    Panel9: TPanel;
    Panel10: TPanel;
    Panel11: TPanel;
    Panel12: TPanel;
    Panel13: TPanel;
    RxLabel4: TRxLabel;
    Bevel1: TBevel;
    DataSource1: TDataSource;
    Table1: TTable;
    DataSource2: TDataSource;
    Table2: TTable;
    GroupBox2: TGroupBox;
    Bevel2: TBevel;
    Button1: TButton;
    Button2: TButton;
```

```
    Table1ID: TStringField;
    Table1StringField10K: TStringField;
    Table1StringField20K: TStringField;
    Table1StringField50K: TStringField;
    Table2information: TStringField;
    Table2data: TStringField;
    Edit1: TEdit;
    UpDown1: TUpDown;
    RxLabel5: TRxLabel;
    RxClock1: TRxClock;
    Image1: TImage;
    Image2: TImage;
    Image3: TImage;
    procedure FormCreate(Sender: TObject);
    procedure UpDown1Click(Sender: TObject; Button: TUDBtnType);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);

  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  lab2form: Tlab2form;

implementation
```

```
uses uPassword, Umainform;

{$R *.DFM}

procedure Tlab2form.FormCreate(Sender: TObject);
Var hSysMnu:Thandle;
begin
    hSysMnu:=GetSystemMenu(Handle, False);
    AppendMenu(hSysMnu,mf_Separator,0,nil);
    AppendMenu(hSysMnu,mf_String,101,'');
    if table1.Active=False then table1.Active:=True;
    with table1 do
    begin
    Setkey;
    fieldbyname('id').AsString:=mainform.Edit1.Text;
    GotoNearest;
    end;
    //Display Dara
    panel11.Caption:=Table1StringField10K.AsString;
    panel12.Caption:=Table1StringField20K.AsString;
    panel13.Caption:=Table1StringField50K.AsString;
end;

procedure Tlab2form.UpDown1Click(Sender: TObject; Button: TUDBtnType);
begin
    if updown1.Position=1 then
    begin
    edit1.text:='10K';
    Image1.Visible:=True;
    Image2.Visible:=False;
```

```
    Image3.Visible:=False;
    end
    else if Updown1.Position=2 then
    begin
    edit1.text:='20K';
    Image1.Visible:=False;
    Image2.Visible:=True;
    Image3.Visible:=False;
    end
    else if Updown1.Position=3 then
    begin
    edit1.Text:='50K';
    Image1.Visible:=False;
    Image2.Visible:=False;
    Image3.Visible:=True;
    end;
end;

procedure Tlab2form.Button1Click(Sender: TObject);
begin
    with Table2 do
    begin
        Refresh;
        SetKey;
        fieldbyname('information').AsString:='read';
        GotoNearest;
    end;
    if edit1.text='10K' then
    begin table1.Edit;
    Table1StringField10K.AsString:=table2Data.AsString;
```

```
    table1.Post;
    Panel11.Caption:=Table1StringField10K.AsString;
    end
    else if edit1.Text='20K' then
    begin table1.Edit;
    Table1StringField20K.AsString:=table2Data.AsString;
    table1.Post;
    Panel12.Caption:=Table1StringField20K.AsString;
    end
    else if edit1.Text='50K' then
    begin table1.Edit;
    Table1StringField50K.AsString:=table2Data.AsString;
    table1.Post;
    Panel13.Caption:=Table1StringField20K.AsString;
    end;
end;

procedure Tlab2form.Button2Click(Sender: TObject);
begin
    lab2form.Close;
    mainform.show;
end;

procedure Tlab2form.FormClose(Sender: TObject; var Action: TCloseAction);
begin
    Action:=caFree;
end;
end.
```

```
*****************************************************************************
```

Source code ส่วน การพิมพ์ผลการทดลอง

```
*****************************************************************************
unit Ulabprint;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls, StdCtrls, RXCtrls, Buttons, Db, DBTables;

type
  TForm1 = class(TForm)
    Panel1: TPanel;
    Panel2: TPanel;
    Panel3: TPanel;
    Panel4: TPanel;
    Panel5: TPanel;
    Panel6: TPanel;
    Panel7: TPanel;
    Panel8: TPanel;
    Panel9: TPanel;
    Panel10: TPanel;
    Panel11: TPanel;
    Panel12: TPanel;
    Panel13: TPanel;
    Panel14: TPanel;
    Panel15: TPanel;
    Panel16: TPanel;
    Panel17: TPanel;
```

Panel18: TPanel;

Panel19: TPanel;

Panel20: TPanel;

Panel21: TPanel;

Panel22: TPanel;

Panel23: TPanel;

Panel24: TPanel;

Panel25: TPanel;

Panel26: TPanel;

Panel27: TPanel;

Panel28: TPanel;

Panel29: TPanel;

Panel30: TPanel;

Panel31: TPanel;

RxLabel1: TRxLabel;

RxLabel2: TRxLabel;

GroupBox1: TGroupBox;

Label1: TLabel;

Label2: TLabel;

RxLabel3: TRxLabel;

RxLabel4: TRxLabel;

BitBtn1: TBitBtn;

DataSource1: TDataSource;

Table1: TTable;

DataSource2: TDataSource;

Table2: TTable;

DataSource3: TDataSource;

Table3: TTable;

DataSource4: TDataSource;

Table4: TTable;

```
  Table1ID: TStringField;
  Table1NAME: TStringField;
  Table1PASSWORD: TStringField;
  Table2ID: TStringField;
  Table2StringField1: TStringField;
  Table2StringField2: TStringField;
  Table2StringField3: TStringField;
  Table2StringField4: TStringField;
  Table2StringField5: TStringField;
  Table3ID: TStringField;
  Table3StringField1: TStringField;
  Table3StringField2: TStringField;
  Table3StringField3: TStringField;
  Table3StringField4: TStringField;
  Table3StringField5: TStringField;
  Table4ID: TStringField;
  Table4StringField10K: TStringField;
  Table4StringField20K: TStringField;
  Table4StringField50K: TStringField;
  Label3: TLabel;
  BitBtn2: TBitBtn;
  procedure FormCreate(Sender: TObject);
  procedure BitBtn1Click(Sender: TObject);
  procedure FormClose(Sender: TObject; var Action: TCloseAction);
  procedure BitBtn2Click(Sender: TObject);
private
  { Private declarations }
public
  { Public declarations }
end;
```

```
var
  Form1: TForm1;

implementation

uses Umainform,Printers;

{$R *.DFM}

procedure TForm1.FormCreate(Sender: TObject);
begin
    label3.Visible:=false;
    label3.caption:=mainform.Edit1.Text;
    with table1 do
    begin
    Setkey;
    FieldByname('id').AsString:=label3.caption;
    GotoNearest;
    end;
    with table2 do
    begin
    Setkey;
    FieldByname('id').AsString:=label3.caption;
    GotoNearest;
    end;
    with table3 do
    begin
    Setkey;
    FieldByname('id').AsString:=label3.caption;
```

```
    GotoNearest;
    end;
    with table4 do
    begin
    Setkey;
    FieldByname('id').AsString:=label3.caption;
    GotoNearest;
    end;
    RxLabel4.Caption:=Table1Id.AsString;
    RxLabel3.Caption:=Table1Name.AsString;
    {Display Data}
    Panel6.Caption:=Table2StringField1.AsString;
    Panel9.Caption:=Table2StringField2.AsString;
    Panel12.Caption:=Table2StringField3.AsString;
    Panel15.Caption:=Table2StringField4.AsString;
    Panel18.Caption:=Table2StringField5.AsString;

    Panel7.Caption:=Table3StringField1.AsString;
    Panel10.Caption:=Table3StringField2.AsString;
    Panel13.Caption:=Table3StringField3.AsString;
    Panel16.Caption:=Table3StringField4.AsString;
    Panel19.Caption:=Table3StringField5.AsString;

    Panel29.Caption:=Table4StringField10K.AsString;
    Panel30.Caption:=Table4StringField20K.AsString;
    Panel31.Caption:=Table4StringField50K.AsString;

end;

procedure TForm1.BitBtn1Click(Sender: TObject);
```

```
Var
  dc: HDC;
  isSrcDcPalDevice: BOOL;
  isPrnDCPalDevice: BOOL;
  MemDc: hdc;
  MemBitmap: hBitmap;
  OldMemBitmap: hBitmap;
  hDibHeader: Thandle;
  pDibHeader: pointer;
  hBits: Thandle;
  pBits: pointer;
  TestInt: integer;
  PageWidth: integer;
  PageHeight: integer;
  PageMargin: TPoint;
  ImagablePageWidth: integer;
  ImagablePageHeight: integer;
  OffsetX: integer;
  OffsetY: integer;
  ScaleX: integer;
  ScaleY: integer;
  ppal: PLOGPALETTE;
  pal: hPalette;
  Oldpal: hPalette;
  i: integer;
begin
    {Allow the button to repaint}
    Application.ProcessMessages;

    {Get the Screen dc for screenshot and conversion}
```

```
dc:=GetDc(0);

{Create a compatible dc}
MemDc:=CreateCompatibleDc(0);

{Create a compatible bitmap}
MemBitmap:=CreateCompatibleBitmap(Dc,form1.width,form1.height);

{Select the Bitmap into the memory dc}
OldMemBitmap:= SelectObject(MemDc, MemBitmap);

{Get the system palette if necessary}
pPal:= nil;
Pal:= 0;
OldPal:= 0;
isSrcDcPalDevice:= false;
if GetDeviceCaps(dc,RASTERCAPS) and RC_PALETTE=RC_PALETTE then begin
GetMem(pPal,sizeof(TLOGPALETTE) + (255*sizeof(TPALETTEENTRY)));
FillChar(pPal^,sizeof(TLOGPALETTE)+ (255*sizeof(TPALETTEENTRY)), #0);
pPal^.palVersion:= $300;
pPal^.palNumEntries:= GetSystemPaletteEntries(dc,0,256,pPal^.palPalEntry);
if pPal^.palNumEntries<>0 then begin
  pal:=CreatePalette(pPal^);
  oldpal:=SelectPalette(MemDc,Pal,false);
  isSrcDcPalDevice:=true end else
  FreeMem(pPal,sizeof(TLOGPALETTE)+(255*sizeof(TPALETTEENTRY)));
end;

{Copy from the screen to the memory dc}
Bitblt(MemDc,0,0,form1.width,form1.height,Dc,form1.Left,form1.Top,SrcCopy);
```

```
if isSrcDcPalDevice=True then
  SelectPalette(MemDc,OldPal,false);

{Unselect the Bitmap from the memory dc}
SelectObject(MemDc, OldMemBitmap);

{Delete the memory dc}
DeleteDC(MemDC);

{Allocate memory for a DIB structure}
hDibHeader:= GlobalAlloc(GHND,sizeof(TBITMAPINFO)+(sizeof(TRGBQUAD)*256));

{Get a Pointer to memory block}
pDibHeader:=GlobalLock(hDibHeader);

{Fill in the Dib header with the way we want the DIB}
FillChar(pDibHeader^,sizeof(TBITMAPINFO)+(sizeof(TRGBQUAD)*256),#0);
PBITMAPINFOHEADER(PDibHeader)^.biSize:=sizeof(TBITMAPINFOHEADER);
PBITMAPINFOHEADER(pDibHeader)^.biPlanes:=1;
if isSrcDcPalDevice then
  PBITMAPINFOHEADER(pDibHeader)^.biBitCount:=8 else
  PBITMAPINFOHEADER(pDibHeader)^.biBitCount:=24;
  PBITMAPINFOHEADER(pDibHeader)^.biWidth:=form1.Width;
  PBITMAPINFOHEADER(pDibHeader)^.biHeight:=form1.Height;
  PBITMAPINFOHEADER(pDibHeader)^.biCompression:=BI_RGB;
{Find out how much memory is required for the bits}
GetDiBits(dc,MemBitMap,0,form1.height,nil,Tbitmapinfo(pDibHeader^),
      DIB_RGB_COLORS);

{Allocate memory for the bits}
```

```
hBits:=GlobalAlloc(GHND,PBitMapInfoHeader(pDibHeader)^.biSizeImage);

{Get a pointer to the memory block}
pBits:=GlobalLock(hBits);

{Get the Bits}
GetDIBits(dc,MemBitMap,0,form1.height,pBits,PBitMapInfo(pDibHeader)^,
        DIB_RGB_COLORS);

{If the video is a palette device the fill in the DIB palette}
if isSrcDcPalDevice=true then begin
for i:=0 to (pPal^.palNumEntries--1) do begin
PBitMapInfo(pDibHeader)^.bmiColors[i].rgbRed := pPal^.PalPalEntry[i].peRed;
PBitMapInfo(pDibHeader)^.bmiColors[i].rgbGreen := pPal^.PalPalEntry[i].peGreen;
PBitMapInfo(pDibHeader)^.bmiColors[i].rgbBlue := pPal^.PalPalEntry[i].peBlue;'
end;
FreeMem(pPal,sizeof(TLOGPALETTE)+(255*sizeof(TPALETTEENTRY)));
end;

{Release the screen dc}
ReleaseDc(0,dc);

{Delete the memory Bitmap}
DeleteObject(MemBitMap);

{Start print job}
Printer.BeginDoc;

{If the Printer is a palette device then use the palette}
isPrnDcPalDevice:=false;
```

```
if ((isSrcDcPalDevice=true) and (GetDeviceCaps(Printer.Canvas.Handle,
                        RASTERCAPS) and
   RC_PALETTE=RC_PALETTE)) then begin
   OldPal:=SelectPalette(Printer.Canvas.Handle,Pal,false);
   isPrnDcPalDevice:=true
end;

{Get the page margin}
Pagemargin.x:=0;
Pagemargin.y:=0;
TestInt:=GETPRINTINGOFFSET;
if Escape(Printer.Canvas.Handle,QUERYESCSUPPORT,sizeof(testint),@Testint,
      nil) <> 0 then begin
  if Escape(Printer.Canvas.Handle,GETPRINTINGOFFSET,0,nil,@PageMargin)
    <=0 then begin
    Pagemargin.x:=0;
    Pagemargin.y:=0;
    end;
end;

{Calculate the imagable area}
ImagablePageWidth:=PageWidth - (2*PageMargin.x);
ImagablePageHeight:=PageHeight - (2*PageMargin.y);

{Scale the printout size}
if ((ImagablePageWidth<=ImagablePageHeight) and
   (Form1.Width>=Form1.Height)) Or
  ((ImagablePageWidth>ImagablePageHeight) and
   (Form1.Width<Form1.Height)) then begin
   ScaleX:=ImagablePageWidth;
```

```
        ScaleY:=Trunc(Form1.Height*(ImagablePageWidth/Form1.Width));
        OffsetX:=PageMargin.x;
        OffsetY:=(PageHeight div 2 )-(ScaleY div 2 );
        end else begin
        ScaleX:=Trunc(Form1.Width*(ImagablePageHeight/Form1.Height));
        ScaleY:=ImagablePageHeight;
        OffsetX:=(PageWidth div 2 )-(ScaleX div 2 );
        OffsetY:=Pagemargin.y;
      end;

      {Send the DIB to the printer}
      StretchDiBits(Printer.Canvas.Handle,OffsetX,OffsetY,ScaleX,ScaleY,0,0,
            Form1.Width,Form1.Height,pBits,pBitMapInfo(pDibHeader)^,
            DIB_RGB_COLORS,SRCCOPY);

      {If the Printer is a palette device then delete the palette}
      if isPrnDcPalDevice=true then
      SelectPalette(Printer.Canvas.Handle,OldPal,false);
      if isSrcDcPalDevice = true then
      DeleteObject(Pal);

      {Clean up allocated memory}
      GlobalUnLock(hBits);
      Globalfree(hBits);
      GlobalUnLock(hDibHeader);
      GlobalFree(hDibHeader);

      {Enf the print job}
      Printer.EndDoc;
    end;
```

```
procedure TForm1.FormClose(Sender: TObject; var Action: TCloseAction);
begin
    Action:=caFree;
end;

procedure TForm1.BitBtn2Click(Sender: TObject);
begin
    Close;
    mainform.Show;
end;
end.
```

****

Source code ส่วน การเปลี่ยน Password

****

```
unit Uchangpass;

interface

uses
 Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
 ExtCtrls, CoolForm, StdCtrls, CoolButton, Db, DBTables;

type
 Tchangeform = class(TForm)
   CoolForm1: TCoolForm;
   GroupBox1: TGroupBox;
   Edit1: TEdit;
   Edit2: TEdit;
   Label1: TLabel;
   Label2: TLabel;
```

```
    CoolButton1: TCoolButton;
    DataSource1: TDataSource;
    Table1: TTable;
    Table1ID: TStringField;
    Table1NAME: TStringField;
    Table1PASSWORD: TStringField;
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure CoolButton1Click(Sender: TObject);
    procedure Edit2KeyPress(Sender: TObject; var Key: Char);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  changeform: Tchangeform;

implementation

uses Umainform;

{$R *.DFM}

procedure Tchangeform.FormClose(Sender: TObject; var Action: TCloseAction);
begin
    Action:=caFree;
end;

procedure Tchangeform.CoolButton1Click(Sender: TObject);
```

```
begin
    Close;
end;

procedure Tchangeform.Edit2KeyPress(Sender: TObject; var Key: Char);
begin
    if Key=#13 then
    if edit1.text<>edit2.Text then
    begin
    messagedlg('The password you typed do not match.',mtinformation
    ,[mbOK],0);
    Close;
    end else
    begin
    table1.Active:=true;
    with table1 do
    begin
        SetKey;
        FieldByName('id').Asstring:=mainform.edit1.text;
        GotoNearest;
    end;
    Table1.edit;
    Table1Password.AsString:=edit1.Text;
    Table1.Post;
    messagedlg('Your Password has been changed.',mtinformation
    ,[mbOK],0);
    Close;
    end;
end;
end.
```

```
*******************************************************************************
                    Source code ส่วน การตรวจสอบ Password
*******************************************************************************
unit uPassword;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  ExtCtrls, CoolForm, StdCtrls, CoolButton, Db, DBTables;

type
  Tpassform = class(TForm)
    CoolForm1: TCoolForm;
    GroupBox1: TGroupBox;
    Label1: TLabel;
    Label2: TLabel;
    Edit1: TEdit;
    Edit2: TEdit;
    CoolButton1: TCoolButton;
    DataSource1: TDataSource;
    Table1: TTable;
    Table1ID: TStringField;
    Table1NAME: TStringField;
    Table1PASSWORD: TStringField;
    procedure CoolButton1Click(Sender: TObject);
    procedure Edit2KeyPress(Sender: TObject; var Key: Char);
    procedure FormCreate(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
  private
```

```
  { Private declarations }
 public
  { Public declarations }
 end;

var
 passform: Tpassform;

implementation

uses Umainform;

{$R *.DFM}

procedure Tpassform.CoolButton1Click(Sender: TObject);
begin
   Halt;
end;

procedure Tpassform.Edit2KeyPress(Sender: TObject; var Key: Char);
begin
      if key=#13 then
      begin
      with Table1 do
      begin
         Setkey;
         FieldByName('id').AsString:=Edit1.Text;
         GotoNearest;
         if FieldByName('id').AsString<>Edit1.Text Then
         //กรณีไม่พบรหัสนักศึกษา
```

```
        begin
        Messagedlg('Username ไม่ถูกต้อง ออกจากโปรแกรม',mtInformation,[mbOK],0);
        Halt;
        end

        else if edit2.text<>table1Password.AsString then
        // กรณี Password ไม่ถูกต้อง
        begin
        Messagedlg('Password ไม่ถูกต้อง ออกจากโปรแกรม',mtInformation,[mbOK],0);
        Halt;
        end
        else
        //ผ่านการตรวจสอบ
        begin
        mainform.edit1.text:=passform.Edit1.text;
        mainform.label8.caption:=table1name.asstring;
        mainform.Label9.caption:=mainform.edit1.text;
        passform.close;
        mainform.show;
        end;
    end;
    end;
end;

procedure Tpassform.FormCreate(Sender: TObject);
begin
   table1.Active:=True;
end;
```

```
procedure Tpassform.FormClose(Sender: TObject; var Action: TCloseAction);
begin
    Action:=caFree;
end;
end.
```