



การประยุกต์ใช้งาน IEEE-488

IEEE-488 APPLICATION

โดย

นาย ปิติ เอี่ยมแยม

นาย ไพฑูรย์ อินทร์กลับ

นาย สกนธ์ สุ่มงคล

วัน เดือน ปี..... 22 ค.ค. 2541  
เลขทะเบียน..... 039125  
เลขเรียกหนังสือ..... T40364 พ.615ก.

ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาอิเล็กทรอนิกส์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2540

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

039125

การประยุกต์ใช้งาน IEEE-488  
IEEE-488 APPLICATION



ปริญญานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต  
สาขาวิชาอิเล็กทรอนิกส์  
คณะวิศวกรรมศาสตร์  
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง  
ปีการศึกษา 2540

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปริญญานิพนธ์ ปีการศึกษา 2540

ภาควิชา อิเล็กทรอนิกส์

คณะวิศวกรรมศาสตร์สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การประยุกต์ใช้งาน IEEE-488

ผู้จัดทำ

นาย ปิติ เอี่ยมเข้ม รหัส 38013235

นาย ไพฑูรย์ อินทร์กลับ รหัส 38013238

นาย สกนธ์ สุ่มงคล รหัส 38013248

..... อาจารย์ที่ปรึกษา  
(.....)

การประยุกต์ใช้งาน IEEE-488  
IEEE-488 APPLICATION

นาย ปิติ เอี่ยมเข้ม รหัส 38013235  
นาย ไพฑูรย์ อินทร์กลับ รหัส 38013238  
นาย สกนธ์ สุ่มงคล รหัส 38013248

โครงการนี้รับการตรวจสอบแล้ว พร้อมทั้งจะทำการสอบได้



(.....)

อาจารย์ที่ปรึกษา

## การประยุกต์ใช้งาน IEEE-488 (GPIB)

นาย ปิติ เอี่ยมแยม รหัส 38013235

นาย ไพฑูรย์ อินทร์กลับ รหัส 38013238

นาย สกนธ์ สุ่มกล รหัส 38013248

รศ.ดร. มนต์ ตั้งวรศิลป์ อาจารย์ที่ปรึกษา

### บทคัดย่อ

ปริญญานิพนธ์นี้เป็นการประยุกต์ใช้งาน IEEE-488 (GPIB) ซึ่งเป็นระบบเชื่อมต่อมาตรฐานกล่าวถึง การควบคุมอุปกรณ์เครื่องมือวัดต่างๆ โดยการนำคอมพิวเตอร์มาใช้เป็นตัวควบคุมระบบ โครงงานนี้แบ่งออกเป็น 2 ส่วน ส่วนแรกคือ ส่วนแสดงผล และควบคุมอุปกรณ์เครื่องมือวัด โดยทั้งสองส่วนจะอยู่บนเมนูหลัก ในส่วนควบคุมจะประกอบไปด้วย ฟังก์ชันที่สร้างเป็นหน้าต่างของอุปกรณ์แต่ละตัว ซึ่งผู้ใช้งานสามารถควบคุมอุปกรณ์โดยการตั้งค่าลงบนฟอร์มเหล่านี้ และส่วนแสดงผลจะทำหน้าที่ในการเก็บข้อมูลที่วัดได้จากการทดลองลงบนตารางฐานข้อมูลที่สร้างขึ้นและสามารถเลือกนำออกแสดงในรูปของกราฟได้

## IEEE-488 APPLICATION.

Mr. Piti Ieamyam ID. 38013235

Mr. Phaitoon Inklub ID. 38013238

Mr. Sakon Sumongkon ID. 38013248

Assoc Prof. Manut Sungwaonsin Avisor

1997

### ABSTRACT

This thesis is the application of IEEE-488 (GPIB) with is the standard for controlling the measurement equipment by computer. This thesis comprises two parts. In the first section, use to display and control GPIB devices. It consists multi form that can link to devices. User capable control GPIB devices by set value through this form. The second section, use to record data on database and display it on the table.

|                | สารบัญ                                                |      |
|----------------|-------------------------------------------------------|------|
| เรื่อง         |                                                       | หน้า |
| บทที่ 1        | บทนำ                                                  | 1    |
| บทที่ 2        | การใช้งานอุปกรณ์ GPIB                                 | 2    |
|                | 2.1 ระบบพื้นฐานของ GPIB                               | 2    |
|                | 2.2 อุปกรณ์ที่มี GPIB ทำหน้าที่                       | 2    |
|                | 2.3 ขีดจำกัดของ IEEE-488                              | 2    |
|                | 2.4 การกำหนด GPIB Addressing                          | 3    |
|                | 2.5 รายละเอียดเกี่ยวกับ GPIB                          | 3    |
|                | 2.6 ความหมายของสัญญาณต่างๆ ภายใน GPIB                 | 4    |
|                | 2.7 การเชื่อมต่ออุปกรณ์ต่างๆ ในระบบ IEEE-488 BUS      | 5    |
|                | 2.8 การใช้โปรแกรม NI-488.2M กับ Windows 95            | 6    |
|                | 2.9 ฟังก์ชันในการติดต่อกับ NI-488.2M                  | 8    |
| บทที่ 3        | การใช้โปรแกรม NI-488.2M กับ 32-bit Delphi Application | 12   |
|                | 3.1 การเขียนโปรแกรมบน NI-488 หรือ NI-488.2M           | 12   |
|                | 3.2 ขั้นตอนการสร้างโปรแกรมประยุกต์                    | 18   |
|                | 3.2.1 Function Generator HM 8130                      | 19   |
|                | 3.2.2 Power Supply HM 8142                            | 21   |
|                | 3.2.3 Multimeter HM 8122-2                            | 23   |
|                | 3.2.4 Oscilloscope HM 1007                            | 25   |
| บทที่ 4        | การประยุกต์ใช้งาน โปรแกรมควบคุมอุปกรณ์                | 30   |
|                | 4.1 ส่วนของ Main Form                                 | 31   |
|                | 4.2 ส่วนแสดงข้อมูลขณะวัด Frequency Response           | 32   |
|                | 4.3 ส่วนเก็บบันทึกและแสดงผล                           | 34   |
|                | 4.4 คาด้าเบสใน Delphi                                 | 35   |
|                | 4.5 การติดตั้งอุปกรณ์และการทดลอง                      | 37   |
|                | 4.6 ผลการทดลอง                                        | 39   |
| บทที่ 5        | สรุปและวิจารณ์                                        | 43   |
| หนังสืออ้างอิง |                                                       | 46   |
| ภาคผนวก        |                                                       |      |

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

## บทที่ 1

### บทนำ

ในการเชื่อมต่ออุปกรณ์ภายนอกกับคอมพิวเตอร์ ได้มีการนำมาตรฐานในการติดต่อ IEEE 488.1-1987 หรือที่นิยมเรียกกันว่า GPIB (General Purpose Interface Bus) ซึ่งมีลักษณะในการติดต่อระหว่างเครื่องมือวัดและอุปกรณ์ควบคุมจากผู้ใช้(User) ลักษณะในการใช้งานจะเกี่ยวกับทางด้านอิเล็กทรอนิกส์ และทางด้านเครื่องกล

**GPIB จะมีลักษณะเป็นสัญญาณดิจิทัลโดยพื้นฐานที่สำคัญจะประกอบด้วย**

- การส่งข้อมูลแบบขนาน มีขนาด 8 บิต มีความเร็วในการส่ง 1 เมกะไบต์ต่อ

วินาที(Mbytes/S)

- การใช้สายสัญญาณ 3 เส้นในการแฮนด์เชก(Handshake) และใช้สายสัญญาณ 5 เส้นในการควบคุม (Controller)

และได้มีการพัฒนามาเป็น IEEE 488.2-1992 โดยใช้กติกาการติดต่อแบบบัส มีการเชื่อมต่อชนิดของข้อมูลและรูปแบบของข้อมูล

เนื่องจากความสามารถของอุปกรณ์ GPIB ที่สามารถเชื่อมต่อกันเป็นระบบในเครื่องมือวัดหรือเครื่องมือที่ใช้ในการทดลอง เราสามารถนำความสามารถเหล่านี้มาประยุกต์ใช้งาน ดังที่ในโครงการนี้ได้ทำการเขียนโปรแกรมด้วยภาษา Delphi ที่สามารถควบคุมและปรับค่าต่างๆ ของอุปกรณ์ซึ่งจะอยู่ในส่วนแสดงผล และการรับค่าจากอุปกรณ์ซึ่งจะอยู่ในส่วนการเก็บบันทึกข้อมูล รวมทั้งยังได้จัดทำส่วนการประยุกต์ใช้งาน ทำการวัดหาค่า Frequency Response ของ System

## บทที่ 2

### การใช้งานอุปกรณ์ GPIB

#### 2.1 ระบบพื้นฐานของ GPIB

ในระบบพื้นฐานของ GPIB จะประกอบด้วยอุปกรณ์พื้นฐานคือ ผู้ส่ง (Talker) ผู้รับ (Listener) และ ผู้ควบคุม (Controller)

- ผู้ส่ง ทำหน้าที่ส่งข้อมูล โดยในระบบสามารถมี ผู้ส่ง ได้หลายตัวแต่จะมีตัวเดียวเท่านั้นที่กำลังทำงานอยู่
- ผู้รับ ทำหน้าที่เป็นตัวรับข้อมูล โดยในระบบเดียวกันสามารถมี ผู้รับ ได้หลายตัวเช่นกัน แต่ ผู้รับสามารถทำได้ครั้งละหลายๆ ตัวได้
- ผู้ควบคุม ทำหน้าที่ควบคุมอุปกรณ์ต่างๆ ในระบบ โดยจะกำหนดให้ ผู้ส่งทำการส่งข้อมูลหรือกำหนดให้ ผู้รับ รับข้อมูล

#### 2.2 อุปกรณ์ที่มี GPIB นั้นสามารถแบ่งหน้าที่ได้ดังนี้

- ทำหน้าที่เป็น ผู้ส่ง เท่านั้นเช่น เครื่องมือวัด เป็นต้น
- ทำหน้าที่เป็น ผู้รับ เท่านั้นเช่น เครื่องพิมพ์ (printer) เครื่องบันทึก (Recorder)
- ทำหน้าที่เป็นทั้ง ผู้ส่ง และผู้รับ เช่น คอมพิวเตอร์ เครื่องวัด
- ทำหน้าที่เป็น ผู้ส่ง ผู้รับ และผู้ควบคุมในตัวเดียวกันเช่นคอมพิวเตอร์ควบคุมระบบ

#### 2.3 ขีดจำกัดของ IEEE-488

- จำนวนอุปกรณ์ของระบบ (ผู้ส่ง , ผู้รับ , ผู้ควบคุม) ที่ต่อกับสายสัญญาณ 1 เส้นจะต้องไม่เกิน 15 เครื่อง
- สายเคเบิลที่ต่อระหว่างอุปกรณ์จะต้องยาวไม่เกิน 4 เมตร และความยาวของสายรวมในระบบจะ ต้องไม่เกิน 20 เมตร
- ความยาวในการส่งข้อมูลจะต้องไม่เกิน 1 Mb/Sec (1 ล้าน ไบต์ต่อหนึ่งวินาที)
- จะต้องมีการจ่ายไฟให้กับอุปกรณ์มากกว่าครึ่งหนึ่งของระบบ

## 2.4 การกำหนด GPIB Addressing

อุปกรณ์ GPIBและบอร์ดจะต้องมีการกำหนดตำแหน่ง(Address)ซึ่งในการกำหนดตำแหน่งนั้นสามารถทำได้ 2 ส่วนคือ ตำแหน่งหลัก(Primary Address)และตำแหน่งสำรอง(Secondary Address) ดังแสดงดังรูปที่ 2.1

| Bit Position | 7 | 6  | 5  | 4                                 | 3 | 2 | 1 | 0 |
|--------------|---|----|----|-----------------------------------|---|---|---|---|
| Meaning      | 0 | TA | LA | GPIB Primary Address (range 0-30) |   |   |   |   |

รูปที่ 2.1แสดงการกำหนดตำแหน่งของ GPIB

- การกำหนดตำแหน่งหลัก คือจะแสดงค่าในช่วง 0-30 และในการควบคุมอุปกรณ์ GPIBจะใช้การกำหนดตำแหน่งในช่องของ TA และ LA

- การกำหนดผู้ส่ง(TA) โดยการเซตที่บิต6

- การกำหนดผู้รับ(LA) โดยการเซตที่บิต5

ตัวอย่าง ถ้าอุปกรณ์อยู่ที่ตำแหน่ง1 แล้วตัวควบคุมส่งข้อมูล 41H (จะทำให้ที่ตำแหน่ง TA เป็น '1')ทำให้อุปกรณ์ทำหน้าที่เป็นผู้ส่ง และโดยทั่วไปตัวควบคุมจะใช้ตำแหน่งหลัก(Primary Address)ที่ตำแหน่ง '0' และจะส่งค่า 20H (จะทำการเซต LA เป็น '1')จะทำให้ตัวเองทำหน้าที่เป็นผู้รับข้อมูลหรือคำสั่งควบคุม

- การกำหนดตำแหน่งสำรอง คือจะแสดงค่าในช่วง 60H-7E H เมื่อตำแหน่งสำรองถูกใช้ ตัวควบคุมจะถูกกำหนดให้เป็นผู้ส่งหรือผู้รับเพื่อส่งค่าให้กับอุปกรณ์ ตามค่าในการกำหนดตำแหน่งสำรอง

## 2.5 รายละเอียดเกี่ยวกับ GPIB

นั่นคือเป็นสายสัญญาณแบบ 24 เส้น ขนานกันและมีขั้วอยู่ทางปลายทางทั้งสองของสายเพื่อต่อกับ อุปกรณ์หรือต่อกันเพื่อให้สายสัญญาณมีความยาวเพิ่มขึ้นจำนวนสายสัญญาณ 24 เส้นนี้ มีเพียง 16 เส้นเท่านั้นที่ทำหน้าที่นำสัญญาณ ส่วนที่เหลืออีก 8 เส้นทำหน้าที่กราวด์ (GROUND) ชิลด์ (SHIELD) โดยจำนวนสายที่นำสัญญาณ 16 เส้น ยังแบ่งได้เป็น 3 ประเภท

### 2.5.1 บัสข้อมูล (Data Bus) จำนวน 8 สาย

DIO1-DIO8

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

### 2.5.2 สายสัญญาณควบคุม (Controller) จำนวน 5 สาย คือ

IFC (Interface Clear)

ATN (Attention)

SRQ (Service Request)

REN (Remote Enable)

EOI (End Or Identify)

### 2.5.3 สายแฮนด์เชก (HAND SHAKE) 3 สาย คือ

DAV (Data Valid)

NRFD (Not Ready For Data)

NDAC (Not Data Accepted)

## 2.6 ความหมายของสัญญาณต่าง ๆ ภายใน GPIB

ดังที่กล่าวมาแล้วว่าสายสัญญาณ ต่าง ๆ ใน GPIB ได้แบ่งออกเป็น 3 กลุ่ม ในหัวข้อนี้จะอธิบายได้ดังนี้

### 2.6.1 กลุ่มสัญญาณข้อมูล

DIO1-DIO8 (Data Input/Output) สัญญาณทั้ง 8 เส้นนี้ทำหน้าที่เป็นทางผ่านข้อมูลของระบบ

### 2.6.2 กลุ่มสัญญาณควบคุมการเชื่อมต่อ (interface)

- IFC (Interface Clear) เป็นสัญญาณรีเซ็ตหรือเคลียร์ระบบ กำเนิดได้โดยตัวควบคุมเท่านั้น เมื่ออุปกรณ์ในบัสได้รับสัญญาณเคลียร์นี้ จะกลับคืนสถานะเริ่มต้นใหม่ซึ่งเป็น สถานะเริ่มก่อนการกำหนดฟังก์ชัน เหมือนแรกเปิด (sw)

- ATN (Attention) เป็นสัญญาณที่ถูกส่งโดยอุปกรณ์ ที่เป็นตัวควบคุมเช่นเดียวกันใช้ในการให้อุปกรณ์ ทุกตัวในระบบเตรียมพร้อมเพื่อรอคำสั่งต่อไป

- SRQ (Service Request) เป็นสัญญาณที่ถูกส่งจากอุปกรณ์ต่างๆ เพื่อเป็นการบอกแก่ระบบว่าขณะนี้อุปกรณ์ดังกล่าวต้องการติดต่อจากตัวควบคุม

- REN (Remote Enable) สัญญาณนี้เป็นสัญญาณที่ถูกส่งจากตัวควบคุม เพียงตัวเดียวเท่านั้น เพื่อใช้สั่งให้อุปกรณ์ต่าง ๆ เปลี่ยนจากโหมดใช้งานปกติมาเป็นการควบคุมโดยตัวควบคุมแทน

- EOI (End Or Identify) เป็นสัญญาณที่ถูกส่งได้ โดยอุปกรณ์ที่เป็นตัวควบคุมหรืออุปกรณ์ที่เป็นตัวส่ง ก็ได้ ใช้สำหรับแสดงว่าข่าวสารที่ส่งเป็นชุดนั้นได้เสร็จสิ้นลงแล้ว

### 2.6.3 กลุ่มสัญญาณการรับส่งข้อมูล

- DAV (Data Valid) เมื่อสัญญาณนี้ถูกดึงเป็นลอจิก “low” โดยอุปกรณ์ที่เป็นตัวส่ง เป็นการขณะนี้ตัวส่งได้ทำการส่งข้อมูลลงไปที่สายข้อมูลเรียบร้อยแล้ว

- NRFD (Nor Read For Data) เมื่อสัญญาณนี้มีลอจิกเป็น “low” จะเป็นการแสดงว่าขณะนี้ระบบบัสยังไม่พร้อมที่จะรับข้อมูล เนื่องจากอุปกรณ์ในระบบยังพร้อมไม่หมดทุกตัวซึ่งสัญญาณเส้นนี้จะไม่เป็น “HI” จนอุปกรณ์ทุกตัวให้ลอจิกเป็น “HI” ครบถ้วนแล้วสัญญาณนี้มีประโยชน์ในกรณีที่อุปกรณ์ระบบมีความเร็วที่แตกต่างกัน

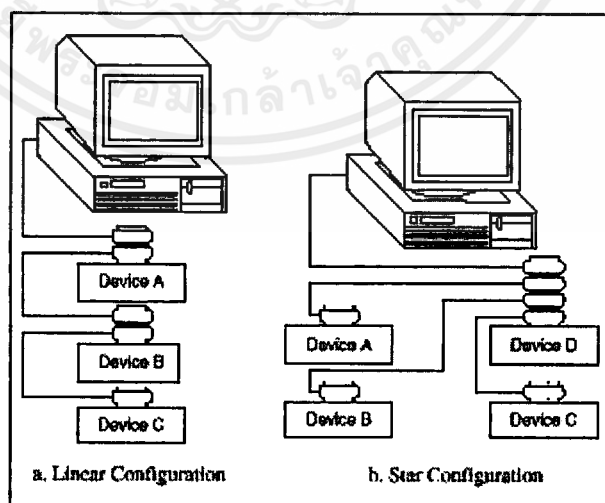
- NDAC (Not Data Accepted) สัญญาณเส้นนี้เป็นสัญญาณที่ถูกควบคุมโดยอุปกรณ์ที่เป็นตัวรับ โดยสัญญาณนี้จะมีลอจิกเป็น “HI” ในขณะอุปกรณ์ที่เป็นตัวกำลังเก็บ ข้อมูลจากสายข้อมูล และจะเป็น “HI” เมื่ออุปกรณ์นั้นได้ทำการอ่านเสร็จเรียบร้อยแล้ว โดยสัญญาณลอจิกที่ใช้ใน (DI-D8) ของ IEEE -488 มีลักษณะเป็น คอมพลิเมนต์ทั้งหมดคือ “1” เท่ากับ (LOW) และ “0” เท่ากับ (HI)

## 2.7 การเชื่อมต่ออุปกรณ์ต่าง ๆ ในระบบ IEEE-488 BUS

สำหรับการเชื่อมต่อระหว่างอุปกรณ์ต่าง ๆ ในระบบ IEEE-488 นั้นมีอยู่ 2 วิธี คือ

### 2.7.1 การเชื่อมแบบเรียงต่อเนื่อง (Linear Configuration)

### 2.7.2 การเชื่อมแบบกระจาย (Star Configuration)

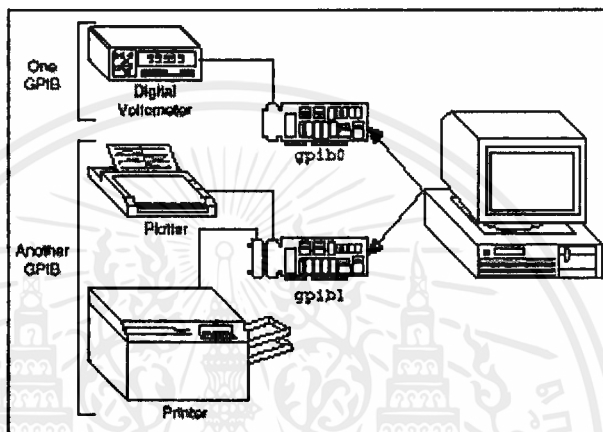


รูปที่ 2.2 แสดงการเชื่อมต่ออุปกรณ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

และจากรูปที่ 2.3 แสดงตัวอย่างการเชื่อมต่อระบบโดยใช้อุปกรณ์ GPIB มากกว่าหนึ่ง

- โดยอุปกรณ์ gpib0 จะใช้ในการควบคุมติดต่อกับ Voltmeter
- และอุปกรณ์ gpib1 จะใช้ในการควบคุมติดต่อกับ Plotter และ Printer



รูปที่ 2.3 แสดงการเชื่อมต่อระบบโดยใช้อุปกรณ์ GPIB มากกว่าหนึ่ง

## 2.8 การใช้โปรแกรม NI-488.2M กับ Windows 95

การใช้โปรแกรม NI-488.2M กับ Windows 95 ดังแสดงในรูปที่ 2.4 โดยแสดงการทำงานเป็นขั้นตอน โดยจะขึ้นกับการติดตั้งของผู้ใช้ที่จะเลือกใช้ ในที่นี่สามารถแบ่งได้ 2 โหมดการทำงาน

- User Mode คือ โหมดผู้ใช้ ซึ่งสามารถใช้โปรแกรมที่เขียนขึ้นแล้วทำการติดต่อกับในส่วน ของ Kernel Mode โดยการเรียกใช้ gpib-32.dll

- Kernel Mode คือ โหมดการติดต่อกับอุปกรณ์ GPIB โดยใช้โปรแกรมที่มีมากับ Driver ของการ์ดซึ่งเป็น โปรแกรมที่ใช้สำหรับทดสอบอุปกรณ์ว่าต่ออยู่กับระบบหรือไม่ VxD จะหมายถึง Virtual Device Driver ทั่วๆ ไปโดย X จะแสดงถึงชนิดของ Device Driver นั้นตัวอย่างเช่น VDD จะหมายถึง Virtual Device Driver สำหรับอุปกรณ์แสดงผล VTD จะหมายถึง Virtual Device Driver สำหรับอุปกรณ์ Timer และ VPD จะหมายถึง Virtual Device Driver สำหรับเครื่องพิมพ์ Windows

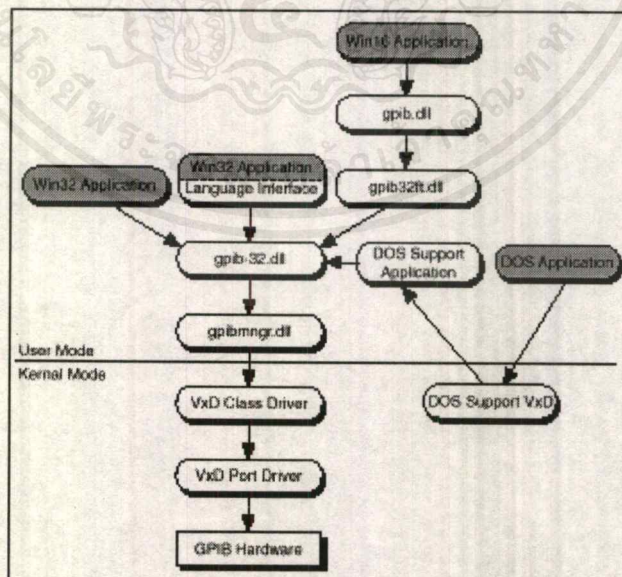
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จะใช้ Virtual Device Driver เพื่อสนับสนุนการทำงานแบบมัลติทาสก์ของ Application แบบ MSDOS based โดยการจำลองส่วนประกอบทางฮาร์ดแวร์บนระบบให้ Application MSDOS เห็น Application แบบ MSDOS based แต่ละตัวเมื่อทำงานภายใต้ Windows95 ดังนั้นจะเรียกแทนด้วยคำว่า MD-DOS Virtual Machine หรือ VM จะทำงานเพื่อเชื่อมต่อกับ Windows ในการ Interrupt และการทำงานของ I/O ให้แก่ Application โดยไม่ทำลายการทำงานของ Application อื่นที่ทำงานอยู่

Windows95 ได้รับประโยชน์จากการจัดเตรียมการสนับสนุน Device Driver จำนวนมาก ซึ่งอยู่ในรูปของ VxD ( Virtual Device Driver ) ในสภาวะแวดล้อมของ Windows แทนที่จะใช้ Device Driver ของ MSDOS ที่ทำงานใน real mode การทำงานที่เคยถูกสนับสนุนแบบเดิมจึงถูกเปลี่ยนมาใช้แบบ VxD ใน Windows95 ซึ่งได้แก่องค์ประกอบดังต่อไปนี้

- ปรับปรุงสมรรถนะการทำงานซึ่งเป็นผลมาจากการใช้โค้ดคำสั่งแบบ 32 บิตและการลดความจำเป็นในการสลับเปลี่ยนไปมาระหว่างการทำงานใน real mode และ protect mode
- ลดการใช้หน่วยความจำปกติโดยการจัดเตรียมการทำงานของ Device Driver และโปรแกรม TSR ใน protect mode
- ปรับปรุงเสถียรภาพและความน่าเชื่อถือในการทำงานให้ดีขึ้น

Virtual Device Driver ใน Windows95 จะถูกบ่งชี้โดยนามสกุลไฟล์ .VXD ส่วนของ Windows3.1 จะถูกบ่งชี้โดยนามสกุลไฟล์ .386



รูปที่ 2.4 แสดงการใช้ NI-488.2M กับ Windows 95

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

## 2.9 ฟังก์ชันในการติดต่อกับ NI-488.2

**2.9.1 Tibclr = function(ud : integer): integer; stdcall;**

- Tibclr คือการเคลียร์สถานะของอุปกรณ์
- ud คือการบอกตำแหน่งชนิดของอุปกรณ์

**2.9.2 Tibdev = function(ud : integer;**

**pad : integer;**

**sad : integer;**

**tmo : integer;**

**eot : integer;**

**eos : integer): integer; stdcall;**

- Tibdev คือการติดต่อและกำหนดค่าเริ่มต้นให้กับอุปกรณ์
- ud คือการบอกตำแหน่งชนิดของอุปกรณ์
- pad คือการบอกตำแหน่งหลัก GPIB ของอุปกรณ์
- sad คือการบอกตำแหน่งสำรอง GPIB ของอุปกรณ์
- eot คือการบอกตำแหน่งสิ้นสุดในการติดต่อ
- eos คือการบอกตำแหน่งข้อความสุดท้าย

**2.9.3 Tibrd = function(ud : integer;**

**var rdbuf;**

**cnt : longint): integer; stdcall;**

- Tibrd คือการอ่านข้อมูลจากอุปกรณ์เข้ามาเก็บไว้ในหน่วยความจำ
- ud คือการบอกตำแหน่งชนิดของอุปกรณ์
- rdbuf คือชื่อของหน่วยความจำที่ใช้เก็บข้อมูล
- cnt คือการกำหนดขนาดของการอ่านข้อมูล

**2.9.4 Tibtrg = function(ud : integer) : integer; stdcall;**

- Tibtrg คือการเลือกชนิดของอุปกรณ์
- ud คือการบอกตำแหน่งชนิดของอุปกรณ์

**2.9.5 Tibwrt = function(ud : integer;**

**var wrtbuf;**

**cnt : longint): integer; stdcall;**

- Tibwrt คือการส่งข้อมูลจากผู้ไปให้อุปกรณ์

- ud คือการบอกตำแหน่งชนิดของอุปกรณ์
- wrtbuf คือชื่อของหน่วยความจำที่ใช้ในการส่งข้อมูล
- cnt คือการกำหนดขนาดของการส่งข้อมูล

**2.9.6 Tibwait = function(ud : integer;**

**mask: integer): integer; stdcall;**

- Tibwait คือการรอเวลาในการติดต่อหรือการดำเนินงานของ GPIB
- ud คือการบอกตำแหน่งชนิดของอุปกรณ์
- mask คือบิตแสดงสถานะของ GPIB ที่บอกถึงเหตุการณ์ในการรอคอยในการติดต่อ

กับ GPIB

**2.9.7 TDevClear = procedure (boardID: integer;**

**address: word); stdcall;**

- TDevClear คือการเคลียร์สถานะของอุปกรณ์
- boardID คือการบอกตำแหน่งของบอร์ด GPIB ที่จะทำการติดต่อ
- address คือตำแหน่งของอุปกรณ์ที่ต้องการจะเคลียร์

**2.9.8 TFindLstn = procedure (boardID: integer;**

**padlist: AddrList;**

**resultlist: AddrList;**

**limit: integer); stdcall;**

- TFindLstn คือการหาการตอบรับของอุปกรณ์บนบัส GPIB
- boardID คือการบอกตำแหน่งของบอร์ด GPIB
- padlist คือรายการของตำแหน่งหลัก
- resultlist คือผลลัพธ์ที่ได้จากการตรวจสอบทั้งหมดจะเก็บข้อมูลในรูปแบบของ array
- limit คือจำนวนที่ต้องการเก็บไว้ใน resultlist

**2.9.9 Tibonl = function (ud: integer;**

**v: integer) : integer; stdcall;**

- Tibonl คือการแสดงถึงอุปกรณ์หรือ GPIB ว่าทำงานหรือไม่ทำงาน
- ud คือการแสดงถึงบอร์ดหรืออุปกรณ์
- v คือถ้ามีค่าเป็น '1' จะเป็นการบอกว่าบอร์ดหรืออุปกรณ์นั้นทำงานและถ้ามีค่าเป็น '0' คือ

ไม่ทำงาน

**2.9.10 TReadStatusByte = procedure (boardID: integer;**

**address: word;**

**var result: integer); stdcall;**

- TReadStatusByte คือการอ่านสถานะของอุปกรณ์
- boardID คือการบอกตำแหน่งการติดต่อ GPIB
- address คือการบอกตำแหน่งของอุปกรณ์
- result คือการกำหนดตำแหน่งในการเก็บข้อมูลในรูปของไบต์

**2.9.11 TReceive = procedure (boardID: integer;**

**address: word;**

**var buffer;**

**cnt: longint;**

**termination: integer); stdcall;**

- TReceive คือการอ่านข้อมูลจากอุปกรณ์
- boardID คือการบอกตำแหน่งการติดต่อ GPIB
- address คือการบอกตำแหน่งของอุปกรณ์
- buffer คือชื่อของตำแหน่งที่ใช้ในการเก็บข้อมูล
- cnt คือความยาวในการอ่านข้อมูล

**2.9.12 TSend = procedure (boardID: integer;**

**address: word;**

**var buffer;**

**datacnt: longint;**

**eotmode: integer); stdcall;**

- Tsend คือการส่งข้อมูลไปให้อุปกรณ์
- boardID คือการบอกตำแหน่งการติดต่อ GPIB
- address คือการบอกตำแหน่งของอุปกรณ์
- buffer คือชื่อของตำแหน่งที่ใช้ในการส่งข้อมูล
- datacnt คือความยาวในการส่งข้อมูล
- eotmode คือการบอกตำแหน่งสุดท้ายในการส่งข้อมูล

**2.9.13 TSendIFC = procedure (boardID: integer); stdcall;**

- TSendIFC คือการรีเซ็ต GPIB โดยการส่งสัญญาณ interfaceเคลียร์
- boardID คือการบอกตำแหน่งการติดต่อ GPIB

**2.9.14 TWaitSRQ = procedure (boardID: integer;  
var result: integer); stdcall;**

- TWaitSRQ คือการคอยจนกระทั่งอุปกรณ์ยืนยันพร้อมที่จะขอบริการจาก GPIB
- boardID คือการบอกตำแหน่งการติดต่อ GPIB
- result คือสถานะของผลลัพธ์จะเป็น '1' เมื่อมีการยืนยันและจะเป็น '0' เมื่อไม่มีการยืนยัน



## บทที่ 3

### การใช้โปรแกรม NI-488.2M กับ 32-bit Delphi Application

#### บทนำ

โปรแกรม Borland Delphi เป็นโปรแกรมที่สามารถใช้สร้าง Application บน Windows ได้ ง่ายและรวดเร็ว โดยถูกพัฒนามาจากภาษา Pascal Delphi สามารถที่จะเรียกใช้ dynamic link libraries (DLLs) ซึ่งทำให้เป็นเรื่องง่ายที่เราจะนำเอาโปรแกรม Delphi มาใช้ติดต่อกับอุปกรณ์ GPIB โดยผ่านโปรแกรม NI-488.2M ซึ่งต่อไปนี้เป็นวิธีการเรียกใช้ NI-488.2M DLL โดยโปรแกรม Delphi

#### 3.1 การเขียนโปรแกรมบน NI-488 หรือ NI-488.2

NI-488 และ NI-488.2 เป็น Software Driver ของการ์ด GPIB-PCII/ IIA ซึ่งสามารถนำมาใช้งานกับ Delphi ได้ทั้ง 2 โปรแกรมแตกต่างกันเพียงแต่รูปแบบของ function และ procedure ที่มีให้เรียกใช้งานอยู่ในไฟล์ .DLL ดังจะได้แสดงตัวอย่างการใช้งานของโปรแกรมทั้ง 2 ในหัวข้อต่อไป

32 bit Delphi Application สามารถที่จะเชื่อมต่อกับ NI-488.2M dynamic link library (gpib-32.dll) โดยการเรียกใช้ gpib-32.dll ดังนี้

##### 3.1.1 การสร้างตัวแปรที่ใช้ชี้ตำแหน่งของ function และใช้บ่งบอกถึงสถานะ (Export Pointers to Status Variables and Function Names)

32 bit GPIB DLL จะให้ค่าชี้ใช้บอกตำแหน่งของ NI-488.2 function subroutines และตัวแปรแบบโคบอล การชี้บอกตำแหน่งของตัวแปรแบบโคบอลของ NI-488.2 (ibsta, iberr, ibcntl) สามารถที่จะทำได้โดยการสร้างตัวแปรขึ้นมารับค่า

```
var    user_ibsta ,
        user_iberr ,
        user_ibcntl : integer ;
```

การเรียกใช้ฟังก์ชันอย่างเช่น ibbna, ibfind, ibrdf และ ibwrt หรือฟังก์ชันทั้งหมดของ NI-488.2 และ subroutines ทั้งหมดจะถูกเรียกใช้ได้โดยการนำค่าออกมาจาก 32 bit GPIB DLL ซึ่งก็หมายความว่าเราสามารถที่จะใช้งานฟังก์ชันเหล่านี้ในการควบคุมอุปกรณ์ โดยเราจำเป็นต้องรู้ตำแหน่งของฟังก์ชันที่เราจะเรียกใช้ทำได้โดยการใช้คำสั่ง GetProcAddress ทำการผ่านชื่อของฟังก์ชันมายังตัวแปรและเราจะใช้ตัวแปรนี้เป็นตัวรับค่าต่างๆที่เราจะใช้ควบคุมอุปกรณ์ตามการทำงาน

ของแต่ละฟังก์ชัน คำอธิบายการทำงานของแต่ละฟังก์ชันสามารถดูเพิ่มได้จากคู่มือ NI-488.2M Function Reference Manual For Win32 ( downloadได้ที่ [ftp.natinst.support.com](http://natinst.support.com))

ฟังก์ชันต่างๆที่เราใช้งานจำเป็นต้องมีการป้อนค่าต่างๆ ให้แก่ฟังก์ชันเหล่านี้ เช่น ฟังก์ชัน `ibbna` ต้องป้อนชื่อของการ์ด( การตั้งชื่อการ์ดดูได้จากคู่มือการติดตั้งชื่อ Getting Started with Your GPIB-PCII/ IIA and The GPIB Software For Windows95) ฟังก์ชัน `ibfind` ต้องการชื่อการ์ดและอุปกรณ์และ ฟังก์ชัน `ibrd` กับ `ibwrt` ต้องใช้ชื่อของไฟล์ แต่จากการที่ Windows NT สามารถเข้าใจได้ทั้งแบบ normal (8บิต) และ unicode (16บิต) character ซึ่ง `gplib-32.dll` จะส่งค่าออกมาทั้ง 2 แบบ แต่ใน Windows 95 เราไม่สามารถใช้ character ขนาด 16บิตได้ ดังนั้นสำหรับ Windows 95 เราจะใช้ฟังก์ชัน 8บิตASCII เช่น `ibbnaA`, `ibrdA`, `ibwrtA`

ในการเพิ่มตัวชี้ถึงตัวแปรที่บอกสถานะของอุปกรณ์และการโหลด `gplib-32.dll` เราต้องทำการกำหนดชนิดข้อมูลสำหรับฟังก์ชันที่เราต้องการเรียกใช้ ชนิดของข้อมูลของแต่ละฟังก์ชันที่ส่งผ่านโดย `gplib-32.dll` สามารถดูได้จากคู่มือชุดเดียวกับที่อธิบายการทำงานของแต่ละฟังก์ชันและสามารถหาข้อมูลเพิ่มเติมได้จาก WIN 32 SDK

### 3.1.2 รายการต่างๆที่ต้องกำหนดใน Application ( Items to Include)

ใน Win 32 Delphi Application ขั้นตอนแรกที่เราต้องทำคือใช้ `type` เพื่อกำหนดตัวแปรแบบโคบอลสำหรับ GPIB ตัวแปร GPIBนี้จะใช้กำหนดฟังก์ชันที่จะส่งค่ากลับมาในรูปของจำนวนเต็ม ตามด้วยการเรียกใช้แบบ `directive`, `stdcall` ตัวอย่างข้างล่างนี้เป็นวิธีการใช้ `type` กำหนดตัวแปรแบบโคบอลของ GPIB

`Tibsta = function : integer ; stdcall ;`

`Tiberr = function : integer ; stdcall ;`

`Tibcntl = function : Longint ; stdcall ;`

ในส่วน `var` ของโปรแกรม `AddrIbXXX` จะถูกกำหนดโดยใช้ตำแหน่งที่ได้จากตัวแปรแบบโคบอล และถูกชี้โดย `PibXXX` ตามตัวอย่างดังต่อไปนี้

`AddrIbsta : Tibsta ;`

`AddrIberr : Tiberr ;`

`AddrIbcntl : Tibcntl ;`

`Pibsta : ^integer ;`

`Piberr : ^integer ;`

`Pibcntl : ^Longint ;`

NI-488 จะถูกเรียกใช้โดยถูกกำหนดให้อยู่ในรูปของฟังก์ชันและ NI-488.2 จะถูกเรียกใช้โดยกำหนดให้อยู่ในรูปของ Procedure ในส่วนที่ใช้กำหนดการเรียกใช้งานจะต้องตามด้วยรายการของตัวแปรและพารามิเตอร์ต่างๆ ของฟังก์ชันที่เรียกใช้ โดยที่ส่วนประกอบสุดท้ายจะเป็นการเรียกแบบ directive (stdcall) การเพิ่มส่วน TYPE เพื่อเพิ่มรูปแบบข้อมูลที่จะใช้กับฟังก์ชันของ GPIB ใน Pascalยูนิค สามารถดูรูปแบบได้ตามตัวอย่างที่ได้ให้ไว้ในส่วนท้ายของบทนี้ ต่อไปนี้จะเป็นตารางที่ใช้ในการเปรียบเทียบรูปแบบของภาษา C กับรูปแบบของ Delphi

| C-Style Declaration | Delphi-Style Declaration                                                    | Description:                                                                 |
|---------------------|-----------------------------------------------------------------------------|------------------------------------------------------------------------------|
| Addr4882_t num      | num : Smallint                                                              | Short integer variable passed by value                                       |
| Addr4882_t *list    | type<br>AddrList = array[0..31] of Smallint<br>var results : AddrList       | Array of short integers                                                      |
| char *udname        | udname : packed array[0..10] of char                                        | Array of characters passed by reference (e.g., for lbbna and lbfina calls)   |
| char *byteBuf       | var byteBuf: byte                                                           | Unsigned byte variable passed by reference (e.g., for lbrpp and lbrsp calls) |
| int num             | num : integer                                                               | Integer variable passed by value                                             |
| int *num            | var num : integer                                                           | Integer variable passed by reference                                         |
| long num            | num : longint                                                               | Long integer variable passed by value                                        |
| short *results      | type<br>ResultsList = array[0..31] of Smallint<br>var results : ResultsList | Array of short integers                                                      |
| short *num          | var num : Smallint                                                          | Short integer variable passed by reference                                   |
| unsigned short num  | num : word                                                                  | Unsigned short integer variable passed by value                              |
| void *buffer        | var buffer                                                                  | String passed by reference                                                   |

### รูปที่ 3.1 แสดงตารางเปรียบเทียบของภาษา C กับภาษา Delphi

ต่อไปนี้เป็นตัวอย่างของการกำหนด ฟังก์ชัน ibdev ในส่วนของ Type

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

ตัวอย่างการกำหนด SendIFC โปรซีเจอร์ในส่วนของ Type

```
TSendIFC = Procedure ( boardID : integer ) ; stdcall ;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

- ตัวอย่างการเรียกใช้โดยกำหนดในส่วนของการสร้างตัวแปร Var

**ibdev : Tibdev ;**

**SendIFC : TSendIFC ;**

นอกจากนี้เรายังต้องกำหนดค่า handle สำหรับ GPIB ไลบรารีไว้ในส่วนของ Var ตามตัวอย่างต่อไปนี้

**Gpib32Lib : Thandle ;**

ในส่วนของ Const ซึ่งเป็นส่วนที่ใช้ประกาศค่าคงที่ในโปรแกรมที่เขียนขึ้น เราสามารถที่จะกำหนดค่าที่ใช้เป็นตัวบอกสถานะของระบบและการเกิด Error ขึ้นในการติดต่อ ซึ่งค่าต่างๆนี้จะมีค่าตามที่ถูกกำหนดไว้ในคู่มืออ้างอิงของ NI-488 ค่าคงที่เหล่านี้สามารถใช้ในโปรแกรมของเราเพื่อช่วยในการตรวจสอบ โดยนำมาเปรียบกับค่าที่อยู่ใน ibsta และ iberr ซึ่งค่าคงที่ต่างๆมีดังนี้

|                                        |                  |                                                    |
|----------------------------------------|------------------|----------------------------------------------------|
| <b>const</b>                           |                  |                                                    |
| <b>(* GPIB status bit definitions.</b> |                  | <b>*)</b>                                          |
| <b>ERR</b>                             | <b>= \$8000;</b> | <b>(* Error detected</b> <b>*)</b>                 |
| <b>TIMO</b>                            | <b>= \$4000;</b> | <b>(* Timeout</b> <b>*)</b>                        |
| <b>ENDgpib</b>                         | <b>= \$2000;</b> | <b>(* EOI or EOS detected</b> <b>*)</b>            |
| <b>SRQI</b>                            | <b>= \$1000;</b> | <b>(* SRQ detected by CIC</b> <b>*)</b>            |
| <b>RQS</b>                             | <b>= \$800;</b>  | <b>(* Device needs service</b> <b>*)</b>           |
| <b>SPOLL</b>                           | <b>= \$400;</b>  | <b>(* Board has been serially polled</b> <b>*)</b> |
| <b>EVENT</b>                           | <b>= \$200;</b>  | <b>(* An event has occurred</b> <b>*)</b>          |
| <b>CMPL</b>                            | <b>= \$100;</b>  | <b>(* I/O completed</b> <b>*)</b>                  |
| <b>LOK</b>                             | <b>= \$80;</b>   | <b>(* Local lockout state</b> <b>*)</b>            |
| <b>REM</b>                             | <b>= \$40;</b>   | <b>(* Remote state</b> <b>*)</b>                   |
| <b>CIC</b>                             | <b>= \$20;</b>   | <b>(* Controller-in-charge</b> <b>*)</b>           |
| <b>ATN</b>                             | <b>= \$10;</b>   | <b>(* Attention asserted</b> <b>*)</b>             |
| <b>TACS</b>                            | <b>= \$8;</b>    | <b>(* Talker active</b> <b>*)</b>                  |
| <b>LACS</b>                            | <b>= \$4;</b>    | <b>(* Listener active</b> <b>*)</b>                |
| <b>DTAS</b>                            | <b>= \$2;</b>    | <b>(* Device trigger state</b> <b>*)</b>           |
| <b>DCAS</b>                            | <b>= \$1;</b>    | <b>(* Device clear state</b> <b>*)</b>             |

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

(\*      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               | *) |

### 3.1.3การโหลด GPIB-32.DLL Library

ในโปรแกรม Win 32 ของเรา เราต้องโหลด GPIB Library( gpib-32.dll)Code ที่แสดงต่อไปนี้  
 นี้ จะแสดงวิธีการโดยการเรียกใช้ฟังก์ชัน LoadLibrary และตามด้วยการตรวจสอบข้อผิดพลาด

```
Gpib32Lib := LoadLibrary('GPIB-32.DLL');
```

```
If Gpib32Lib = 0 Then
```

```
begin
```

```
    (* Report an error here. *)
```

```
end ;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
 ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

### 3.1.4 การหาตำแหน่ง ( Getting the Addresses)

หลังจากที่มีการกำหนดค่าคงที่และมีการสร้างรูปแบบข้อมูลในส่วนของ Type แล้วต่อไปก็คือการหาตำแหน่งของตัวแปร โคบอล ที่ใช้บอกสถานะและฟังก์ชันที่เราจำเป็นต้องใช้ในโปรแกรมของเรา Codeที่จะแสดงต่อไปนี้ เป็นตัวอย่างที่แสดงวิธีการเอาตำแหน่งของตัวแปรมาใช้

```
@AddrIbsta := GetProcAddress(Gpib32Lib, 'user_ibsta');
@AddrIberr := GetProcAddress(Gpib32Lib, 'user_iberr');
@AddrIbcntl := GetProcAddress(Gpib32Lib, 'user_ibcntl');

Pibsta := @AddrIbsta;
Piberr := @AddrIberr;
Pibcntl := @AddrIbcntl;

(* NI-488 Calls)
@ibdev := GetProcAddress(Gpib32Lib, 'ibdev');
@ibonl := GetProcAddress(Gpib32Lib, 'ibonl');

(* NI-488.2 Calls)
@SendIFC := GetProcAddress(Gpib32Lib, 'SendIFC');
```

ถ้าฟังก์ชัน GetProcAddress เกิดข้อผิดพลาด มันจะส่งค่า NIL กลับมาที่พอยเตอร์ Codeต่อไปนี้จะแสดงวิธีการตรวจสอบว่า ฟังก์ชัน GetProcAddress เกิดข้อผิดพลาดหรือไม่

```
If (@AddrIbsta = NIL) Or
(@AddrIberr = NIL) Or
(@AddrIbcntl = NIL) Or
(@ibdev = NIL) Or
(@ibonl = NIL) then
begin
```

```
(* Report an Error. *)
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
FreeLibrary( Gpib32Lib); (* Free the library. *)
end;
```

### 3.1.5 การติดต่อกับอุปกรณ์ GPIB และตัวแปร( Accessing the GPIB Calls and Variables)

ชุดคำสั่งต่อไปนี้ จะแสดงวิธีการเรียกใช้ฟังก์ชันและโพรซีเจอร์ รวมทั้งตัวแปรแสดงสถานะภายในโปรแกรมที่เราเขียนขึ้น

```
dev := ibdev(0,1,0,13,1,0);
if (Pibsta^ and ERR) <> 0 then
  gpiberr('ibdev Error');

SendIFC(0);
if (Pibsta^ and ERR) <> 0 then
  gpiberr('SendIFC Error');
```

### การยกเลิกการใช้ Library (Freeing the Library)

ก่อนที่จะออกจากโปรแกรมเราจะต้องทำการยกเลิกการติดต่อกับ gpib-32.dll ก่อน เพื่อให้โปรแกรมอื่นสามารถเรียกใช้ gpib-32.dll ได้ ด้วยชุดคำสั่งต่อไปนี้

```
FreeLibrary(Gpib32Lib);
```

## 3.2 ขั้นตอนการสร้างโปรแกรมประยุกต์

เมื่อเราได้ทำการติดต่อกับ GPIB โดยผ่าน gpib-32.dll โดยการกำหนดสิ่งต่างๆ ตามที่อธิบายในหัวข้อ 3.1 เรียบร้อยแล้ว ในการสร้างโปรแกรมประยุกต์ด้วย Delphi เพื่อติดต่อกับอุปกรณ์ GPIB เราจะสร้างโพรซีเจอร์ (Procedure) ชื่อ LoadDLL (ดู Source Code ได้ในภาคผนวก) ซึ่งจะทำหน้าที่ในการโหลดไลบรารี (Load Library) gpib-32.dll และทำการตรวจสอบความผิดพลาดต่างๆ พร้อมทั้งแสดง Dialog box แจ้งให้ผู้ใช้ทราบเมื่อไม่สามารถโหลดไลบรารี (Load Library) ได้

ถ้าการโหลดไลบรารีสำเร็จ เราก็จะสามารถใช้ฟังก์ชัน และ โพรซีเจอร์ ต่างๆ เพื่อติดต่อกับควบคุมอุปกรณ์ GPIB ได้ โดยสิ่งแรกที่ต้องทำคือ การส่งคำสั่งเพื่อสั่งให้อุปกรณ์ทั้งหมด และตัวเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



คอนโทรล อยู่ในสถานะที่ว่าง (Idle State) โดยใช้คำสั่ง SendIFC สำหรับ NI-488.2 หรือ ibdev สำหรับ NI-488 จากนั้นก็จะทำการติดต่อกับอุปกรณ์ โดยในการติดต่อก็จะต้องมีการอ้างถึงตำแหน่งของอุปกรณ์ แต่ละตัว ซึ่งจะต้องไม่ซ้ำกัน โดยในโครงงานนี้จะใช้อุปกรณ์ 4 ตัวคือ

Function Generator HM8130 ซึ่งมีค่า Address เป็น 12

Power Supply HM8142 ซึ่งมีค่า Address เป็น 8

Multimeter HM8112-2 ซึ่งมีค่า Address เป็น 7

Oscilloscope HM1007 ซึ่งมีค่า Address เป็น 11

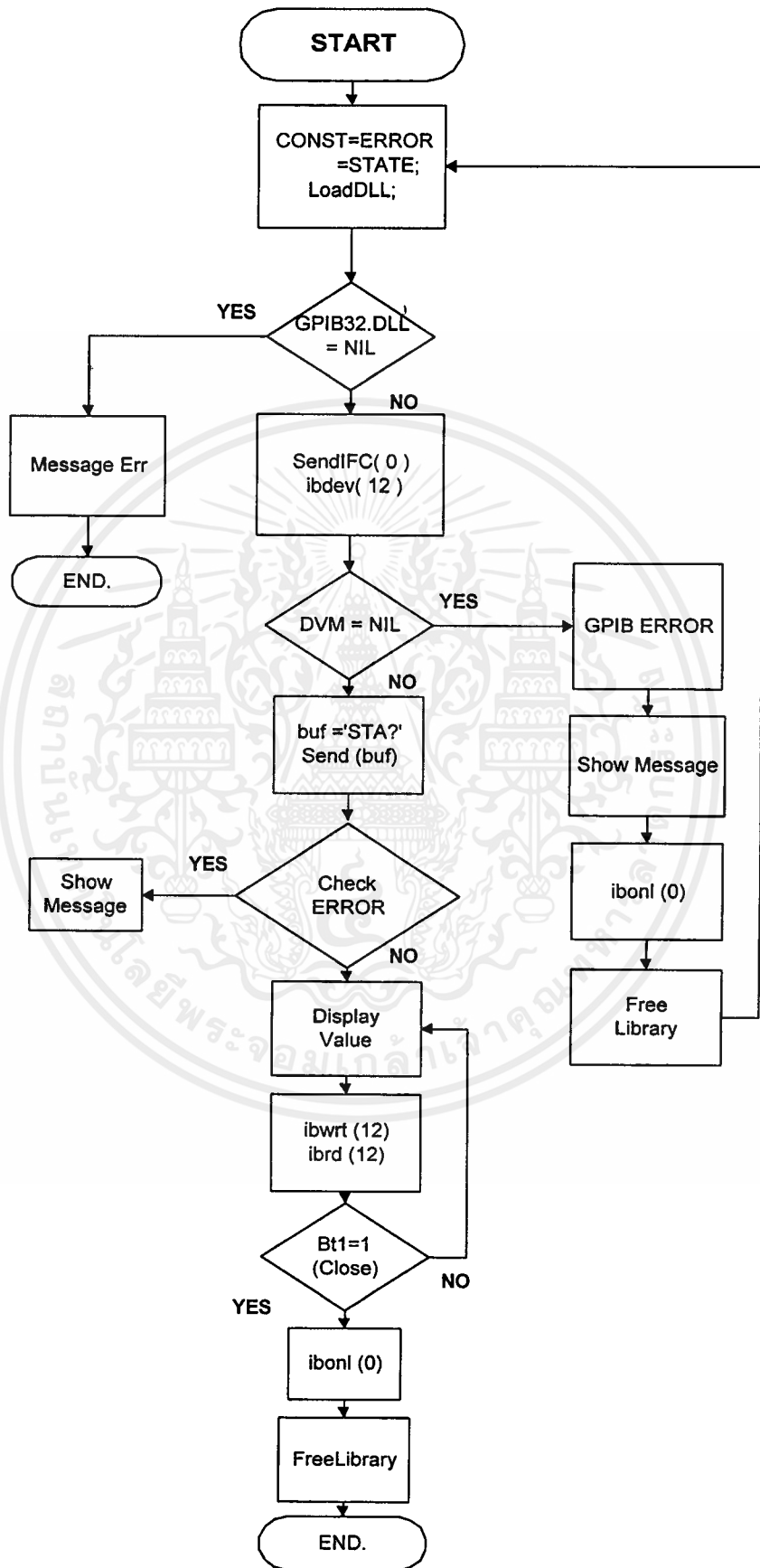
ค่าตำแหน่งของอุปกรณ์แต่ละตัว ที่ใช้นี้เราสามารถทำการเปลี่ยนแปลงได้ โดยการเซต dip switch ที่ตัวอุปกรณ์ นอกจากค่าของตำแหน่งที่ต้องกำหนดแล้ว ยังมีชุดคำสั่งเฉพาะของอุปกรณ์แต่ละตัว ที่เราต้องส่งไปด้วย ซึ่งชุดคำสั่งนี้เราสามารถศึกษาได้จากคู่มือของอุปกรณ์แต่ละตัว

ในโครงงานนี้เราได้ทดลองติดต่อกับอุปกรณ์ทั้ง 4 ตัว โดย เขียนโปรแกรมที่ทำให้ผู้ใช้สามารถที่จะควบคุม แล้วตั้งค่าต่างๆของอุปกรณ์ ผ่านตัวโปรแกรมที่เขียนขึ้นได้ ซึ่งจะได้อธิบายแต่ละอุปกรณ์ดังนี้

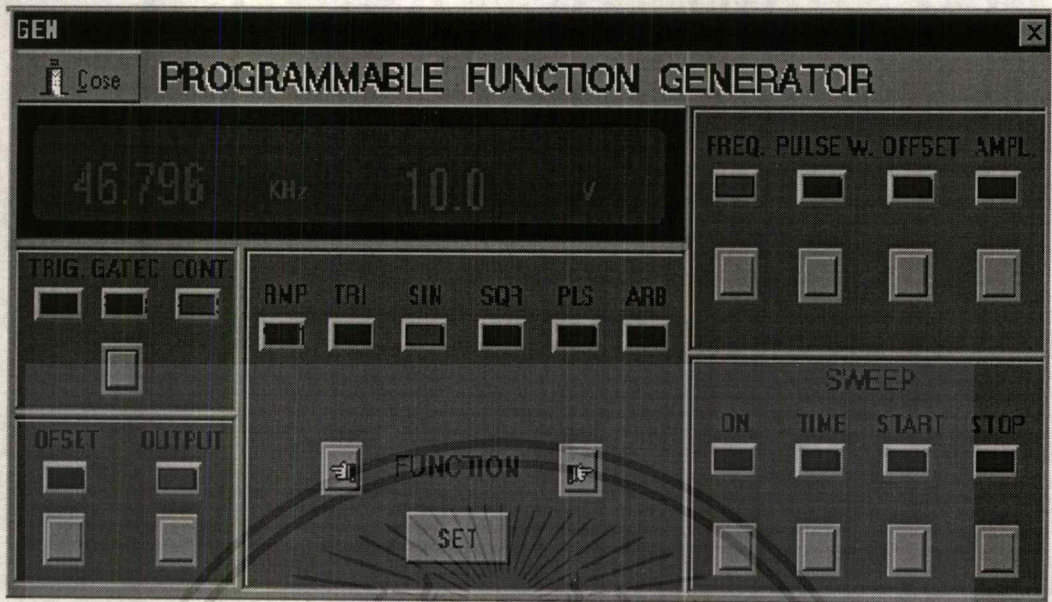
### 3.2.1 Function Generator HM8130

ในการเขียนโปรแกรมควบคุมการทำงานของฟังก์ชันเจนเนอเรเตอร์ เริ่มจากการส่งคำสั่ง SendIFC เพื่อกำหนดให้อุปกรณ์ทุกตัวอยู่ในสถานะที่ว่าง แล้งใช้คำสั่ง ibfind เพื่อทำการตรวจสอบว่าพบอุปกรณ์ HM8130 หรือไม่ ถ้าเกิดข้อผิดพลาด ก็จะแจ้งให้ผู้ใช้ทราบถึงชนิดของข้อผิดพลาดที่เกิดขึ้น การทำงานของโปรแกรมจะเริ่มจากการตรวจสอบสถานะเดิมของอุปกรณ์แล้วจึงทำการส่งค่านี้นอกแสดงผลที่ตัวโปรแกรม หน้าจอคอมพิวเตอร์ ซึ่งลำดับการติดต่อต่างๆ สามารถแสดงได้ด้วย Flow chart ดังรูปที่ 3.2

### รูปที่ 3.2แสดง Flow Chart การทำงาน Function Generator HM 8130



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

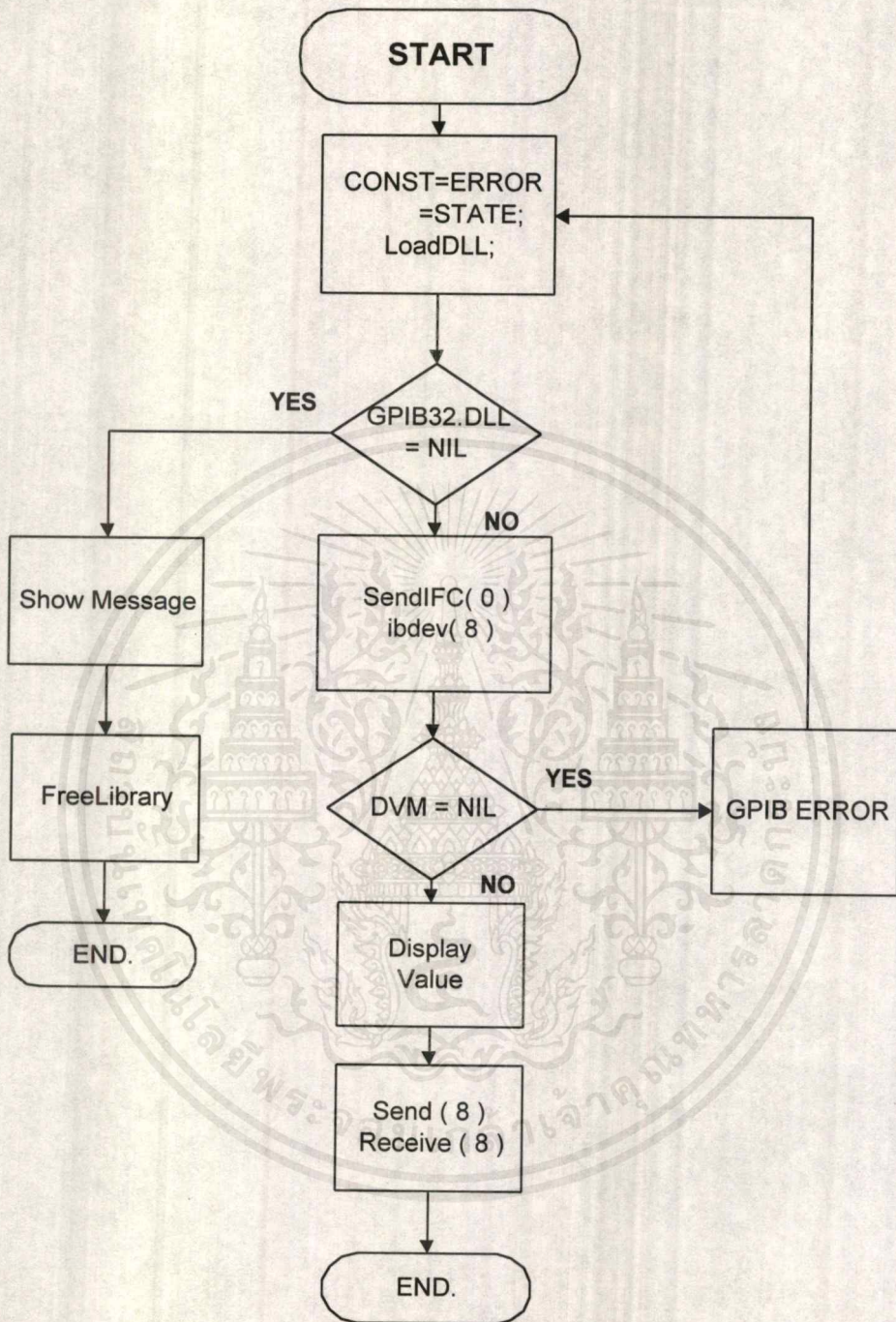


รูปที่ 3.3 ส่วนควบคุม Function Generator

### 3.2.2 Power Supply HM8142

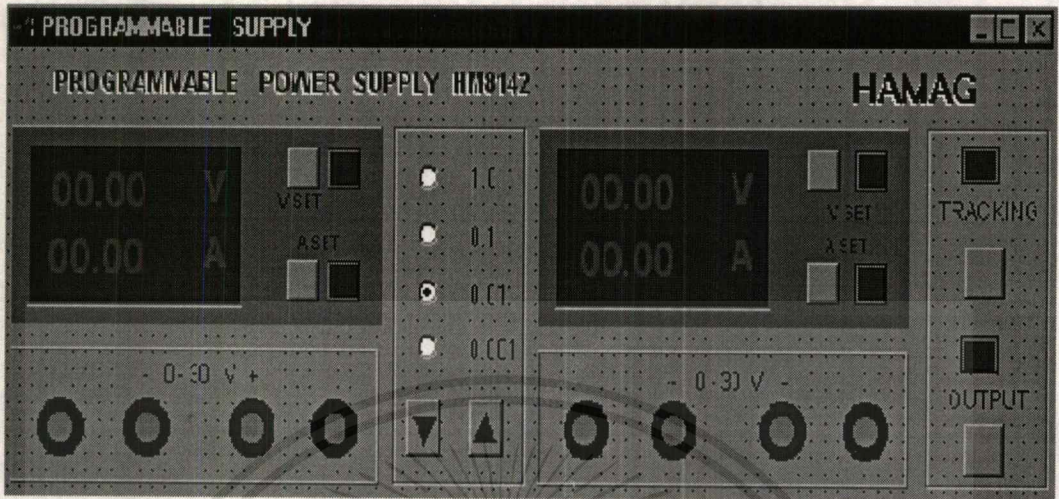
ในส่วนของ power supply จะมีค่า address เป็น 8 ลำดับขั้นตอนในการติดต่อสามารถทำได้ดังนี้

- ทำการกำหนดค่า const และตัวแปรรับค่า address ของฟังก์ชัน NI-488
- ทำการโหลด gpib-32.dll พร้อมทั้งตรวจสอบความผิดพลาด
- ทำการตรวจสอบว่า power supply ต่ออยู่กับระบบหรือไม่ โดยใช้คำสั่ง SendIFC และ ibdev พร้อมทั้งตรวจสอบค่าที่ส่งกลับมาจากฟังก์ชัน ibdev
- เมื่อติดต่อกับอุปกรณ์ได้ จะทำการตรวจสอบ สถานะของอุปกรณ์ นำค่าที่ตรวจสอบได้ ออกแสดงบนโปรแกรม
- รอคอยรับเหตุการณ์ต่างๆ ที่เกิดขึ้นกับ คอนโทรลต่างๆ ที่ได้เขียนชุดคำสั่งให้บริการไว้
- จบโปรแกรม



รูปที่ 3.4 แสดง Flow Chart การทำงาน Power Supply HM 8142

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



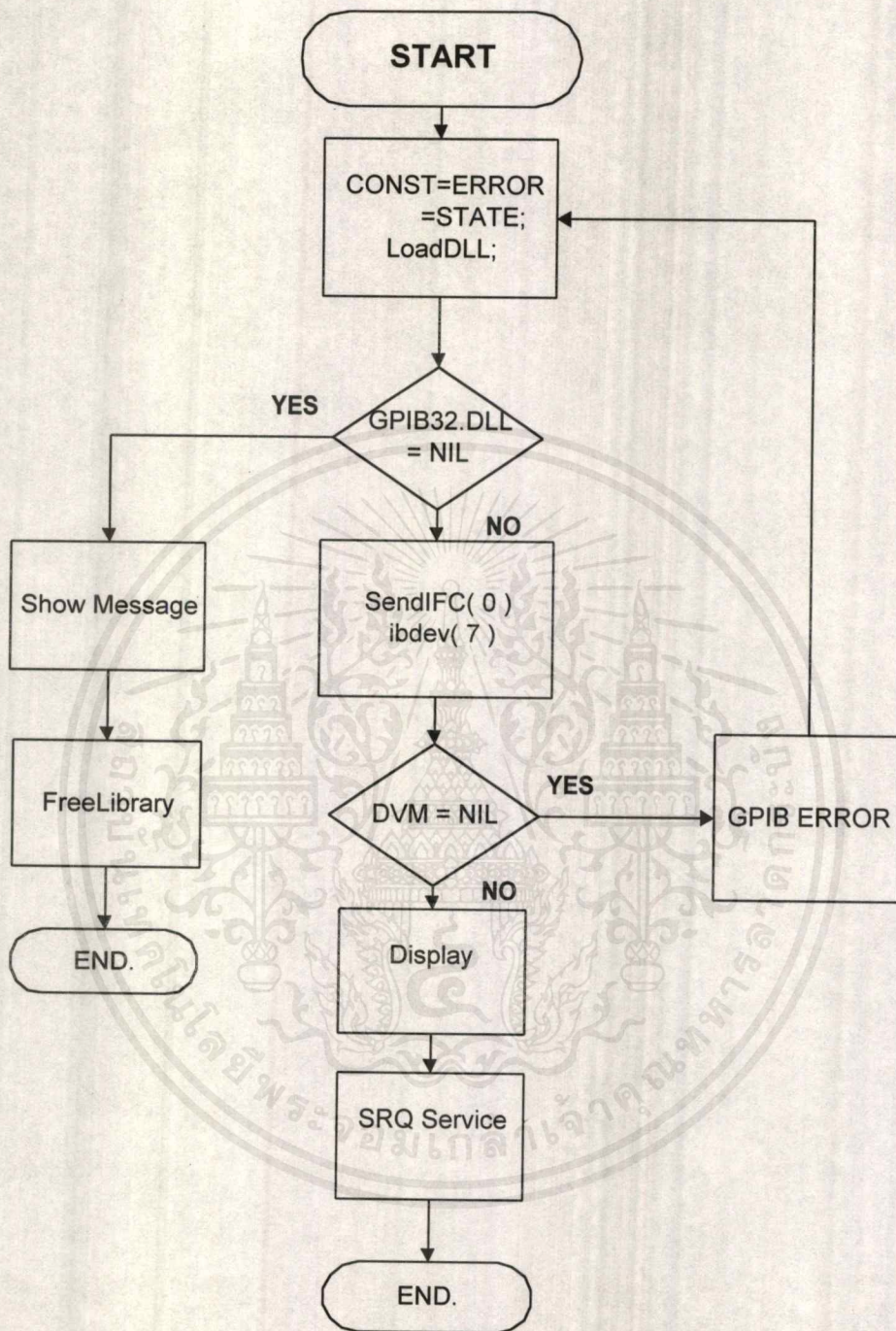
รูปที่ 3.5 ส่วนควบคุม Power Supply

### 3.2.3 Multimeter HM8112-2

มัลติมิเตอร์เป็นอุปกรณ์ที่ทำงานช้าที่สุด ในจำนวนอุปกรณ์ที่ได้เขียนโปรแกรมควบคุมทั้งหมด ดังนั้นในการเขียนโปรแกรมควบคุม จึงจำเป็นต้องใช้การหน่วงเวลา ระหว่างช่วงเวลาในการส่งและช่วงเวลาในการรับ อีกทั้งยังต้องเพิ่มฟังก์ชันที่คอยตรวจสอบสัญญาณการขอบริการจากมัลติมิเตอร์

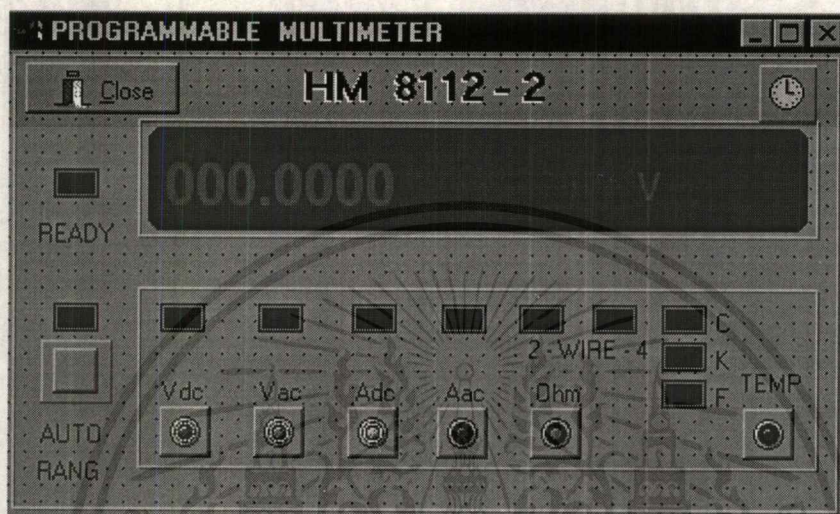
ลำดับขั้นตอนการเขียนโปรแกรมควบคุม มีดังนี้

- กำหนดค่า const ของ Error และตัวแปรของฟังก์ชันที่ต้องเรียกใช้ในโปรแกรม
- ทำการโหลดไบนารี พร้อมทั้งทำการตรวจสอบข้อผิดพลาด ถ้าเกิดข้อผิดพลาดก็จะแสดงให้ผู้ใช้งานทราบ พร้อมทั้งออกจากโปรแกรม
- ทำการเชื่อมต่ออุปกรณ์เข้ากับระบบ พร้อมทั้งส่งคำสั่งอ่านค่าของอุปกรณ์ แล้วนำออกแสดงบนโปรแกรม
- คอยตรวจสอบสัญญาณการขอบริการจากอุปกรณ์ เมื่อมีสัญญาณ โปรแกรมจะทำการอ่านค่าจากอุปกรณ์ แล้วนำออกแสดงทางหน้าจอ
- ออกจากโปรแกรม เมื่อผู้ใช้ กดปุ่มปิด Close



รูปที่ 3.6 แสดง Flow Chart การทำงาน Multimeter HM 8122-2

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.7 ส่วนควบคุม Multimeter

### 3.2.4 Oscilloscope HM 1007

การใช้ Scope ใน storage mode สโคป HM 1007 จะสามารถแปลงสัญญาณจากอนาล็อกเป็นดิจิตอลขนาด 8 บิตได้ทั้ง 2 ช่องสัญญาณ โดยมีค่าความถี่ในการ Sampling สัญญาณสูงสุด 40 MS/s (40 ล้านครั้งต่อวินาที) สัญญาณที่ทำการบันทึกจะเหมือนกับสัญญาณที่ปรากฏอยู่บนหน้าจอของสโคป การแก้ไขรูปคลื่นสามารถทำได้โดยการจัดการกับจุดต่างๆ ที่ถูกบันทึกไว้เป็นรูปกราฟ

เมื่อสโคปทำงานใน storage mode ปุ่ม TIME/DIV จะทำหน้าที่เป็นตัวกำหนดความถี่ของการ Sampling ปุ่มปรับละเอียดจะไม่สามารถใช้งานได้และจากการที่ความถี่ในการ Sampling มีค่า 40 MS/S โดยที่หน่วยความจำที่ใช้เก็บค่าการ Sampling ในแต่ละช่องสัญญาณมีค่า 2 Kbytes ทำให้สามารถเก็บค่าในการ Sampling ได้ประมาณ 2000 ค่าซึ่งจะถูกแสดงบนจอของสโคปที่ถูกแบ่งย่อยออกเป็น 10 ช่องค่าเวลาที่น้อยที่สุดเมื่อสโคปทำงานใน Storage mode นี้คือ 5  $\mu$ S/DIV และสามารถเพิ่มขึ้นสูงสุดได้ 50 ms/DIV โดยการปรับที่ปุ่ม TIM/DIV โดยที่ย่านของเวลาที่สามารถ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปรับได้จะอยู่ในช่วงของเส้นกรอบสีดำที่อยู่รอบๆ ปุ่มและสามารถหน่วงเวลาจาก  $\mu\text{S}$  เป็น  $\text{mS}$  ได้ โดยการกดปุ่มที่อยู่ทางด้านซ้าย

- ความละเอียดในการเก็บรูปสัญญาณทางด้านแนวตั้ง จะมีค่าเท่ากับ  $2^8$  หรือ 256 จุด โดยที่ความสูงของแนวตั้งมีค่า 10 cm ดังนั้นในช่วง 1 cm จะมีจำนวนจุดประมาณ 25 จุด แต่บนจอสโคปจะแสดงไว้เพียง 8 cm โดยแบ่งออกเป็น 8 ช่องเท่านั้น

- ความละเอียดในการเก็บรูปสัญญาณทางด้านแนวนอนจะมีค่าเท่ากับ  $2^{11}$  ซึ่งเท่ากับ 2048 จุด แต่จะมีเพียง 2000 จุด ที่จะถูกแสดงบนจอที่มีความกว้าง 10 cm ซึ่งถูกแบ่งออกเป็น 10 ช่อง ดังนั้นในแต่ละช่องจะมีจำนวนจุดเท่ากับ 200 จุด อีก 48 จุดจะอยู่นอกออกไปจากจอของสโคป นอกจากนี้ใน storage mode เรายังสามารถขยายรูปของสัญญาณตามแนวแกน X ได้โดยการใช้ปุ่ม  $\times 10$  และ  $\times 5$  ที่อยู่บริเวณด้านบนบนใกล้กับปุ่ม power ในการขยายจะทำให้จำนวนจุดต่อช่องของสัญญาณมีค่าลดลง โดยมีค่าเป็น

$$X \times 5 = 40 \text{ dots/DIV}$$

$$X \times 10 = 20 \text{ dots/DIV}$$

$$X \times 50 = 4 \text{ dots/DIV (เมื่อกดปุ่ม } \times 10 \text{ และ } \times 5 \text{)}$$

ความถี่สูงสุดที่สามารถจะทำการเก็บรูปสัญญาณได้ ขึ้นอยู่กับจำนวนของสัญญาณและขนาดของแอมพลิฟายเออร์ช่วงเวลาห่างระหว่างการ Sampling แต่ละครั้งจะขึ้นอยู่กับการปรับ Time-base โดยความถี่ของการ Sampling สูงสุดคือ 40 MS/S โดยการตั้งปุ่ม TIM/DIV ไว้ที่ 5  $\mu\text{S/DIV}$  ดังนั้นในทุกๆ 25 nS จะทำการ Sampling สัญญาณ โดยการกำหนดว่าในช่วงของสัญญาณ 1 คาบ จะต้องทำการ Sampling 10 ครั้ง ดังนั้นเราจะได้เวลาคือ  $10 \times 25 = 250 \text{ nS}$  นั่นคือ สามารถ Sampling สัญญาณที่มีความถี่ได้สูงสุด 4 MHz

### ชุดคำสั่งควบคุม HM 1007

<sp> = Space - Bar

<EOS> = เครื่องหมาย “;”

ID? เป็นคำสั่งที่ใช้ถามหมายเลขอุปกรณ์โดยสโคปจะส่งค่า “HM 408” กลับมาเมื่อมันต่ออยู่กับระบบ

DIG<SP>3<EOS> เป็นคำสั่งที่ใช้สำหรับอ่านข้อมูลที่เก็บอยู่ในหน่วยความจำของสโคปโดยสามารถแยกอ่านได้ที่ช่องสัญญาณ หรืออ่านทั้ง 2 ช่องสัญญาณพร้อมกัน

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

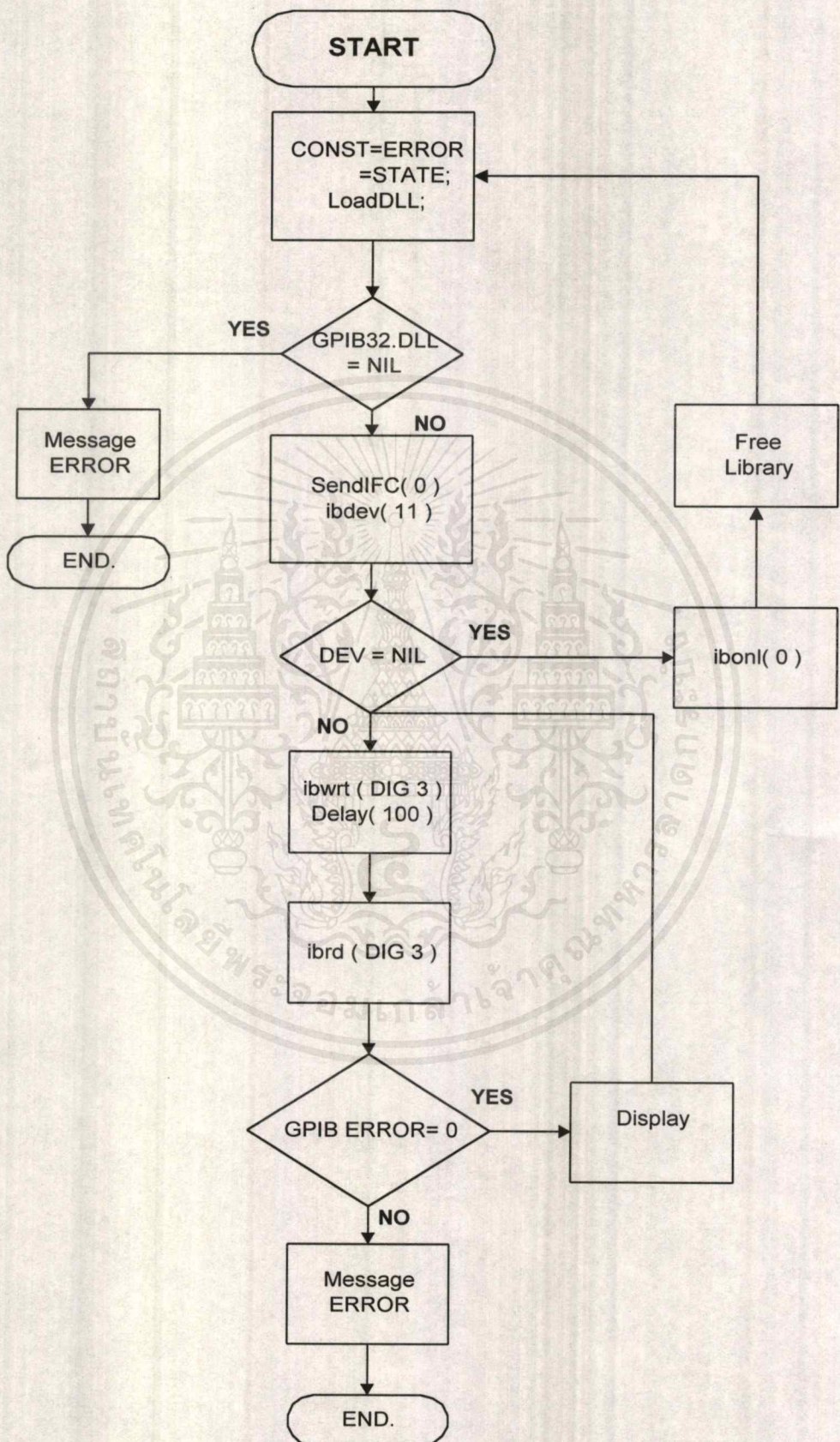
STA เป็นคำสั่งที่ใช้ตรวจสอบสถานะของการตั้งช่องสัญญาณ โดยสโคปจะส่งค่ากลับมาในรูปของรหัส ASCII สามารถดูความหมายในภาคผนวกหรือคู่มือ HO 79-4/7

ในส่วนของ Oscilloscope จะมีตำแหน่ง Address เป็น 11 ลำดับการทำงานมีดังต่อไปนี้

- กำหนดค่า const ของ Error และตัวแปรของฟังก์ชันที่ต้องเรียกใช้ในโปรแกรม
- ทำการโหลดไคบารี พร้อมทั้งทำการตรวจสอบข้อผิดพลาด ถ้าเกิดข้อผิดพลาดก็จะแสดงให้ผู้ใช้งานพร้อมทั้งออกจากโปรแกรม
- การส่งคำสั่ง SendIFC เพื่อกำหนดให้อุปกรณ์ทุกตัวอยู่ในสถานะที่ว่าง แล้งใช้คำสั่ง ibfind เพื่อทำการตรวจสอบว่าพบอุปกรณ์ HM1007 หรือไม่ ถ้าเกิดข้อผิดพลาด ก็จะแจ้งให้ผู้ใช้งานถึงชนิดของข้อผิดพลาดที่เกิดขึ้น
- การทำงานถ้าติดต่อสำเร็จ จะส่งคำสั่ง ibwrt และ delay เพื่อจะหน่วยเวลาในการอ่าน

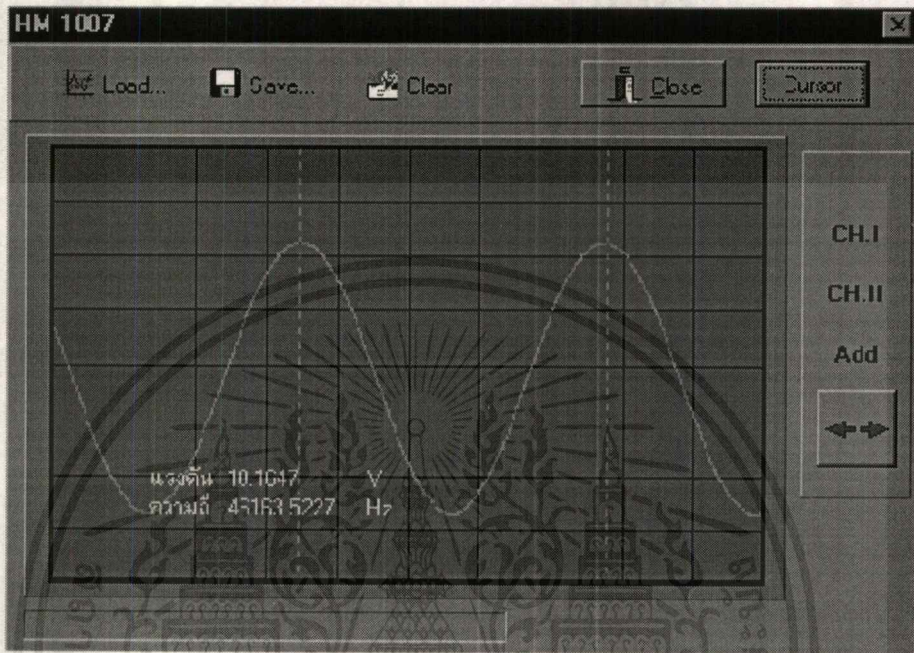
ข้อมูล จากนั้นทำการอ่านข้อมูล ใช้คำสั่ง ibrd

- ทำการตรวจสอบข้อมูล ถ้าพบ Error ก็จะทำการส่งข้อความ Error ออกมา
- ถ้าไม่เกิดข้อผิดพลาด ก็จะแสดงค่าออก Display



รูปที่ 3.8 แสดง Flow Chart การทำงาน Oscilloscope HM 1007

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 3.9 ส่วนควบคุม Oscilloscope

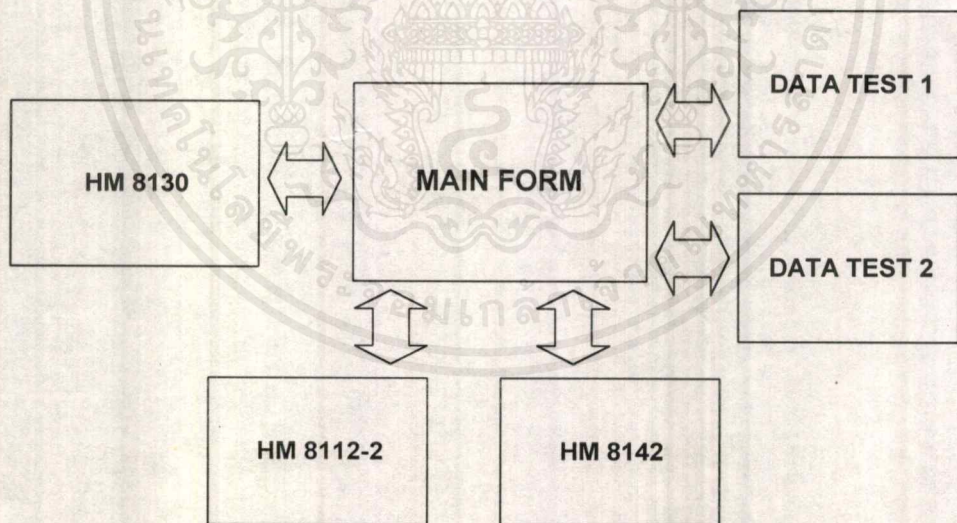
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

## บทที่ 4

### การประยุกต์ใช้งานโปรแกรมควบคุมอุปกรณ์

จากในบทก่อนเราได้ทำการสร้างโปรแกรมที่สามารถทำการควบคุมอุปกรณ์ในแต่ละตัวได้แล้ว ในบทนี้เราจะนำโปรแกรมเหล่านั้นมารวมกับโปรแกรมที่เขียนขึ้นเพิ่มเติมเพื่อใช้ประโยชน์เป็นโปรแกรมที่สามารถวัดหาค่าการตอบสนองความถี่ของระบบหรือวงจรได้ โดยที่ในการตั้งค่าต่างๆในการวัดเราจะเรียกใช้จากโปรแกรมควบคุมอุปกรณ์ ส่วนการเก็บข้อมูลและการแสดงผลจะแสดงในรูปของตารางและกราฟซึ่งจะถูกเก็บอยู่ในฐานข้อมูล เราจะแสดงในโปรแกรมที่เขียนขึ้นเพิ่มเติม

ดังนั้นเราจึงจำเป็นต้องสร้างฟอร์มที่เป็นฟอร์มหลัก ที่สามารถเรียกใช้ ยูนิต แต่ละยูนิตได้ โดยมีการส่งค่าให้และรับค่ากลับในแต่ละยูนิต

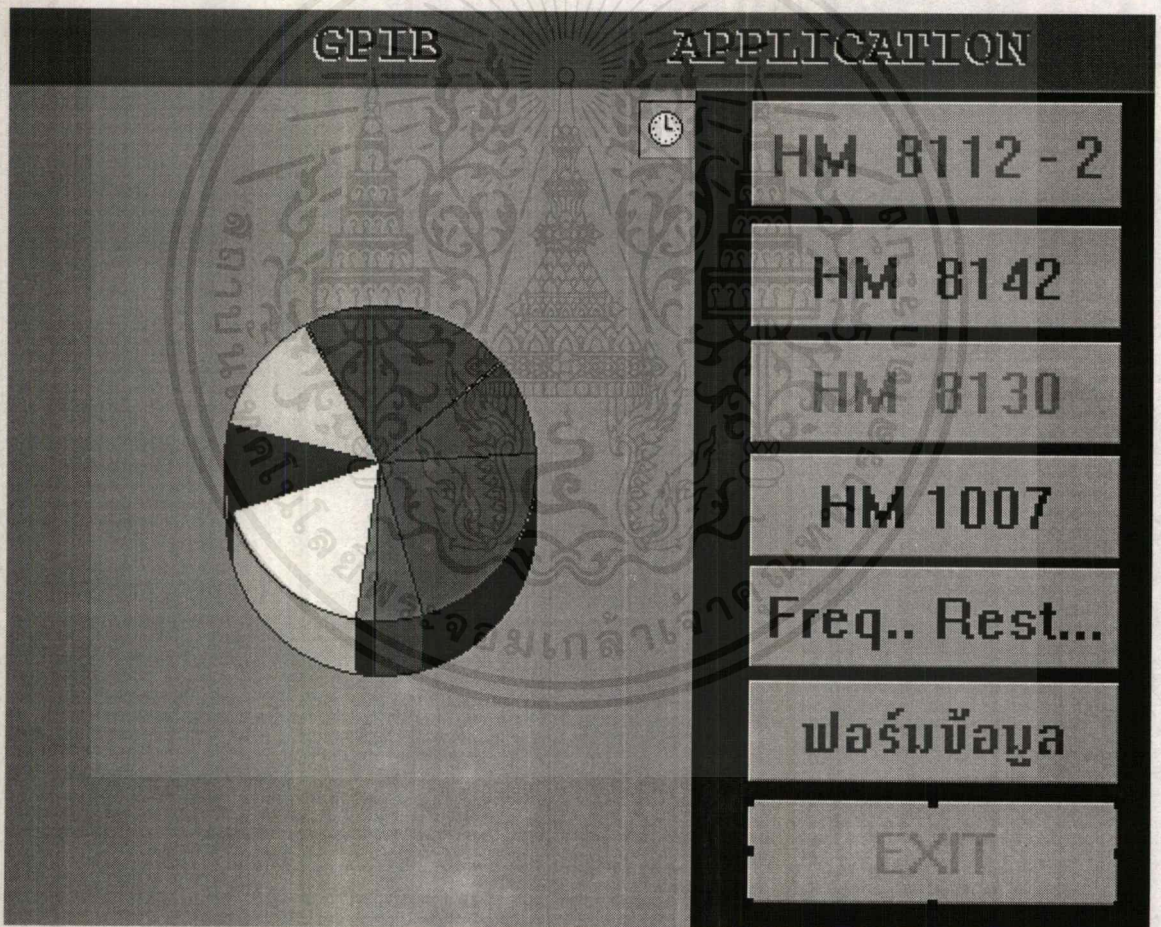


รูปที่ 4.1 การแสดงจำนวนยูนิตย่อยทั้งหมดและการเชื่อมต่อ

ดังนั้นในบทนี้จะต้องมีการสร้างยูนิตเพิ่มขึ้นมาอีก3ยูนิตคือ ยูนิตที่เป็นฟอร์มหลัก,ยูนิตที่เป็นส่วนตั้งค่าและแสดงข้อมูลในขณะ run และยูนิตที่เป็นส่วนเก็บบันทึกและแสดงข้อมูลเก่าที่ได้เก็บไว้

- Data Test 1 ทำหน้าที่แสดงข้อมูลขณะ RUN
- Data Test 2 ทำหน้าที่แสดงข้อมูลเก่า

#### 4.1 ส่วนของ MAIN FORM

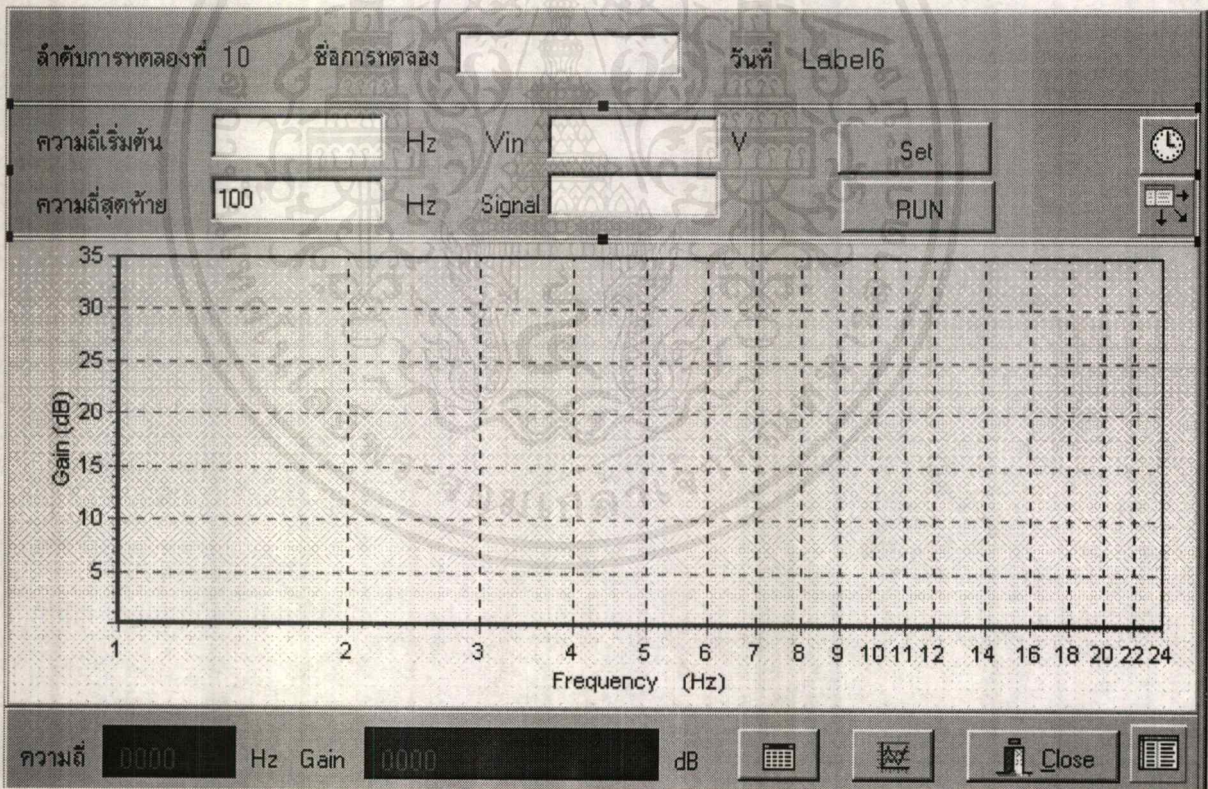


รูปที่ 4.2 แสดงส่วนของ Main Form

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จะเป็นฟอร์มแรกที่ปรากฏขึ้นเมื่อเริ่ม run โปรแกรม บนฟอร์มจะประกอบด้วยปุ่มต่างๆ ที่ใช้เรียกแต่ละยูนิท เมื่อเราต้องการใช้ส่วนใดก็เพียงแต่กดปุ่มที่มีชื่อของอุปกรณ์นั้น ฟอร์มของอุปกรณ์นั้นก็จะปรากฏขึ้นมา ให้เราสามารถใส่ตั้งค่าหรืออ่านค่าต่างๆของอุปกรณ์ได้

#### 4.2 ส่วนแสดงข้อมูลในขณะวัด Frequency Response



รูปที่ 4.3 ส่วนแสดงข้อมูลในขณะวัด Frequency Response

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

จะเป็นฟอร์มที่ใช้ตั้งค่าการวัดหา Frequency ของระบบหรือวงจรจะปรากฏขึ้นเมื่อเรากดปุ่ม Frequency Response บนฟอร์มหลัก บน Frequency Response ฟอร์มจะประกอบไปด้วย

- ส่วนแสดงลำดับการทดลอง ในส่วนนี้จะแสดงว่าการทดลองครั้งนี้เป็นการทดลองครั้งที่เท่าไร โดยนับจากจำนวนผลการทดลองที่ได้มีการบันทึกไว้ก่อนหน้านี้แล้ว

- ส่วนตั้งชื่อการทดลอง ในการตั้งชื่อการทดลอง ถ้าหากมีการตั้งชื่อซ้ำกับชื่อเดิมที่มีอยู่หรือไม่มีการตั้งชื่อการทดลอง โปรแกรมจะแจ้งให้ผู้ใช้ทราบและบอกให้ผู้ใช้ตั้งชื่อใหม่

- ส่วนแสดงวันที่และเวลาที่ทำการทดลอง

- ค่าความถี่เริ่มต้นสามารถตั้งค่าได้โดยการเรียกใช้ ฟังก์ชันเจนเนอเรเตอร์ แล้วทำการตั้งค่า

- ส่วนความถี่สุดท้าย ผู้ทดลองจะเป็นผู้กำหนดโดยถ้าหากมีค่าน้อยกว่าค่าความถี่เริ่มต้น โปรแกรมจะแจ้งให้ผู้ใช้ทราบและให้ผู้ใช้ใส่ค่าใหม่

- ส่วน Vin จะรับค่าที่ตั้งมาจาก ฟังก์ชันเจนเนอเรเตอร์

- ส่วน Signal จะแสดงชื่อของสัญญาณที่ป้อนให้แก่ระบบ โดยถูกตั้งค่ามาจากฟังก์ชันเจนเนอเรเตอร์

- ปุ่ม Set เป็นปุ่มที่ใช้เรียกฟังก์ชันเจนเนอเรเตอร์ และ Multimeter เพื่อใช้ตั้งค่าต่าง ๆ ที่จะป้อนให้กับระบบ

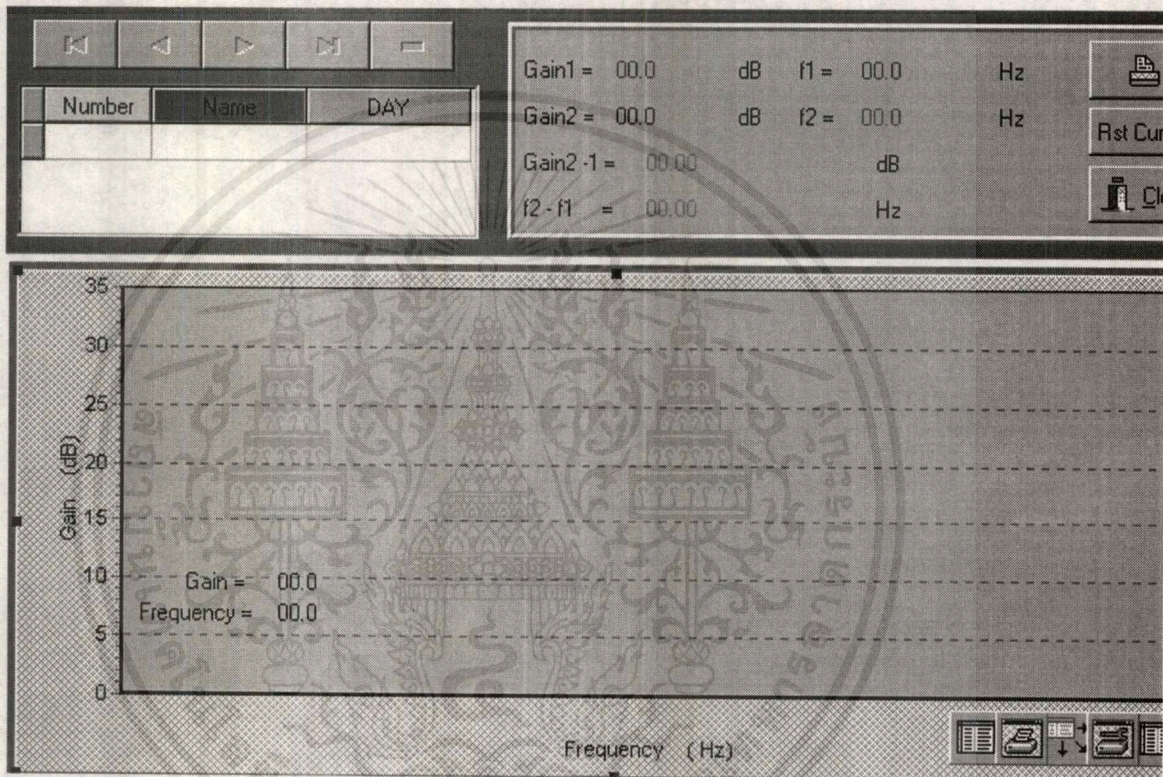
- ปุ่ม Run เริ่มทำการวัดและแสดงข้อมูล

- ส่วนตารางแสดงข้อมูล จะแสดงค่าความถี่, แรงดันอินพุต, แรงดันเอาต์พุต และเกณฑ์ของระบบ

- ส่วนแสดงความถี่ในขณะที่วัด
- ส่วนแสดงเกณฑ์ของระบบ ณ ความถี่ที่วัด
- ปุ่มแสดงตาราง ใช้สำหรับเพื่อต้องการดูผลการวัดในรูปของตารางข้อมูล
- ปุ่มแสดงกราฟ ใช้สำหรับเมื่อต้องการดูผลการทดลองในรูปของกราฟ
- ปุ่ม Close สำหรับปิดฟอร์ม

ในขณะที่ Run เราจะเห็นข้อมูลในตารางเพิ่มขึ้นและเราสามารถเลือกดูข้อมูลในรูปของกราฟได้ด้วย เมื่อทำการวัดจนถึงค่าความถี่สุดท้ายแล้ว โปรแกรมจะทำการบันทึกข้อมูลลงในฐานข้อมูลชื่อ GPIB DATA และจะแสดงฟอร์มที่ใช้เก็บข้อมูลขึ้นมาพร้อมทั้งกราฟและชื่อการทดลอง

### 4.3 ส่วนเก็บบันทึกและแสดงข้อมูล



รูปที่ 4.4 แสดงส่วนเก็บบันทึกและแสดงข้อมูล

เมื่อทำการ Run ควรทดลองจนหาค่าความถี่สุดท้ายแล้ว หากผู้ทำการทดลองต้องการที่จะบันทึกผลการทดลองนั้นไว้ ข้อมูลทั้งหมดก็จะถูกบันทึกลงในฐานข้อมูลในรูปแบบของไฟล์\*.DB พร้อมทั้งจะปรากฏฟอร์มนี้ขึ้นเพื่อแสดงกราฟของข้อมูลที่วัดได้ พร้อมทั้งชื่อการทดลอง,ลำดับการเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ทดลอง และวันที่ทำการทดลองในตาราง อีกทั้งยังสามารถดูข้อมูลเก่าที่เคยทำการทดลองไว้ก่อนได้ รวมทั้งค่าความถี่ คัทออฟ ของระบบ ซึ่งมีรายละเอียดดังนี้

- 1 ใช้สำหรับเลื่อนดูข้อมูลในตาราง พร้อมทั้งกราฟของข้อมูลด้วย
- 2 ตารางบันทึกชื่อการทดลองและลำดับการทดลอง
- 3 สั่งพิมพ์กราฟออกทางเครื่องพิมพ์
- 4 ปิดโปรแกรม
- 5 แสดงตำแหน่งบนแกน X ของเคอร์เซอร์
- 6 แสดงตำแหน่งบนแกน Y ของเคอร์เซอร์
- 7 แสดงค่าความถี่คัทออฟ ของระบบ

#### 4.4 ดาต้าเบสใน Delphi

เนื่องจากการเก็บข้อมูลเป็นเรื่องสำคัญ ซึ่งในยุคนี้สามารถเก็บได้ทั้ง ภาพ เสียง และวีดีโอ โดย Delphi จะอาศัย Borland Data Engine ( BDE ) เป็นเครื่องมือในการใช้งาน ดาต้าเบส ด้วย BDE

##### คำศัพท์เกี่ยวกับ ดาต้าเบส

- ฟิลด์ คือส่วนประกอบที่เล็กที่สุดใน ดาต้าเบส ฟิลด์จะมีชื่อเรียกและชนิด
- เรคอร์ด คือกลุ่มของฟิลด์ที่มีข้อมูลสัมพันธ์กัน
- ตาราง คือกลุ่มของเรคอร์ดที่มีโครงสร้างฟิลด์เหมือนกัน
- ดาต้าเบส คือกลุ่มของตารางที่สัมพันธ์กัน โดยปกติมันจะกำหนดโดยไคลเอนท์ที่เก็บ

ตารางที่โยงไปยังดาต้าเบส

- อินเด็กซ์หลัก คือจะเป็นตัวควบคุมลำดับในการแสดงเรคอร์ดของดาต้าเบส โดยควรจะมีตัวเดียว นั่นคือภายในฟิลด์จะมีข้อมูลซ้ำกันไม่ได้
- อินเด็กซ์รอง คืออินเด็กซ์อีกประเภทหนึ่งที่ขึ้นอยู่กับการผสมผสานของฟิลด์ ซึ่งข้อมูลภายในฟิลด์สามารถซ้ำกันได้ อินเด็กซ์ประเภทนี้จะใช้ในตารางที่มีการลิงค์
- คิวรี คือคำสั่งใน SQL ( Structured Query Language ) ซึ่งใช้สำหรับเลือกเรคอร์ดและฟิลด์จากตารางตั้งแต่หนึ่งตัวขึ้นไป

##### การดึงข้อมูลจากแหล่งข้อมูล

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตารางเป็นส่วนประกอบที่เรียบง่าย สำหรับส่วนประกอบแหล่งข้อมูลจะทำหน้าที่เป็นตัวกลางในการแลกเปลี่ยนข้อมูลระหว่างตารางและส่วนประกอบข้อมูล พร็อพเพอร์ตี้ DataSet ของมันจะชี้ไปยังตารางซึ่งเป็นที่ที่มีการไหลของข้อมูล ถ้า AutoEdit มีค่าเป็น True การไหลของข้อมูลจะเกิดขึ้นทั้งสองด้าน นั่นคือการแลกเปลี่ยนข้อมูลในส่วนประกอบที่เชื่อมต่อกับแหล่งข้อมูลนี้ ๑ ให้ข้อมูลในตารางเปลี่ยนแปลงโดยอัตโนมัติ แหล่งข้อมูลจะได้รับแจ้งเมื่อข้อมูลในตารางหรือส่วนประกอบมีการเปลี่ยนแปลง และเมื่อสถานะของตัวมันเองเปลี่ยน

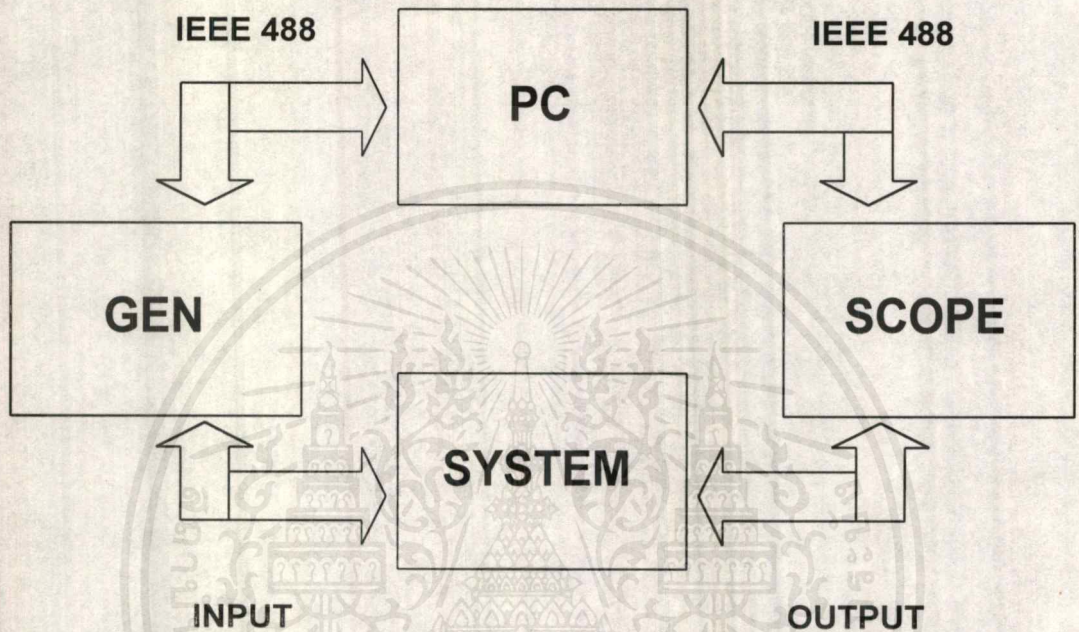
### การแสดงผลข้อมูลอย่างรวดเร็วด้วยกริดข้อมูล

การสร้างตารางข้อมูลในมุมมองแบบกริดเป็นเรื่องง่ายๆ เริ่มจากสร้างกริดบนฟอร์ม กำหนดพร็อพเพอร์ตี้ DataSource ให้เป็นแหล่งข้อมูลที่เชื่อมกับตาราง ถ้าทำให้ตาราง Active ในขณะออกแบบจะทำให้เห็นข้อมูลในกริดทันที กริดข้อมูลนี้จะใช้พร็อพเพอร์ตี้หลายๆ ตัวร่วมกับกริดสตริง และกริดการวาดนั่นคือพร็อพเพอร์ตี้พิเศษเกี่ยวกับกริดข้อมูล

### การสร้างเอดิเตอร์ฐานข้อมูล

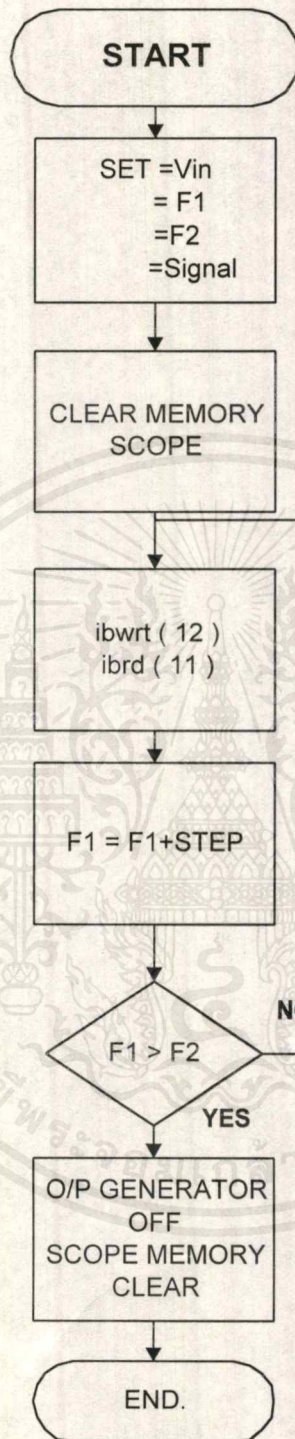
ในการสร้างส่วนประกอบตารางและแหล่งข้อมูลและไดอะล็อก Open File โดยกำหนดพร็อพเพอร์ตี้ DataSet ของแหล่งข้อมูลในตารางกำหนดพร็อพเพอร์ตี้ ReadOnly ของตารางให้เป็น True และกำหนดพร็อพเพอร์ตี้ AutoEdit ของแหล่งข้อมูลเป็น False ส่วนประกอบปีเวลบนฟอร์มกำหนดชนิดเป็น bsFrame และกำหนดพร็อพเพอร์ตี้ align เป็น alTop แล้วกำหนดพร็อพเพอร์ตี้ของไดอะล็อก Open File แล้วทำการดับเบิลคลิกพร็อพเพอร์ตี้ Filter สองตัวชื่อ DatabaseFiles โดยกำหนด File ตรงกันคือ \*.DB ; \*.DBF และ All Files จะกำหนดเป็น \*\*

#### 4.5 การติดตั้งอุปกรณ์และการทดลอง



รูปที่ 4.5 แสดงการติดตั้งอุปกรณ์และการทดลอง

จากการทดลองจะทำการต่อระบบดังรูปที่ 4.5 จากนั้นก็จะใช้ Application ที่เขียนขึ้นด้วย Delphi เป็นตัวสั่งงานทั้งหมดเริ่มตั้งแต่การกำหนดขนาดของสัญญาณอินพุตที่จะป้อนให้กับระบบ การกำหนดความถี่ รวมทั้งการเก็บบันทึกขนาดของสัญญาณเอาต์พุตที่ได้ โดยมีลำดับการทำงานดังนี้ ดังแสดงดังรูปที่ 4.6

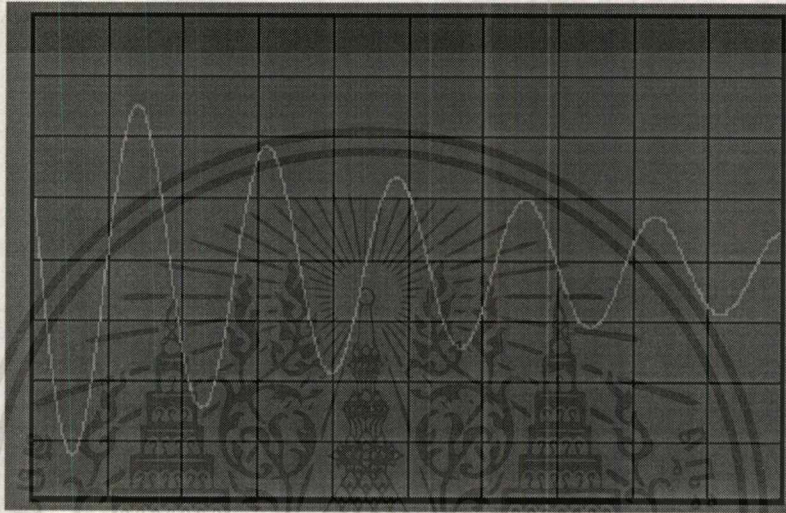


รูปที่ 4.6 แสดง Flow Chart การทำงานของการทดสอบ

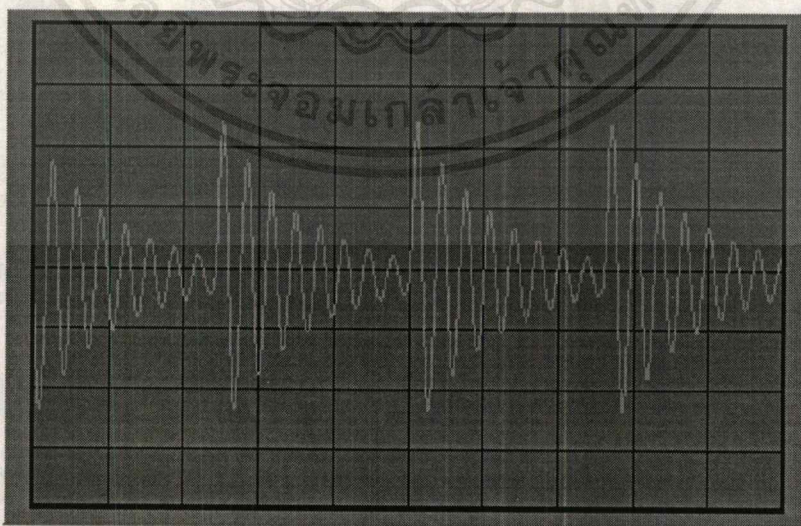
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

## 4.6 ผลการทดลอง

### 4.6.1 การนำสัญญาณต่างๆจากสโคปมาแสดงบนจอคอมพิวเตอร์

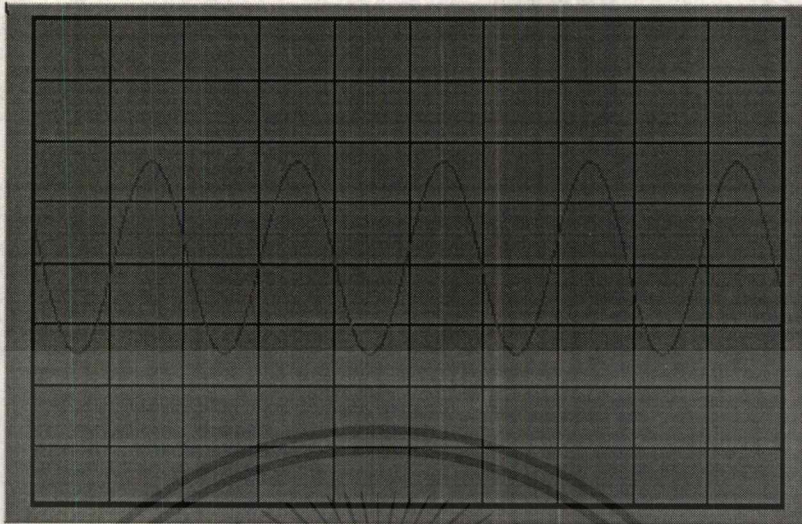


รูปที่ 4.7 แสดงการวัดสัญญาณ Scope Sweep

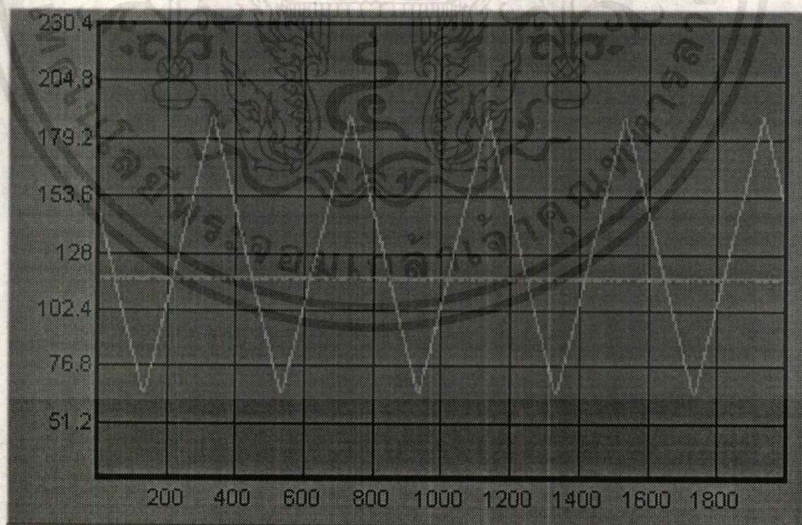


รูปที่ 4.8 แสดงการวัดสัญญาณ ARB

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

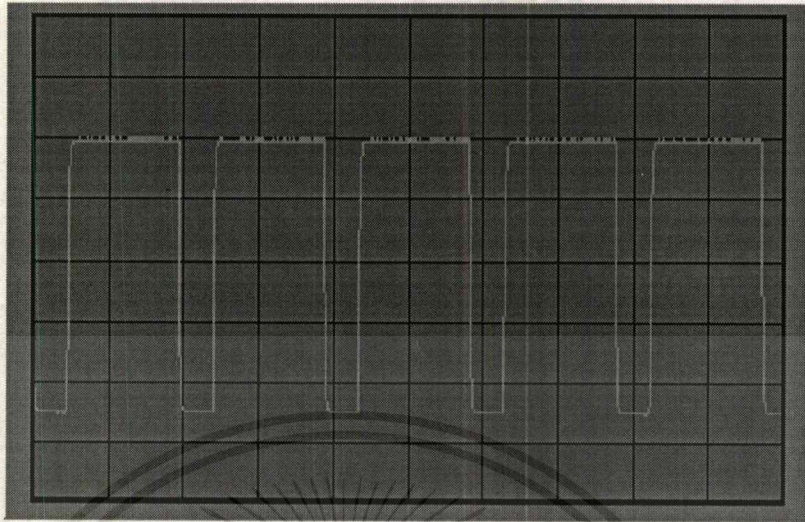


รูปที่ 4.9 แสดงการนำสัญญาณ SIN ที่ปรากฏบนจอสโคปมาแสดงบนมโนเตอร์



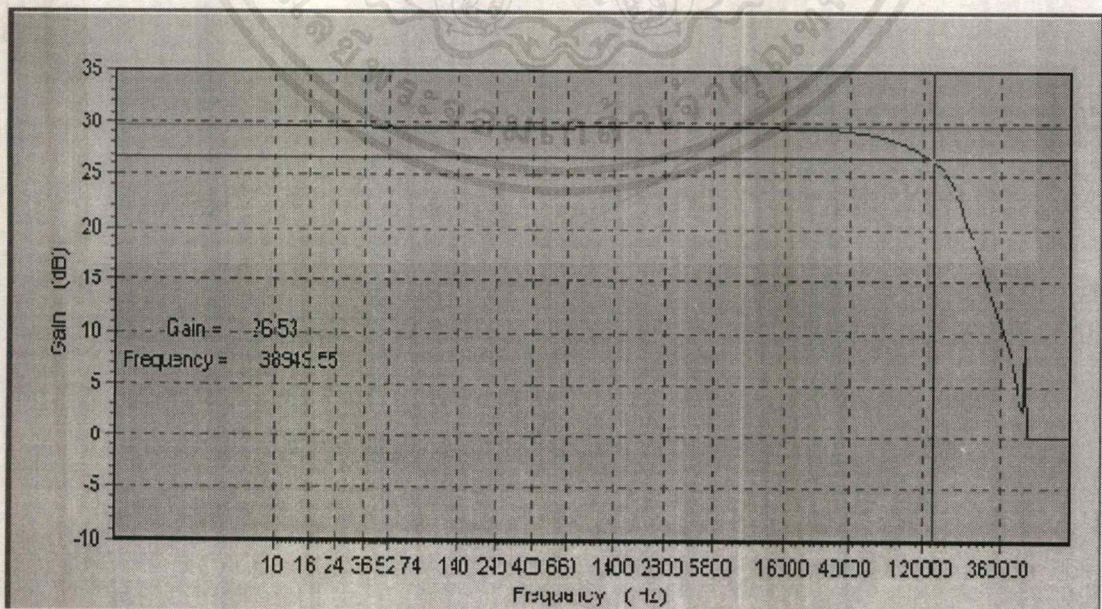
รูปที่ 4.10 แสดงการนำสัญญาณ TRI จากสโคปมาแสดงบนจอมอนิเตอร์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 4.11 แสดงการนำสัญญาณ PLUSE จากสโคปมาแสดงบนจอมอนิเตอร์

#### 4.6.2 การแสดงผลการวัดหา Frequency Response LOW PASS FILTER

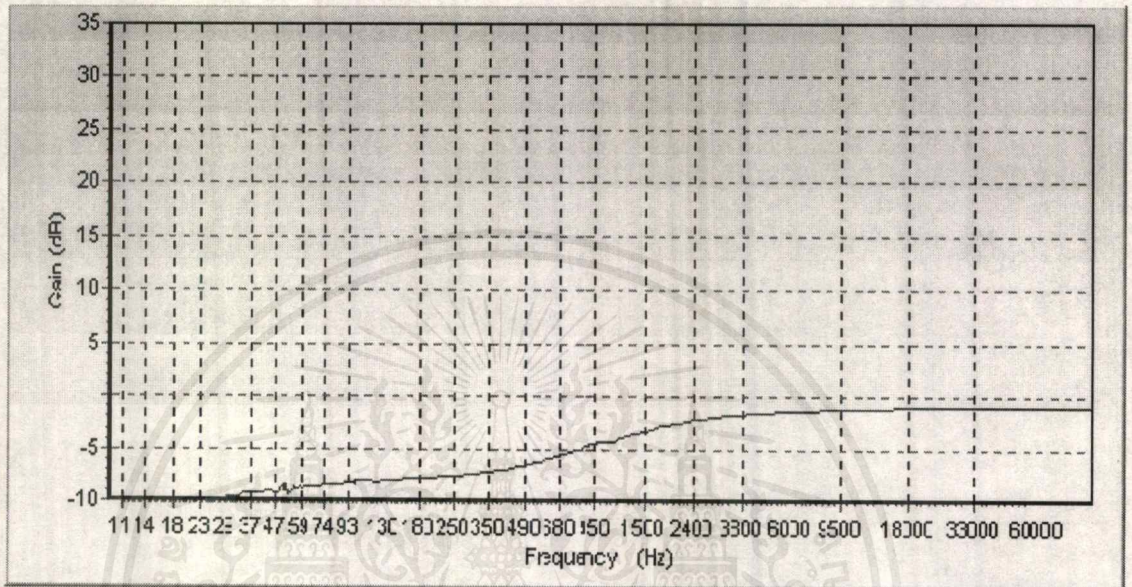


รูปที่ 4.12 แสดงการวัดหา Frequency Response ของ LOW PASS FILTER

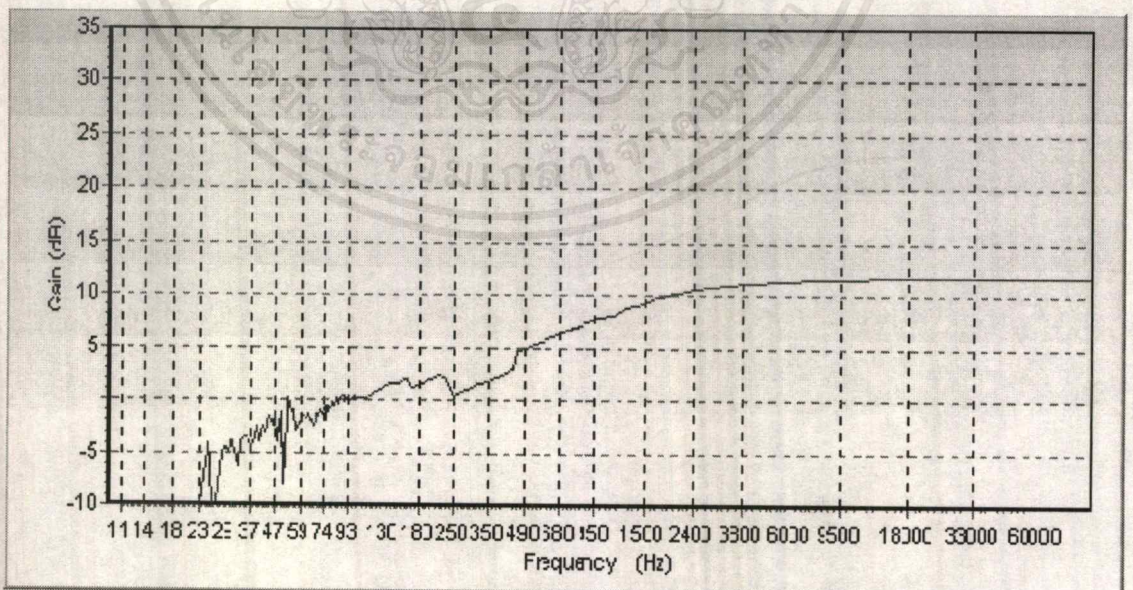
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น เมื่ออนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

#### 4.6.3 การวัดหา Frequency Response ของ HIGHT PASS FILTER

$V_{in} = 2 \text{ Vp-p}$      $FT = 5000 \text{ Hz}$



$V_{in} = 4 \text{ Vp-p}$      $FT = 1200 \text{ Hz}$



รูปที่ 4.13 แสดงการวัดหา Frequency Response ของ HIGHT PASS FILTER

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

## บทที่ 5

### สรุปและวิจารณ์

การจัดทำโครงการนี้เป็นการศึกษาทดลองใช้โปรแกรมที่เขียนขึ้นด้วย Delphi ติดต่อกับการ์ด GPIB-PCII/ IIA เพื่อใช้ควบคุมอุปกรณ์ GPIBต่างๆ ซึ่งในโครงการนี้ได้เขียนโปรแกรมควบคุมติดต่อกับฟังก์ชันเจนเนอเรเตอร์ ,Power Supply,Multimeter และ Scope ซึ่งในเนื้อหาโครงการสามารถแบ่งออกเป็น 2 ส่วนคือ

- 1 ส่วนที่แสดงหน้าปัดของอุปกรณ์แต่ละตัวซึ่งสามารถควบคุมอุปกรณ์ได้
- 2 ส่วนการประยุกต์ใช้งาน

#### 5.1 ปัญหาที่พบในส่วนแสดงหน้าปัดของอุปกรณ์

ในการเขียน โปรแกรมของส่วนนี้ ก็เพื่อที่จะใช้ควบคุมและตั้งค่าต่างๆ ให้กับอุปกรณ์ โดยผ่านตัวโปรแกรม จากการทดลองพบว่าสามารถที่จะใช้คำสั่งควบคุมอุปกรณ์ได้ดี ตามรูปแบบและฟังก์ชันของอุปกรณ์ที่มีในคู่มือ ซึ่งจะขอแยกอธิบายทีละตัว

**Function Generator HM 8130** เป็นอุปกรณ์ที่มีชุดคำสั่งควบคุมอยู่ทุกฟังก์ชัน จึงทำให้สามารถควบคุมได้อย่างสมบูรณ์ที่สุด

**Power Supply HM 8142** ในการทดลองเขียนคำสั่งควบคุมจะพบปัญหาความไม่มีเสถียรภาพของ Power Supply แต่ละเครื่อง จึงทำให้ต้องมีการปรับปรุงตัวโปรแกรมในเรื่องวิธีการส่ง ให้สามารถควบคุมได้ทุกเครื่อง อีกทั้งปัญหาที่พบบ่อยมากคือ ค่าแรงดันที่จ่ายออกมาของ Power Supply ในขณะที่เข้าที่พุท OFF กับเมื่อเข้าที่พุท ON จะมีค่าไม่เท่ากัน ซึ่งเป็นความผิดพลาดที่เกิดขึ้นจากตัวอุปกรณ์ ทางคณะผู้จัดทำได้แก้ไขปัญหานี้โดยให้ตัวโปรแกรมควบคุมทำการตรวจสอบค่าแรงดันของ Power Supply เมื่อมีการเปลี่ยนแปลงค่า

**Multimeter HM 8122-2** เป็นอุปกรณ์ที่มีฟังก์ชันการวัดอยู่หลายค่า แต่ทางคณะผู้เขียนได้เลือกเขียนควบคุมเฉพาะฟังก์ชันการวัดที่ต้องใช้เป็นประจำโดยทำการตัดส่วนของฟังก์ชันต่าง ๆ ที่ไม่ค่อยได้ใช้ออกไป เช่น การวัดอุณหภูมิและการวัดค่าความต้านทานแบบ 4 สาย ในการทดลองเขียนโปรแกรมควบคุมจะพบว่า Multimeter เป็นอุปกรณ์ที่มีค่าความเชื่อถือในการส่งและรับคำสั่งจากตัวโปรแกรมน้อยที่สุด ทั้งนี้เนื่องมาจากเป็นอุปกรณ์ที่มีการทำงานที่ช้า ดังนั้นในตัวโปรแกรมระหว่างคำสั่งส่งค่าและรับค่าจากMultimeter จะต้องมีการหน่วงเวลาซึ่งได้ทำการทดลอง และกำหนดค่าที่เหมาะสมที่ทำให้การส่งและรับสามารถกระทำได้อย่างสมบูรณ์แล้ว อีกทั้งตัวโปรแกรมจะต้อง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ทำการอ่านข้อมูลจากมิเตอร์ที่วัด เพื่อเป็นการปรับให้ค่าที่แสดงบนหน้าจอ และค่าที่มิเตอร์วัดได้เป็นค่าเดียวกันหรือ ใกล้เคียงกันมากที่สุดและด้วยค่าเวลาที่น้อยที่สุดที่ Multimeter สามารถ ที่จะทำงานได้

Oscilloscope HM 1007 เป็นอุปกรณ์ที่มีฟังก์ชันการทำงานน้อยที่สุดในการเขียนโปรแกรมควบคุม สามารถทำได้เพียงแค่ทำการอ่านข้อมูล ที่เป็นรูปสัญญาณที่เก็บไว้มาแสดงบนหน้าจอเท่านั้น และไม่สามารถทำการอ่านค่าแบบต่อเนื่องได้ เนื่องจากฟังก์ชันของ V/DIV และ Time/DIV ไม่สามารถควบคุมโดยผ่านตัวโปรแกรมได้

## 5.2 ปัญหาที่พบในส่วนของการประยุกต์ใช้งาน

เป็นส่วนที่ใช้เก็บบันทึกข้อมูลทั้งในรูปแบบของตารางและกราฟ ซึ่งจะแสดงให้ผู้ทำการทดลองได้ค่าทั้งในขณะที่ทำการทดลองอยู่ และสามารถทำการเก็บบันทึกผลการทดลองนั้นไว้ ปัญหาที่พบในส่วนนี้คือ

- ระยะเวลาที่ในการทดลองแต่ละครั้งซึ่งจะขึ้นอยู่กับค่าของการเพิ่มของความถี่ในแต่ละครั้ง ค่าของความถี่ที่เพิ่มขึ้นในแต่ละครั้งจะมีผลต่อความละเอียดของข้อมูลที่ได้จากการทดลอง ดังนั้นการตั้งค่าจึงขึ้นอยู่กับความต้องการของผู้ทดลอง

- ความเร็วในการอ่านข้อมูล ปัญหาในส่วนนี้ เกิดขึ้นจากความเร็วในการรับข้อมูลและส่งข้อมูลของอุปกรณ์แต่ละตัว ซึ่งตัว Multimeter จะเป็นอุปกรณ์ตัวที่ทำงานช้ามากที่สุด ทำให้โปรแกรมที่เขียนขึ้นสามารถอ่านและเขียนข้อมูลได้ที่ 0.6 นาที ต่อครั้ง ซึ่งเป็นค่าเวลาที่น้อยที่สุดที่ทำให้การทำงานของโปรแกรมไม่เกิดข้อผิดพลาด และตัวอุปกรณ์สามารถเปลี่ยนแปลงค่าได้ทันที

## 5.3 แนวทางการพัฒนาและแก้ไข

ในการจัดทำโครงงานในส่วนของการใช้โปรแกรมที่เขียนขึ้นด้วย Delphi มาทำการควบคุมอุปกรณ์ GPIB นี้คณะผู้จัดทำได้มีเวลาในการทำเพียงเทอมเดียว คือ ในเทอมที่ 2 ของปีการศึกษา ซึ่งในเทอมแรกคณะผู้จัดทำได้ทำการศึกษาเกี่ยวกับการทำงานและการประยุกต์ใช้งานของโปรแกรม LabView ซึ่งเป็นตัวโปรแกรมที่มีส่วนของ Source Code เป็นกราฟฟิกและในเทอมที่ 2 จึงได้มีแนวความคิด ที่จะพัฒนาตัวโปรแกรมที่ทำงานแบบกราฟฟิกขึ้นมาเพื่อใช้เปรียบเทียบการทำงานกับโปรแกรมที่เขียนขึ้นโดยใช้ Labview

ดังนั้นจึงยังมีส่วนอื่นๆ ที่ผู้สนใจสามารถเข้ามาพัฒนาทำต่อไปได้อีกอย่างเช่น การปรับปรุงความเร็วในการอ่านข้อมูลให้สามารถทำได้เร็วขึ้น การปรับปรุงรูปแบบในส่วนแสดงผลให้สวยงามมากขึ้นการเพิ่มฟังก์ชันต่างๆ ในตัวโปรแกรมสามารถทำงานได้กว้างมากขึ้น หรือการทำให้

สามารถทำการทดลองได้หลายๆการทดลองในเวลาเดียวกัน ทั้งนี้ก็ขึ้นอยู่กับความสามารถและเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่นิยมนำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ความชำนาญในการเขียนโปรแกรม Delphi ซึ่งยังมีความสามารถในส่วนอื่นๆ อีกมากที่ทางคณะผู้  
 ทำเห็นว่าจะเป็นส่วนที่สามารถนำมาใช้พัฒนาโครงการนี้ต่อไปได้



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
 ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

## หนังสืออ้างอิง

- 1 แรกเริ่มเรียนรู้เรื่องการเขียนโปรแกรมด้วย Delphi โดย Neil J. Rubenking  
เรียบเรียงโดย ทรงศักดิ์ บรรจงมณี
- 2 Delphi 3 SUPERBIBLE Paul Thurrott, Gary Brent, Richard Bagdazian, Steve Tendon
- 3 เรียนรู้เทอร์โบปาสคาล สุชาดา รัตนคงเนตร
- 4 Delphi บุญเลิศ เอี่ยมทัศนาศ
- 5 National Instrument Getting Started with Your GPIB-PCII/IIA and the GPIB Software  
for Windows95
- 6 National Instrument NI-488.2M Function Reference Manual For Win32
- 7 National Instrument NI-488.2m Manual for Windows95 and Windows NT

**unit** MainForm;

**interface**

**uses**

Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,  
StdCtrls, Buttons, TeEngine, Series, TeeProcs, Chart, ExtCtrls, TeeFunci;

**Type**

TMainDev = class(TForm)

Panel1: TPanel;

SpeedButton1: TSpeedButton;

SpeedButton2: TSpeedButton;

SpeedButton3: TSpeedButton;

SpeedButton4: TSpeedButton;

SpeedButton5: TSpeedButton;

Panel2: TPanel;

Panel3: TPanel;

Timer1: TTimer;

Chart1: TChart;

Series1: TPieSeries;

Chart2: TChart;

Series2: TPieSeries;

Label1: TLabel;

Label2: TLabel;

SpeedButton6: TSpeedButton;

SpeedButton7: TSpeedButton;

procedure SpeedButton2Click(Sender: TObject);

procedure SpeedButton1Click(Sender: TObject);

procedure SpeedButton5Click(Sender: TObject);

procedure SpeedButton4Click(Sender: TObject);

procedure Label2Click(Sender: TObject);

procedure Timer1Timer(Sender: TObject);

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
procedure FormCreate(Sender: TObject);  
procedure SpeedButton3Click(Sender: TObject);  
procedure FormActivate(Sender: TObject);  
procedure FormDeactivate(Sender: TObject);  
procedure SpeedButton6Click(Sender: TObject);  
procedure SpeedButton7Click(Sender: TObject);
```

**private**

```
{ Private declarations }
```

**public**

```
{ Public declarations }
```

**end;**

**Var**

```
MainDev: TMainDev;
```

**implementation**

```
uses DataTest1, ProgramSupplyButt, Multimeter1, GENERATOR, DataTest2, scope;
```

```
{ $R *.DFM }
```

```
procedure TMainDev.SpeedButton2Click(Sender: TObject);
```

```
begin
```

```
    supply.Show;
```

```
end;
```

```
procedure TMainDev.SpeedButton1Click(Sender: TObject);
```

```
begin
```

```
    METER.Show;
```

```
end;
```

```
procedure TMainDev.SpeedButton5Click(Sender: TObject);
```

```
begin
```

```
    close;
```

```
end;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
procedure TMainDev.SpeedButton4Click(Sender: TObject);
```

```
begin
```

```
    Gen.show;
```

```
end;
```

```
procedure TMainDev.Label2Click(Sender: TObject);
```

```
begin
```

```
    SpeedButton1.Click;
```

```
end;
```

```
procedure TMainDev.Timer1Timer(Sender: TObject);
```

```
begin
```

```
    Series1.Rotate(5);
```

```
    Series2.Rotate(355);
```

```
end;
```

```
procedure TMainDev.FormCreate(Sender: TObject);
```

```
begin
```

```
    Series1.FillSampleValues(5);
```

```
    Series2.FillSampleValues(5);
```

```
end;
```

```
procedure TMainDev.SpeedButton3Click(Sender: TObject);
```

```
begin
```

```
    DataRun.Show;
```

```
end;
```

```
procedure TMainDev.FormActivate(Sender: TObject);
```

```
begin
```

```
    Timer1.Enabled := True;
```

```
end;
```

```
procedure TMainDev.FormDeactivate(Sender: TObject);
```

```
begin
```

```
    Timer1.Enabled := False;
```

```
end;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
procedure TMainDev.SpeedButton6Click(Sender: TObject);  
  
begin  
    Data2.Show;  
  
end;  
  
procedure TMainDev.SpeedButton7Click(Sender: TObject);  
  
begin  
    ScopeGpib.Show;  
  
end;  
  
end.
```



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
unit DataTest1;
{
interface
```

**uses**

Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,  
StdCtrls, Db, DBTables, Grids, DBGrids, DBCtrls, ExtCtrls, TeEngine,  
Series, TeeProcs, Chart, DBChart, Buttons, TeeFunci,Math;

**type**

```
TDataRun = class(TForm)
```

```
DBGrid1: TDBGrid;
```

```
DataSource1: TDataSource;
```

```
Label1: TLabel;
```

```
Label2: TLabel;
```

```
Timer1: TTimer;
```

```
Panel1: TPanel;
```

```
Label3: TLabel;
```

```
Label4: TLabel;
```

```
Label5: TLabel;
```

```
Label6: TLabel;
```

```
Edit1: TEdit;
```

```
Label7: TLabel;
```

```
Panel2: TPanel;
```

```
Edit2: TEdit;
```

```
Edit3: TEdit;
```

```
Edit4: TEdit;
```

```
Label8: TLabel;
```

```
Label9: TLabel;
```

```
Label10: TLabel;
```

```
Label11: TLabel;
```

```
Label12: TLabel;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

BitBtn1: TBitBtn;  
DBChart1: TDBChart;  
Label14: TLabel;  
Label16: TLabel;  
Label17: TLabel;  
Label18: TLabel;  
Panel4: TPanel;  
Shape1: TShape;  
Label19: TLabel;  
Shape2: TShape;  
Label20: TLabel;  
Label21: TLabel;  
Label22: TLabel;  
Table1: TTable;  
SpeedButton2: TSpeedButton;  
SpeedButton1: TSpeedButton;  
Series1: TFastLineSeries;  
TeeFunction1: TAddTeeFunction;  
Label13: TLabel;  
BitBtn2: TBitBtn;  
Table1Frequency: TFloatField;  
Table1Vin: TFloatField;  
Table1Vout: TFloatField;  
Table1Gain: TFloatField;  
BitBtn3: TBitBtn;  
Label15: TLabel;  
Edit5: TEdit;  
Label23: TLabel;  
Label24: TLabel;  
procedure Button1Click(Sender: TObject);  
procedure Timer1Timer(Sender: TObject);  
procedure BitBtn1Click(Sender: TObject);

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
procedure FormShow(Sender: TObject);  
procedure SpeedButton2Click(Sender: TObject);  
procedure SpeedButton1Click(Sender: TObject);  
procedure BitBtn3Click(Sender: TObject);
```

```
private  
    { Private declarations }
```

```
public  
  
    { Public declarations }
```

```
end;
```

```
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 *)  
ATN    = $10;    (* Attention asserted *)  
TACS    = $8;    (* Talker active *)  
LACS    = $4;    (* Listener active *)  
DTAS    = $2;    (* Device trigger state *)  
DCAS    = $1;    (* Device clear state *)
```

(\* 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. \*)

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

AddrList = array [0..31] of SmallInt;

Tibsta = function : integer; stdcall;

Tiberr = function : integer; stdcall;

Tibcntl = function : Longint; stdcall;

Tibclr = function(ud : integer): integer; stdcall;

Tibdev = function(ud : integer;

pad : integer;

sad : integer;

tmo : integer;

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
eot : integer;  
eos : integer): integer;stdcall;
```

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

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

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

```
TFindLstn = procedure (boardID: integer;  
    padlist: AddrList;  
    resultlist: AddrList;  
    limit: integer); stdcall;
```

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

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

```
TSend = procedure (boardID: integer;  
    address: word;  
    var buffer;  
    datacnt: longint;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

eotmode: integer); stdcall;

TSendIFC = procedure (boardID: integer); stdcall;

**var**

DataRun: TDataRun;

NewTable: TTable;

NewIndexOptions: TIndexOptions;

TableFound: Boolean;

Freq : real;

buf : packed array[0..100] of char;

AddrIbsta : Tibsta;

AddrIberr : Tiberr;

AddrIbcntl : Tibcntl;

Pibsta : ^integer;

Piberr : ^integer;

Pibcntl : ^Longint;

Gpib32Lib: THandle;

ibclr : Tibclr;

ibdev : Tibdev;

ibrd : Tibrd;

ibwrt : Tibwrt;

DevClear: TDevClear;

FindLstn: TFindLstn;

Receive: TReceive;

Send: TSend;

SendIFC: TSendIFC;

ibonl: Tibonl;

dvm : integer; { \* device Number \* }

msg : String;

Vo,Vi : Variant;

f1 : real;

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
implementation
```

```
uses DataTest2,generator,Multimeter1;
```

```
{ $R *.DFM }
```

```
procedure loadDLL;
```

```
var
```

```
str : string;
```

```
begin
```

```
Gpib32Lib := LoadLibrary('GPIB-32.DLL');
```

```
If Gpib32Lib = 0 Then
```

```
Begin
```

```
str := 'LoadLibrary FAILED!';
```

```
MessageDlg(str, mtError, [mbOK], 0);
```

```
halt;
```

```
End;
```

```
@AddrIbsta := GetProcAddress(Gpib32Lib, 'user_ibsta');
```

```
@AddrIberr := GetProcAddress(Gpib32Lib, 'user_iberr');
```

```
@AddrIbcntl := GetProcAddress(Gpib32Lib, 'user_ibcntl');
```

```
@ibclr := GetProcAddress(Gpib32Lib, 'ibclr');
```

```
@ibdev := GetProcAddress(Gpib32Lib, 'ibdev');
```

```
@ibrd := GetProcAddress(Gpib32Lib, 'ibrd');
```

```
@ibwrt := GetProcAddress(Gpib32Lib, 'ibwrt');
```

```
@DevClear := GetProcAddress(Gpib32Lib, 'DevClear');
```

```
@FindLstn := GetProcAddress(Gpib32Lib, 'FindLstn');
```

```
@ibonl := GetProcAddress(Gpib32Lib, 'ibonl');
```

```
@Receive := GetProcAddress(Gpib32Lib, 'Receive');
```

```
@Send := GetProcAddress(Gpib32Lib, 'Send');
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

@SendIFC      := GetProcAddress(Gpib32Lib, 'SendIFC');
if (@ibdev    = NIL) or
   (@ibclr    = NIL) or
   (@ibrd     = NIL) or
   (@ibwrt    = NIL) or
   (@AddrIbctl = NIL) Or
   (@DevClear  = NIL) Or
   (@FindLstn  = NIL) Or
   (@ibonl    = NIL) Or
   (@Receive   = NIL) Or
   (@Send      = NIL) Or
   (@SendIFC   = NIL) Then
Begin
    str := 'GetProcAddresses FAILED!';
    MessageDlg(str, mtError, [mbOK], 0);
    FreeLibrary(Gpib32Lib);
    halt;
End;

Pibsta := @AddrIbsta;
Piberr := @AddrIberr;
Pibcntl := @AddrIbcntl;

end;

```

```

procedure TDataRun.Button1Click(Sender: TObject);

```

```

begin

```

```

    NewTable := TTable.Create(Self);

```

```

    NewIndexOptions := [ixPrimary, ixUnique];

```

```

    with NewTable do

```

```

        begin

```

```

            Active := False;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

DatabaseName := 'GPIBData';
TableName := Edit1.Text;
TableType := ttDefault;
FieldDefs.Clear;
FieldDefs.Add('Frequency', ftInteger, 0, False);
FieldDefs.Add('Vin', ftInteger, 0, False);
FieldDefs.Add('Vout', ftInteger, 0, False);
IndexDefs.Clear;
IndexDefs.Add('PrimaryIndex', 'Frequency', NewIndexOptions);
end;
TableFound := False;
if TableFound = True then
if MessageDlg('Overwrite existing table ' + Edit1.Text + '?', mtConfirmation,
    mbYesNoCancel, 0) = mrYes then
    TableFound := False;
if not TableFound then
    NewTable.CreateTable;
    NewTable.Active := True;
    DataSource1.DataSet := NewTable;

end;

```

```

procedure TDataRun.Timer1Timer(Sender: TObject);

```

```

    var f2 : real; Gain : real; msg : string[12];

```

```

    buf : array[0..100] of char; i,j,x,y:integer;

```

```

    Fstrat : string[15];

```

```

begin

```

```

    f2 := StrToFloat(Edit3.Text);

```

```

    Vi := StrToFloat(Edit4.Text);

```

```

    loadDll;

```

```

    SendIFC(0);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

Fstrat := floatToStr(f1);
buf := "";
buf := 'FRQ:~';
for j := 1 to Length(Fstrat) do
    buf[3 + j] := Fstrat[j];
buf[3 + j] := '~';
Send(0,12,buf,15,1);
dvm:= ibdev(0,7,0,13,0,1);
buf := 'VA:~';
ibwrt(dvm,buf,5);
for i := 0 to 2 do for y := 0 to 10 do X := 1;
    ibrd(dvm,buf,32);
msg := buf;
Vo := strtfloat(msg);
Vo := Vo*2*sqrt(2);
label19.caption := Fstrat;
if Vo <= Vi then Gain := 0
else Gain := 20*log10(Vo/Vi);
label20.caption := floattostr(Gain);
NewTable.AppendRecord([f1,Vi,Vo,Gain]);
NewTable.Refresh;
Table1 := NewTable;
Table1.Refresh;
series1.AddXY(f1,Gain,"clRed);
if f1 >= f2 then
    begin Timer1.Enabled := False;
        buf := "";
        buf := 'OT0:~';
        send(0,12,buf,10,1);
        if MessageDlg(Label13.Caption,mtconfirmation,mbYesNoCancel,0) = mrYes then
            begin
                Data2.Table1.Open;

```

```

Data2.Table1.AppendRecord([data2.table1.recordcount+1,Edit1.Text,Date]);
Data2.Show;
Data2.Table2.Close;
Data2.Table2.TableName := Edit1.Text;
NewTable.Active := False;
Data2.Table2.Open;
end
else begin

```

```

    NewTable.Active := FALSE;
    NewTable.DeleteTable;
    freq := 0;
end;
end;
if f1 < 100 then f1 := f1 + 2
else if f1 < 1000 then f1 := f1 + 20
    else if f1 < 10000 then f1 := f1 + 200
        else if f1 < 100000 then f1 := f1 + 2000
            else f1 := f1 + 20000;
end;

```

```

procedure TDataRun.BitBtn1Click(Sender: TObject);
var f2,v1 : real;
begin
    Series1.Clear;
    f1 := StrToFloat(Edit2.Text); f2 := StrToFloat(Edit3.Text) ;
    if f1 > f2 then
        begin
            ShowMessage(Label17.Caption);
            Edit2.SetFocus;
        end;
    exit;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

end;
v1 := StrToFloat(Edit4.Text);
if v1 > 10 then
begin
    ShowMessage(Label18.Caption);
    Edit4.SetFocus;
    Exit;
end;
NewTable := TTable.Create(Table1);
NewIndexOptions := [ixPrimary, ixUnique];
with NewTable do
begin
    Active := False;
    DatabaseName := 'GPIBData';
    TableName := Edit1.Text;
    TableType := ttDefault;
    FieldDefs.Clear;
    FieldDefs.Add('Frequency', ftFloat, 0, False);
    FieldDefs.Add('Vin', ftFloat, 0, False);
    FieldDefs.Add('Vout', ftFloat, 0, False);
    FieldDefs.Add('Gain', ftFloat, 0, False);
    IndexDefs.Clear;
    IndexDefs.Add('PrimaryIndex', 'Frequency', NewIndexOptions);
end;
TableFound := False;
if TableFound = True then
    if MessageDlg('Overwrite existing table ' + Edit1.Text + '?', mtConfirmation,
        mbYesNoCancel, 0) = mrYes then
        TableFound := False;
if not TableFound then
    try
        NewTable.CreateTable;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

NewTable.Active := True;

DataSource1.DataSet := NewTable;

except

    Table1.Active := False;

    timer1.Enabled := False;

    Table1.TableName := Edit1.Text;

    ShowMessage(Label16.caption);

    Edit1.SetFocus;

    exit;

end;

Panel4.Visible := True;

loadDll;

SendIFC(0);

buf := "";

buf := 'OT1;';

send(0,12,buf,10,1);

timer1.Enabled := True;

end;

procedure TDataRun.FormShow(Sender: TObject);

begin

    Label6.Caption := DateToStr(Now);

    label4.caption := inttostr(data2.table1.recordcount+1);

    Edit2.Enabled := False;

    Edit4.Enabled := False;

    Edit5.Enabled := False;

end;

procedure TDataRun.SpeedButton2Click(Sender: TObject);

begin

    DbGrid1.Visible := True;

    DBChart1.Visible := False;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
SpeedButton2.Enabled := False;

SpeedButton1.Enabled := True;

end;

procedure TDataRun.SpeedButton1Click(Sender: TObject);
begin
    DbGrid1.Visible := False;
    DBChart1.Visible := True;
    SpeedButton2.Enabled := True;
    SpeedButton1.Enabled := False;

end;

procedure TDataRun.BitBtn3Click(Sender: TObject);
begin
    Gen.Show;
    Meter.Show;
end;

end.
```



```
unit DataTest2;
```

```
interface
```

```
uses
```

```
Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,  
ExtCtrls, TeeProcs, TeEngine, Chart, DBChart, Grids, DBGrids, StdCtrls,  
Buttons, Series, DBCtrls, Db, DBTables;
```

```
type
```

```
TData2 = class(TForm)
```

```
Table1: TTable;
```

```
DataSource1: TDataSource;
```

```
Panel1: TPanel;
```

```
DBGrid1: TDBGrid;
```

```
DBNavigator1: TDBNavigator;
```

```
Table2: TTable;
```

```
Panel2: TPanel;
```

```
SpeedButton1: TSpeedButton;
```

```
BitBtn3: TBitBtn;
```

```
Panel3: TPanel;
```

```
DBChart1: TDBChart;
```

```
Series1: TFastLineSeries;
```

```
Table1Name: TStringField;
```

```
Table1Number: TFloatField;
```

```
Table1DAY: TDateField;
```

```
PrinterSetupDialog1: TPrinterSetupDialog;
```

```
PrintDialog1: TPrintDialog;
```

```
Label11: TLabel;
```

```
Label12: TLabel;
```

```
Label13: TLabel;
```

```
Label14: TLabel;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

Label1: TLabel;
Label2: TLabel;
Label3: TLabel;
Label4: TLabel;
Label5: TLabel;
Label6: TLabel;
Label7: TLabel;
Label8: TLabel;
Label9: TLabel;
Label10: TLabel;
Button1: TButton;
Label15: TLabel;
Label16: TLabel;
Label17: TLabel;
Label18: TLabel;
Label19: TLabel;
Label20: TLabel;
Label21: TLabel;
Label22: TLabel;
procedure DBNavigator1Click(Sender: TObject; Button: TNavigateBtn);
procedure SpeedButton1Click(Sender: TObject);
procedure DBChart1MouseMove(Sender: TObject; Shift: TShiftState; X,
Y: Integer);
procedure FormCreate(Sender: TObject);
procedure DBChart1MouseDown(Sender: TObject; Button: TMouseButton;
Shift: TShiftState; X, Y: Integer);
procedure Button1Click(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
OldX,OldY:Longint;

```

```

CrossHairColor:TColor;
CrossHairStyle:TPenStyle;

end;

var
  Data2: TData2;
  LineCnt : integer;
  Fr1,Fr2,Fr3 : real;
  G1,G2 : real;
implementation

{$R *.DFM}

procedure TData2.DBNavigator1Click(Sender: TObject; Button: TNavigateBtn);
begin
  Table1.Edit;
  Table2.Active := False;
  Table2.TableName := Table1.FieldValues['Name'];
  Table2.Active := True;
  DBChart1.AutoRefresh := True;
  Table2.Refresh;
  Table1.Post;
end;

procedure TData2.SpeedButton1Click(Sender: TObject);
begin
  PrintDialog1.Execute;
end;

procedure TData2.DBChart1MouseMove(Sender: TObject; Shift: TShiftState; X,
Y: Integer);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

{ This procedure draws the crosshair lines }

**Procedure** DrawCross(AX,AY:Integer);

**begin**

With DBChart1,Canvas do

**begin**

Pen.Color:=CrossHairColor;

Pen.Style:=CrossHairStyle;

Pen.Mode:=pmXor;

Pen.Width:=1;

MoveTo(ax,ChartRect.Top);

LineTo(ax,ChartRect.Bottom);

MoveTo(ChartRect.Left,ay);

LineTo(ChartRect.Right,ay);

**end;**

**end;**

**Var** tmpX,tmpY:Double;G3 : real;

**begin**

**if** LineCnt < 2 **then**

**begin**

**if** (OldX<>-1) **then**

**begin**

DrawCross(OldX,OldY); { draw old crosshair }

OldX:=-1;

**end;**

{ check if mouse is inside Chart rectangle }

**if** PtInRect( DBChart1.ChartRect, Point(X,Y) ) **then**

**begin**

DrawCross(x,y); { draw crosshair at current position }

{ store old position }

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

OldX:= x;
OldY:= y;
With Series1 do
begin
  GetCursorValues(tmpX,tmpY); { <-- get values under mouse cursor }
  Label11.Caption:=GetVertAxis.LabelValue(tmpY);
  Label12.Caption :=GetHorizAxis.LabelValue(tmpX);
end;
if LineCnt < 1 then
begin
  Label1.Caption := Label11.Caption;
  Label2.Caption := Label12.Caption;
  G1 := StrToFloat(Label1.Caption);
  G2 := StrToFloat(Label7.Caption);
  Fr1 := StrToCurr(Label2.Caption);
  Fr2 := StrToCurr(Label8.Caption);
  G3 := G2 - G1;
  Fr3 := Fr2 - Fr1;
  Label15.Caption := floatToStr(G3);
  Label16.Caption := currToStr(Fr3);
end
else
begin
  Label7.Caption := Label11.Caption;
  Label8.Caption := Label12.Caption;
  G1 := StrToFloat(Label1.Caption);
  G2 := StrToFloat(Label7.Caption);
  Fr1 := StrToCurr(Label2.Caption);
  Fr2 := StrToCurr(Label8.Caption);
  G3 := G2 - G1;
  Fr3 := Fr2 - Fr1;
  Label15.Caption := floatToStr(G3);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

    Label16.Caption := currToStr(Fr3);
end;

end;

end

else begin
    G1 := StrToFloat(Label1.Caption);
    G2 := StrToFloat(Label7.Caption);
    Fr1 := StrToCurr(Label2.Caption);
    Fr2 := StrToCurr(Label8.Caption);
    G3 := G2 - G1;
    Fr3 := Fr2 - Fr1;
    Label15.Caption := floatToStr(G3);
    Label16.Caption := currToStr(Fr3);
end;

end;

```

```

procedure TData2.FormCreate(Sender: TObject);

```

```

begin

```

```

    LineCnt := 0 ;

```

```

    OldX:=-1;           { initialize variables }

```

```

    CrossHairColor:=clRed;

```

```

    CrossHairStyle:=psSolid;

```

```

end;

```

```

procedure TData2.DBChart1MouseDown(Sender: TObject; Button: TMouseButton;

```

```

    Shift: TShiftState; X, Y: Integer);

```

```

Procedure DrawCross(AX,AY:Integer);

```

```

begin

```

```

    With DBChart1,Canvas do

```

```

    begin

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

Pen.Color:=CrossHairColor;

Pen.Style:=CrossHairStyle;

Pen.Mode:=pmXor;

Pen.Width:=1;

MoveTo(ax,ChartRect.Top-Height3D);

LineTo(ax,ChartRect.Bottom-Height3D);

MoveTo(ChartRect.Left+Width3D,ay);

LineTo(ChartRect.Right+Width3D,ay);

end;

end;

var

code : integer; lng : string[9];

begin

if LineCnt < 2 Then

begin

case LineCnt of

0 : begin

CrossHairColor:=clRed;

DrawCross(x,y);

Label1.Caption := Label11.Caption;

Label2.Caption := Label12.Caption;

LineCnt := LineCnt + 1 ;

end;

1 : begin

CrossHairColor:=clblue;

DrawCross(x,y);

Label7.Caption := Label11.Caption;

Label8.Caption := Label12.Caption;

LineCnt := LineCnt + 1 ;

end;

end; { Case }

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

end;

end;

procedure TData2.Button1Click(Sender: TObject);
begin
    DBChart1.Canvas.Refresh;
    DBChart1.Repaint;
    DBChart1.Repaint;
    CrossHairColor:=clRed;
    LineCnt := 0;
end;
end.

```



**unit** GENERATOR;

**interface**

**uses**

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

**type**

TGEN = class(TForm)

Panel1: TPanel;

Panel2: TPanel;

Panel3: TPanel;

Panel4: TPanel;

Panel5: TPanel;

Panel6: TPanel;

Shape1: TShape;

Panel7: TPanel;

Shape2: TShape;

Panel8: TPanel;

Shape3: TShape;

Panel9: TPanel;

Shape4: TShape;

Panel10: TPanel;

Button1: TButton;

Label1: TLabel;

Label2: TLabel;

Label3: TLabel;

Panel11: TPanel;

Shape5: TShape;

Panel12: TPanel;

Shape6: TShape;

Panel13: TPanel;



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Shape7: TShape;  
Panel14: TPanel;  
Shape8: TShape;  
Panel15: TPanel;  
Panel16: TPanel;  
Panel17: TPanel;  
Label4: TLabel;  
Shape9: TShape;  
Panel18: TPanel;  
Shape10: TShape;  
Panel19: TPanel;  
Shape11: TShape;  
Panel20: TPanel;  
Shape12: TShape;  
Panel21: TPanel;  
Shape13: TShape;  
Panel22: TPanel;  
Shape14: TShape;  
Panel23: TPanel;  
Button4: TButton;  
Panel24: TPanel;  
Button5: TButton;  
Panel25: TPanel;  
Button6: TButton;  
Panel26: TPanel;  
Button7: TButton;  
Panel27: TPanel;  
Button8: TButton;  
Panel28: TPanel;  
Button9: TButton;



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Panel29: TPanel;  
Button10: TButton;  
Panel30: TPanel;  
Button11: TButton;  
Panel31: TPanel;  
Button12: TButton;  
Panel32: TPanel;  
Button13: TButton;  
Panel33: TPanel;  
Shape15: TShape;  
Panel34: TPanel;  
Shape16: TShape;  
Panel35: TPanel;  
Shape17: TShape;  
Panel36: TPanel;  
Shape18: TShape;  
Panel37: TPanel;  
Shape19: TShape;  
Label5: TLabel;  
Label6: TLabel;  
SpeedButton1: TSpeedButton;  
SpeedButton2: TSpeedButton;  
Label7: TLabel;  
Label8: TLabel;  
Label9: TLabel;  
Label10: TLabel;  
Label11: TLabel;  
Label12: TLabel;  
Label13: TLabel;  
Label14: TLabel;



```
Label15: TLabel;
Label16: TLabel;
Label17: TLabel;
Label18: TLabel;
Label19: TLabel;
Label20: TLabel;
Panel38: TPanel;
Shape20: TShape;
Label21: TLabel;
Label22: TLabel;
SpeedButton3: TSpeedButton;
BitBtn1: TBitBtn;
Label23: TLabel;
Label24: TLabel;
Label25: TLabel;
Label26: TLabel;
Label27: TLabel;
procedure FormShow(Sender: TObject);
procedure SpeedButton1Click(Sender: TObject);
procedure SpeedButton2Click(Sender: TObject);
procedure Button1Click(Sender: TObject);
procedure Button4Click(Sender: TObject);
procedure Button5Click(Sender: TObject);
procedure Button6Click(Sender: TObject);
procedure Button7Click(Sender: TObject);
procedure Button8Click(Sender: TObject);
procedure Button9Click(Sender: TObject);
procedure Button11Click(Sender: TObject);
procedure Button10Click(Sender: TObject);
procedure Button13Click(Sender: TObject);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

procedure Button12Click(Sender: TObject);

procedure SpeedButton3Click(Sender: TObject);

procedure FormActivate(Sender: TObject);

private

    { Private declarations }

public

    { Public declarations }

end;

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

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

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) \*)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

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

```
AddrList = array [0..31] of SmallInt;
```

(\* 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;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

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;

TFindLstn = procedure (boardID: integer;

    padlist: AddrList;

    resultlist: AddrList;

    limit: integer); stdcall;

Tibonl = function (ud: integer;

    v: integer) : integer; stdcall;

TReadStatusByte = procedure (boardID: integer;

    address: word;

    var result: 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;

```

```

TWaitSRQ = procedure (boardID: integer;
                     var result: integer); stdcall;

```

```

var

```

```

    GEN: TGEN;

```

```

    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;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

ibwait: Tibwait;
ibwrt : Tibwrt;
DevClear: TDevClear;
FindLstn: TFindLstn;
ReadStatusByte: TReadStatusByte;
Receive: TReceive;
ReceiveSetup: TReceiveSetup;
RcvRespMsg: TRcvRespMsg;
SendDataBytes: TSendDataBytes;
SendSetup: TSendSetup;
Send: TSend;
SendIFC: TSendIFC;
WaitSRQ: TWaitSRQ;
(* Declaration for the NI-488 GPIB call *)
ibonl: Tibonl;
dvm : integer; { * device Number * }
buf : Packed array[0..32] of char;
implementation
uses SETGEN,DataTest1;
{$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);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

halt;

End;

(\* Get the addresses of the GPIB Global Variables. \*)

@AddrIbsta := GetProcAddress(Gpib32Lib, 'user\_ibsta');

@AddrIberr := GetProcAddress(Gpib32Lib, 'user\_iberr');

@AddrIbcntl := GetProcAddress(Gpib32Lib, 'user\_ibcntl');

(\* 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');

@FindLstn := GetProcAddress(Gpib32Lib, 'FindLstn');

@ibonl := GetProcAddress(Gpib32Lib, 'ibonl');

@ReadStatusByte := GetProcAddress(Gpib32Lib, 'ReadStatusByte');

@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');

@WaitSRQ := GetProcAddress(Gpib32Lib, 'WaitSRQ');

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
(@FindLstn   = NIL) Or
(@ibonl      = NIL) Or
(@ReadStatusByte = NIL) Or
(@RcvRespMsg = NIL) or
(@ReceiveSetup = NIL) or
(@Receive    = NIL) Or
(@SendDataBytes = NIL) or
(@SendSetup   = NIL) or
(@Send        = NIL) Or
(@SendIFC     = NIL) Or
(@WaitSRQ     = NIL) Then

```

Begin

```

str := 'GetProcAddresses FAILED!';
MessageDlg(str, mtError, [mbOK], 0);
(* Free the GPIB library. *)
FreeLibrary(Gpib32Lib);

```

halt;

End;

(\* Initialize GPIB global pointers to point to address location \*)

```

Pibsta := @AddrIbsta;
Piberr := @AddrIberr;
Pibcntl := @AddrIbcntl;

```

end;

**procedure** TGEN.FormShow(Sender: TObject);

**var** msg : string[12];

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

va : real;

code : integer; A : Char;

begin
    loadDLL;
    SendIFC(0);
    //buf := 'FRQ:1E3;';
    //send(0,12,buf,10,1);
    buf := 'STA?;';
    Send(0,12,buf,5,1);
    if (Pibsta^ and err) <> 0 then
        begin
            MessageDlg('Deviece Error',mtError,[mbOK],0);
            //ibonl(0,12);
            //Freelibrary(Gpib32lib);
            //halt;
            release;
        end;
    Receive(0,12,buf,35,1);
    A := buf[19];
    { * Check Signal * }
    case buf[10] of
        'T' : begin Shape5.Brush.Color := clMaroon;
                Shape6.Brush.Color := clMaroon;
                Shape7.Brush.Color := clRed;
                Shape8.Brush.Color := clMaroon;
                Shape9.Brush.Color := clMaroon;
                Shape20.Brush.Color := clMaroon;
                DataRun.Edit5.Text := 'SIN';
            end;
        'R' : begin if buf[11] = 'T' then

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

begin

Shape5.Brush.Color := clMaroon;

Shape6.Brush.Color := clRed;

Shape7.Brush.Color := clMaroon;

Shape8.Brush.Color := clMaroon;

Shape9.Brush.Color := clMaroon;

Shape20.Brush.Color := clMaroon;

DataRun.Edit5.Text := 'TRI';

end

else begin

Shape5.Brush.Color := clMaroon;

Shape6.Brush.Color := clMaroon;

Shape7.Brush.Color := clMaroon;

Shape8.Brush.Color := clMaroon;

Shape9.Brush.Color := clMaroon;

Shape20.Brush.Color := clRed;

DataRun.Edit5.Text := 'SRB';

end;

end;

'Q' : begin Shape5.Brush.Color := clMaroon;

Shape6.Brush.Color := clMaroon;

Shape7.Brush.Color := clMaroon;

Shape8.Brush.Color := clRed;

Shape9.Brush.Color := clMaroon;

Shape20.Brush.Color := clMaroon;

DataRun.Edit5.Text := 'SQR';

end;

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

'L' : begin Shape5.Brush.Color := clMaroon;
        Shape6.Brush.Color := clMaroon;
        Shape7.Brush.Color := clMaroon;
        Shape8.Brush.Color := clMaroon;
        Shape9.Brush.Color := clRed;
        Shape20.Brush.color := clMaroon;
        DataRun.Edit5.Text := 'PLS';

```

```

end;

```

```

'M' : begin Shape5.Brush.Color := clRed;
        Shape6.Brush.Color := clMaroon;
        Shape7.Brush.Color := clMaroon;
        Shape8.Brush.Color := clMaroon;
        Shape9.Brush.Color := clMaroon;
        Shape20.Brush.Color := clMaroon;
        DataRun.Edit5.Text := 'RMP';

```

```

end;

```

```

end; { End Cae Check Signal }

```

```

if buf[2] = '0' then Shape14.Brush.Color := clMaroon

```

```

else Shape14.Brush.Color := clRed;

```

```

if buf[5] = '0' then Shape13.Brush.color := clMaroon

```

```

else Shape13.Brush.Color := clRed;

```

```

{ * Check Mode * }

```

```

case buf[12] of

```

```

    'C' : begin Shape4.Brush.Color := clRed;

```

```

            Shape3.Brush.Color := clMaroon;

```

```

            Shape2.Brush.Color := clMaroon;

```

```

        end;

```

```

    'G' : begin Shape3.Brush.Color := clRed;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        Shape2.Brush.Color := clMaroon;

        Shape4.Brush.Color := clMaroon;

    end;

    'T' : begin Shape2.Brush.Color := clRed;

        Shape3.Brush.Color := clMaroon;

        Shape4.Brush.Color := clMaroon;

    end;

end; { * Case Mode * }

{ Check Sweep Mode }

if buf[8] = 'O' then Shape10.Brush.Color := clMaroon
else Shape10.Brush.Color := clRed;

if buf[16] = 'W' then
    begin Shape16.Brush.Color := clRed;

        Shape15.Brush.Color := clMaroon;

        Shape17.Brush.Color := clMaroon;

        Shape18.Brush.Color := clMaroon;

        Shape11.Brush.Color := clMaroon;

        Shape12.Brush.Color := clMaroon;

        Shape19.Brush.color := clMaroon;

        buf := "";

        buf := 'WDT?';

        Send(0,12,buf,15,1);

        Receive(0,12,buf,35,1);

        msg := buf;

        delete(msg,1,5);

        delete(msg,7,5);

        Label19.Caption := msg;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

{ Check Unit }

case buf[13] of
    '0' : Label20.Caption := 'S';
    '3' : Label20.Caption := 'mS';
    '6' : Label20.Caption := 'uS';
end; { case }

end

else
begin
    Shape16.Brush.Color := clMaroon;
    if (buf[16] = 'S') and (buf[17] = 'T') then
        begin
            Shape12.Brush.Color := clRed;
            Shape15.Brush.Color := clRed;
            buf := "";
            buf := 'STT?';
            Send(0,12,buf,5,1);
            Receive(0,12,buf,15,1);
            msg := buf ;
            delete(msg,1,4);
            delete(msg,8,5);
            Label19.Caption := msg;

            { Check Unit }

            case buf[13] of
                '0' : Label20.Caption := 'Hz';
                '3' : Label20.Caption := 'KHz';
                '6' : Label20.Caption := 'MHz';
            end; { case Check Unit }

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

end;

if (buf[16] = 'S') and (buf[17] = 'P') then
begin
    Shape15.Brush.Color := clRed;
    Shape19.Brush.Color := clRed;
    buf := "";
    buf := 'STP?';
    Send(0,12,buf,5,1);
    Receive(0,12,buf,15,1);
    msg := buf ;
    delete(msg,1,4);
    delete(msg,8,5);
    Label19.Caption := msg;

    { Check Unit }

    case buf[13] of
        '0' : Label20.Caption := 'Hz';
        '3' : Label20.Caption := 'KHz';
        '6' : Label20.Caption := 'MHz';
    end; { case Check Unit }

```

```

end;

{ Check Time Sweep }

```

```

if (buf[16] = 'S') and (buf[17] = 'W') then
begin

```

```

    Shape10.Brush.Color := clRed;
    Shape12.Brush.color := clMaroon;
    Shape19.Brush.Color := clMaroon;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

Shape15.Brush.Color := clMaroon;
Shape16.Brush.color := clMaroon;
Shape17.Brush.Color := clMaroon;
Shape18.Brush.Color := clMaroon;
buf := "";
buf := 'SWT?';
Send(0,12,buf,5,1);
Receive(0,12,buf,15,1);
msg := buf;
delete(msg,1,5);
delete(msg,7,5);
Label19.Caption := msg;
Label20.Caption := 'S';
end { if }
else if (buf[16] = 'F') and (buf[17] = 'R') then
begin
Shape15.Brush.Color := clRed;
buf := "";
buf := 'FRQ?';
Send(0,12,buf,5,1);
Receive(0,12,buf,15,1);
msg := buf ;
delete(msg,1,4);
delete(msg,8,5);
Label19.Caption := msg;
DataRun.Edit2.Text := msg;

{ Check Unit }

```

```

case buf[13] of

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        '0' : Label20.Caption := 'Hz';
        '3' : Label20.Caption := 'KHz';
        '6' : Label20.Caption := 'MHz';
    end; { case Check Unit }

end;

end; { else }

if A = 'O' then
begin
    Shape17.Brush.Color := clRed;
    Shape15.Brush.Color := clMaroon;
    Shape16.Brush.Color := clMaroon;
    Shape18.Brush.Color := clMaroon;
    Shape11.Brush.Color := clMaroon;
    buf := "";
    buf := 'OFS?';
    Send(0,12,buf,5,1);
    Receive(0,12,buf,15,1);
    Label22.Caption := 'V';
    msg := buf;
    delete(msg,1,4);
    delete(msg,5,5);
    Label21.Caption := msg;
end { if }
else begin
    buf := "";
    buf := 'AMP?';
    Send(0,12,buf,5,1);
    Receive(0,12,buf,15,1);

```

```
msg := buf;
```

```
delete(msg,1,4);
```

```
Datarun.Edit4.Text := msg ;
```

```
delete(msg,6,5);
```

```
if Shape9.Brush.color = clRed then
```

```
begin
```

```
val(msg,va,code);
```

```
va := va/2;
```

```
str(va:6:4,msg);
```

```
end;
```

```
Label21.Caption := msg;
```

```
if buf[11] = '0' then Label22.Caption := 'V'
```

```
else Label22.Caption := 'mV';
```

```
end; { else }
```

```
ibonl(0,0);
```

```
FreeLibrary(Gpib32Lib);
```

```
end;
```

```
procedure TGEN.SpeedButton1Click(Sender: TObject);
```

```
var a : char;
```

```
va : real;
```

```
code : integer;
```

```
msg : String[15];
```

```
begin
```

```
loadDLL;
```

```
SendIFC(0);
```

```
BUF := 'sta?';
```

```
send(0,12,buf,5,1);
```

```
receive(0,12,buf,35,1);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

a := buf[19];
{ *Check Signal * }
    //check := buf[11];
if buf[11]='N' then
    begin
        //check := buf[10];
        case buf[10] of
            'I' : begin
                buf := 'TRI;';
                DataRun.Edit5.Text := 'TRI';
                shape6.Brush.color := clred;
                shape5.Brush.color := clmaroon;
                shape20.Brush.color := clmaroon;
                shape7.Brush.color := clmaroon;
                shape8.Brush.color := clmaroon;
                shape9.Brush.color := clmaroon;
                send(0,12,buf,5,1);
                if a = 'A' then
                    begin
                        buf := "";
                        buf := 'AMP?';
                        Send(0,12,buf,5,1);
                        Receive(0,12,buf,15,1);
                        msg := buf;
                        delete(msg,1,4);
                        delete(msg,6,5);
                        Label21.Caption := msg;
                        if buf[11] = '0' then Label22.Caption := 'V'
                        else Label22.Caption := 'mV';
                    end;
                end;
            end;
        end;
    end;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

end;

'M' : begin
    buf := 'RMP';
    DataRun.Edit5.Text := 'RMP';
    shape5.Brush.color := clred;
    shape20.Brush.color := clmaroon;
    shape6.Brush.color := clmaroon;
    shape7.Brush.color := clmaroon;
    shape8.Brush.color := clmaroon;
    shape9.Brush.color := clmaroon;
    Send(0,12,buf,5,1);
    if a = 'A' then
    begin
        buf := "";
        buf := 'AMP?';
        Send(0,12,buf,5,1);
        Receive(0,12,buf,15,1);
        msg := buf;
        delete(msg,1,4);
        delete(msg,6,5);
        Label21.Caption := msg;
        if buf[11] = '0' then Label22.Caption := 'V'
        else Label22.Caption := 'mV';
    end;

end;

end;

end; { Case }

end

else

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

begin

case buf[11] of

'T' : begin

buf := 'RMP';

DataRun.Edit5.Text := 'RMP';

shape5.Brush.color := clred;

shape20.Brush.color := clmaroon;

shape6.Brush.color := clmaroon;

shape7.Brush.color := clmaroon;

shape8.Brush.color := clmaroon;

shape9.Brush.color := clmaroon;

send(0,12,buf,5,1);

if a = 'A' then

begin

buf := ";

buf := 'AMP?';

Send(0,12,buf,5,1);

Receive(0,12,buf,15,1);

msg := buf;

delete(msg,1,4);

delete(msg,6,5);

Label21.Caption := msg;

if buf[11] = '0' then Label22.Caption := 'V'

else Label22.Caption := 'mV';

end;

end;

'R' : begin

buf := 'SIN';

DataRun.Edit5.Text := 'SIN';

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

shape7.Brush.color := clred;
shape6.Brush.color := clmaroon;
shape20.Brush.color := clmaroon;
shape5.Brush.color := clmaroon;
shape8.Brush.color := clmaroon;
shape9.Brush.color := clmaroon;
send(0,12,buf,5,1);
if a = 'A' then
begin
    buf := "";
    buf := 'AMP?';
    Send(0,12,buf,5,1);
    Receive(0,12,buf,15,1);
    msg := buf;
    delete(msg,1,4);
    delete(msg,6,5);
    Label21.Caption := msg;
    if buf[11] = '0' then Label22.Caption := 'V'
    else Label22.Caption := 'mV';
end;

end;

'S' : begin
    buf := 'SQR';
    DataRun.Edit5.Text := 'SQR';
    shape8.Brush.color := clred;
    shape7.Brush.color := clmaroon;
    shape6.Brush.color := clmaroon;
    shape20.Brush.color := clmaroon;
    shape5.Brush.color := clmaroon;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

shape9.Brush.color := clmaroon;
send(0,12,buf,5,1);
if a = 'A' then
begin
    buf := "";
    buf := 'AMP?';
    Send(0,12,buf,5,1);
    Receive(0,12,buf,15,1);
    msg := buf;
    delete(msg,1,4);
    delete(msg,6,5);
    Label21.Caption := msg;
    if buf[11] = '0' then Label22.Caption := 'V'
    else Label22.Caption := 'mV';
end;
end;
'B' : begin
    buf := 'PLS';
    DataRun.Edit5.Text := 'PLS';
    shape9.Brush.color := clred;
    shape8.Brush.color := clmaroon;
    shape6.Brush.color := clmaroon;
    shape7.Brush.color := clmaroon;
    shape20.Brush.color := clmaroon;
    shape5.Brush.color := clmaroon;
    send(0,12,buf,5,1);
    if a = 'A' then
begin
    buf := "";

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

buf := 'AMP?';
Send(0,12,buf,5,1);
Receive(0,12,buf,15,1);
msg := buf;
delete(msg,1,4);
delete(msg,6,5);
if Shape9.Brush.color = clRed then
begin
    val(msg,va,code);
    va := va/2;
    str(va:6:4,msg);
end;
Label21.Caption := msg;
if buf[11] = '0' then Label22.Caption := 'V'
else Label22.Caption := 'mV';
end;
end;
'P' : begin
    { buf := 'TRI';
    shape5.Brush.color := clred;
    shape9.Brush.color := clmaroon;
    shape6.Brush.color := clmaroon;
    shape7.Brush.color := clmaroon;
    shape4.Brush.color := clmaroon;
    shape8.Brush.color := clmaroon;
    send(0,12,buf,5,1);    }
end;
end; { Case }
end; { else }

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        ibonl(0,12);

        FreeLibrary(Gpib32Lib);

end;

procedure TGEN.SpeedButton2Click(Sender: TObject);

var A : char;

    code : integer;

    va : real;

    msg : string[15];

begin
    loadDLL;
    SendIFC(0);
    BUF := 'sta?';
    send(0,12,buf,5,1);
    receive(0,12,buf,35,1);
    a := buf[19];
    //check := buf[11];
    if buf[11]='N' then
        begin
            //check := buf[10];
            case buf[10] of
                'I' : begin
                    buf := 'SQR;';
                    DataRun.Edit5.Text := 'SQR';
                    shape8.Brush.color := clred;
                    shape6.Brush.color := clmaroon;
                    shape20.Brush.color := clmaroon;
                    shape5.Brush.color := clmaroon;
                    shape7.Brush.color := clmaroon;
                    shape9.Brush.color := clmaroon;

```

```

send(0,12,buf,5,1);

if a = 'A' then
begin
    buf := "";
    buf := 'AMP?';
    Send(0,12,buf,5,1);
    Receive(0,12,buf,15,1);
    msg := buf;
    delete(msg,1,4);
    delete(msg,6,5);
    Label21.Caption := msg;
    if buf[11] = '0' then Label22.Caption := 'V'
    else Label22.Caption := 'mV';
end;
end;
'M' : begin
    buf := 'TRI';
    DataRun.Edit5.Text := 'TRI';
    shape6.Brush.color := clred;
    shape20.Brush.color := clmaroon;
    shape5.Brush.color := clmaroon;
    shape7.Brush.color := clmaroon;
    shape8.Brush.color := clmaroon;
    shape9.Brush.color := clmaroon;
    send(0,12,buf,5,1);
    if a = 'A' then
begin
    buf := "";
    buf := 'AMP?';
    Send(0,12,buf,5,1);

```

```

        Receive(0,12,buf,15,1);

        msg := buf;

        delete(msg,1,4);

        delete(msg,6,5);

        Label21.Caption := msg;

        if buf[11] = '0' then Label22.Caption := 'V'

        else Label22.Caption := 'mV';

    end;

end;

end; { Case }

end { if }

else

begin

    case buf[11] of

        'I' : begin

            buf := 'SIN';

            DataRun.Edit5.Text := 'SIN';

            shape7.Brush.color := clred;

            shape5.Brush.color := clmaroon;

            shape20.Brush.color := clmaroon;

            shape6.Brush.color := clmaroon;

            shape8.Brush.color := clmaroon;

            shape9.Brush.color := clmaroon;

            send(0,12,buf,5,1);

            if a = 'A' then

                begin

                    buf := "";

                    buf := 'AMP?';

                    Send(0,12,buf,5,1);

                    Receive(0,12,buf,15,1);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

msg := buf;
delete(msg,1,4);
delete(msg,6,5);
Label21.Caption := msg;
if buf[11] = '0' then Label22.Caption := 'V'
else Label22.Caption := 'mV';
end;
end;
'R' : begin
    buf := 'PLS;';
    DataRun.Edit5.Text := 'PLS';
    shape9.Brush.color := clred;
    shape7.Brush.color := clmaroon;
    shape20.Brush.color := clmaroon;
    shape5.Brush.color := clmaroon;
    shape6.Brush.color := clmaroon;
    shape8.Brush.color := clmaroon;
    send(0,12,buf,5,1);
    if a = 'A' then
        begin
            buf := "";
            buf := 'AMP?';
            Send(0,12,buf,5,1);
            Receive(0,12,buf,15,1);
            msg := buf;
            delete(msg,1,4);
            delete(msg,6,5);
            if Shape9.Brush.color = clRed then
                begin
                    val(msg,va,code);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        va := va/2;

        str(va:6:4,msg);

    end;

    Label21.Caption := msg;

    if buf[11] = '0' then Label22.Caption := 'V'

    else Label22.Caption := 'mV';

end;

end;

'S' : begin

    buf := 'ARB';

    DataRun.Edit5.Text := 'ARB';

    shape20.Brush.color := clred;

    shape8.Brush.color := clmaroon;

    shape6.Brush.color := clmaroon;

    shape9.Brush.color := clmaroon;

    shape5.Brush.color := clmaroon;

    shape7.Brush.color := clmaroon;

    send(0,12,buf,5,1);

    if a = 'A' then

        begin

            buf := "";

            buf := 'AMP?';

            Send(0,12,buf,5,1);

            Receive(0,12,buf,15,1);

            msg := buf;

            delete(msg,1,4);

            delete(msg,6,5);

            Label21.Caption := msg;

            if buf[11] = '0' then Label22.Caption := 'V'

            else Label22.Caption := 'mV';

```

```

end;

end;

'B' : begin
    shape20.Brush.color := clred;
    shape8.Brush.color := clmaroon;
    shape6.Brush.color := clmaroon;
    shape7.Brush.color := clmaroon;
    shape9.Brush.color := clmaroon;
    shape5.Brush.color := clmaroon;
end;

'P' : begin
    buf := 'TRI';
    DataRun.Edit5.Text := 'TRI';
    shape6.Brush.color := clred;
    shape9.Brush.color := clmaroon;
    shape5.Brush.color := clmaroon;
    shape7.Brush.color := clmaroon;
    shape20.Brush.color := clmaroon;
    shape8.Brush.color := clmaroon;
    send(0,12,buf,5,1);

    if a = 'A' then
        begin
            buf := "";
            buf := 'AMP?';
            Send(0,12,buf,5,1);
            Receive(0,12,buf,15,1);
            msg := buf;
            delete(msg,1,4);
            delete(msg,6,5);

```

```

Label21.Caption := msg;

if buf[11] = '0' then Label22.Caption := 'V'

else Label22.Caption := 'mV';

end;

end;

end; { Case }

end; { else }

ibonl(0,12);

Freelibrary(Gpib32Lib);

end;

procedure TGEN.Button1Click(Sender: TObject);

begin

loadDLL;

SendIFC(0);

BUF := 'sta?';

send(0,12,buf,5,1);

receive(0,12,buf,35,1);

//check := buf[12];

case buf[12] of

'C' : begin

buf := 'GTM;';

shape3.Brush.color := clred;

shape2.Brush.color := clmaroon;

shape4.Brush.color := clmaroon;

send(0,12,buf,5,1);

end;

'G': begin

buf := 'TRM;';

shape2.Brush.color := clred;

shape3.Brush.color := clmaroon;

```

```

        shape4.Brush.color := clmaroon;
        send(0,12,buf,5,1);
    end;
'T' : begin
    buf := 'CTM;';
    shape4.Brush.color := clred;
    shape2.Brush.color := clmaroon;
    shape3.Brush.color := clmaroon;
    send(0,12,buf,5,1);
end;
end; { Case }
ibonl(0,12);
FreeLibrary(Gpib32Lib);

end;
procedure TGEN.Button4Click(Sender: TObject);
begin
    loadDLL;
    SendIFC(0);
    BUF := 'sta?';
    send(0,12,buf,5,1);
    receive(0,12,buf,35,1);
    //check := buf[6];
    case buf[5] of
        '0' : begin
            buf := 'OF1;';
            shape13.Brush.Color := clred;
            send(0,12,buf,5,1);
        end;
        '1' : begin

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        buf := 'OF0;';

        shape13.Brush.Color := clmaroon;

        send(0,12,buf,5,1);

    end;

end; { Case }

ibonl(0,12);

FreeLibrary(Gpib32Lib);

end;

procedure TGEN.Button5Click(Sender: TObject);

begin

    loadDLL;

    SendIFC(0);

    BUF := 'sta?';

    send(0,12,buf,5,1);

    receive(0,12,buf,35,1);

    //check := buf[3];

    case buf[2] of

        '0' : begin

            buf := 'OT1;';

            shape14.Brush.Color := clred;

            send(0,12,buf,5,1);

        end;

        '1' : begin

            buf := 'OT0;';

            shape14.Brush.Color := clmaroon;

            send(0,12,buf,5,1);

        end;

    end; { Case }

    ibonl(0,12);

    FreeLibrary(Gpib32Lib);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

end;

procedure TGEN.Button6Click(Sender: TObject);

var msg : string[15];

begin
    loadDLL;
    SendIFC(0);
    buf := "";
    buf := 'STA?';
    Send(0,12,buf,5,1);
    Receive(0,12,buf,15,1);
    if buf[8] = '0' then
        begin
            Shape10.Brush.color := clRed;
            buf := "";
            buf := 'SW1';
            Send(0,12,buf,5,1);
            if (Shape15.Brush.Color = clRed) and (Shape19.Brush.Color = clMaroon) then
                begin
                    Shape12.Brush.Color := clRed;
                    buf := "";
                    buf := 'DST';
                    Send(0,12,buf,5,1);
                    buf := "";
                    buf := 'STT?';
                    Send(0,12,buf,5,1);
                    Receive(0,12,buf,15,1);
                    msg := buf;
                    delete(msg,1,4);
                    delete(msg,8,5);
                    Label19.Caption := msg;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

{ Check Unit }

case buf[13] of
    '0' : Label20.Caption := 'Hz';
    '3' : Label20.Caption := 'KHz';
    '6' : Label20.Caption := 'MHz';
end; { case Check Unit }
end; { if }
end
else
begin
    Shape10.Brush.Color := clMaroon;
    buf := "";
    buf := 'SW0';
    Send(0,12,buf,5,1);
end; { else }
ibonl(0,12);
Freelibrary(Gpib32Lib);

end;

procedure TGEN.Button7Click(Sender: TObject);
var msg : String[15];
begin
    loadDLL;
    SendIFC(0);
    if Shape11.Brush.color = clMaroon then
        begin
            Shape11.Brush.color := clRed;
            buf := "";

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
buf := 'DSW';  
Send(0,12,buf,5,1);  
Shape12.Brush.color := clMaroon;  
Shape19.Brush.Color := clMaroon;  
Shape15.Brush.Color := clMaroon;  
Shape16.Brush.Color := clMaroon;  
Shape17.Brush.Color := clMaroon;  
Shape18.Brush.Color := clMaroon;
```

```
end;
```

```
buf := "";
```

```
buf := 'SWT?';
```

```
Send(0,12,buf,5,1);
```

```
Receive(0,12,buf,15,1);
```

```
msg := buf;
```

```
delete(msg,1,4);
```

```
delete(msg,8,5);
```

```
Label19.Caption := msg;
```

```
case buf[13] of
```

```
'0' : Label20.Caption := 'S';
```

```
'3' : Label20.Caption := 'mS';
```

```
'6' : Label20.Caption := 'uS';
```

```
end; { case }
```

```
ibonl(0,12);
```

```
Freelibrary(Gpib32Lib);
```

```
end;
```

```
procedure TGEN.Button1Click(Sender: TObject);
```

```
var msg: String[15];
```

```
begin
```

```
loadDLL;
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

SendIFC(0);
if Shape12.Brush.Color = clMaroon then
begin
    Shape11.Brush.Color := clMaroon;
    Shape19.Brush.Color := clMaroon;
    Shape12.Brush.Color := clRed;
    buf := "";
    buf := 'DST';
    Send(0,12,buf,5,1);
end;
Shape15.Brush.color := clRed;
Shape16.Brush.Color := clMaroon;
Shape17.Brush.color := clMaroon;
Shape18.Brush.Color := clMaroon;
buf := "";
buf := 'STT?';
Send(0,12,buf,5,1);
Receive(0,12,buf,15,1);
msg := buf;
delete(msg,1,4);
delete(msg,8,5);
Label19.Caption := msg;
{ Check Unit }
case buf[13] of
    '0' : Label20.Caption := 'Hz';
    '3' : Label20.Caption := 'KHz';
    '6' : Label20.Caption := 'MHz';
end; { case Check Unit }
ibonl(0,12);
Freelibrary(Gpib32Lib);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

end;

procedure TGEN.Button9Click(Sender: TObject);
var  msg : String[15];
begin
    loadDLL;
    SendIFC(0);
    if Shape19.Brush.Color = clMaroon then
        begin
            Shape11.Brush.Color := clMaroon;
            Shape12.Brush.Color := clMaroon;
            Shape19.Brush.Color := clRed;
            buf := "";
            buf := 'DSP;';
            Send(0,12,buf,5,1);
        end;
    Shape15.Brush.color := clRed;
    Shape16.Brush.Color := clMaroon;
    Shape17.Brush.color := clMaroon;
    Shape18.Brush.Color := clMaroon;
    buf := "";
    buf := 'STP?;';
    Send(0,12,buf,5,1);
    Receive(0,12,buf,15,1);
    msg := buf;
    delete(msg,1,4);
    delete(msg,8,5);
    Label19.Caption := msg;

```

{ Check Unit }

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

case buf[13] of
    '0' : Label20.Caption := 'Hz';
    '3' : Label20.Caption := 'KHz';
    '6' : Label20.Caption := 'MHz';
end; { case Check Unit }

ibonl(0,12);
Freelibrary(Gpib32Lib);

end;

procedure TGEN.Button11Click(Sender: TObject);
var msg : String[15];
begin
    loadDLL;
    SendIFC(0);
    if Shape16.Brush.Color = clMaroon then
        begin
            Shape16.Brush.Color := clRed;
            Shape11.Brush.Color := clMaroon;
            Shape12.Brush.Color := clMaroon;
            Shape19.Brush.Color := clMaroon;
            Shape15.Brush.Color := clMaroon;
            Shape17.Brush.Color := clMaroon;
            Shape18.Brush.Color := clMaroon;
            buf := '';
            buf := 'DWT?';
            Send(0,12,buf,5,1);
            buf := '';
            buf := 'WDT?';
            Send(0,12,buf,5,1);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

Receive(0,12,buf,15,1);

msg := buf;

delete(msg,1,5);

delete(msg,7,5);

Label19.Caption := msg;

{ Check Unit }

case buf[13] of

    '0' : Label20.Caption := 'S';

    '3' : Label20.Caption := 'mS';

    '6' : Label20.Caption := 'uS';

end; { case }

end; { if }

buf := "";

//buf := 'DAM;';

// Send(0,12,buf,5,1);

buf := "";

buf := 'AMP?;';

Send(0,12,buf,5,1);

Receive(0,12,buf,15,1);

msg := buf;

delete(msg,1,4);

delete(msg,6,5);

Label21.Caption := msg;

if buf[11] = '0' then Label22.Caption := 'V'

else Label22.Caption := 'mV';

ibonl(0,12);

Freelibrary(Gpib32Lib);

end;

```

```

procedure TGEN.Button10Click(Sender: TObject);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
 ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

var msg : String[15];

begin
    loadDLL;
    SendIFC(0);
    if Shape15.Brush.Color = clMaroon then
        begin
            Shape15.Brush.Color := clRed;
            Shape16.Brush.Color := clMaroon;
            Shape17.Brush.Color := clMaroon;
            Shape18.Brush.Color := clMaroon;
            Shape11.Brush.Color := clMaroon;
            Shape19.Brush.Color := clMaroon;
            if Shape10.Brush.Color = clRed then
                begin
                    Shape12.Brush.Color := clRed;
                    buf := "";
                    buf := 'DST?';
                    Send(0,12,buf,5,1);
                    buf := "";
                    buf := 'STT?';
                    Send(0,12,buf,5,1);
                    Receive(0,12,buf,15,1);
                    msg := buf;
                    delete(msg,1,4);
                    delete(msg,8,5);
                    Label19.Caption := msg;

                    { Check Unit }

                case buf[13] of

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        '0' : Label20.Caption := 'Hz';
        '3' : Label20.Caption := 'KHz';
        '6' : Label20.Caption := 'MHz';

    end; { case Check Unit }

end { if }

else

    . begin

        buf := "";
        buf := 'DFR?';
        Send(0,12,buf,5,1);
        buf := "";
        buf := 'FRQ?';
        Send(0,12,buf,5,1);
        Receive(0,12,buf,15,1);
        msg := buf;
        delete(msg,1,4);
        delete(msg,8,5);
        Label19.Caption := msg;

        { Check Unit }

        case buf[13] of

            '0' : Label20.Caption := 'Hz';
            '3' : Label20.Caption := 'KHz';
            '6' : Label20.Caption := 'MHz';

        end; { case Check Unit }

    end; { else }

end { if UP }

```

else

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

begin
  if Shape10.Brush.Color = clMaroon then
    begin
      Shape12.Brush.Color := clMaroon;
      Shape19.Brush.Color := clMaroon;
      buf := "";
      buf := 'DFR;';
      Send(0,12,buf,5,1);
      buf := "";
      buf := 'FRQ?;';
      Send(0,12,buf,5,1);
      Receive(0,12,buf,15,1);
      msg := buf;
      delete(msg,1,4);
      delete(msg,8,5);
      Label19.Caption := msg;

      { Check Unit }

      case buf[13] of
        '0' : Label20.Caption := 'Hz';
        '3' : Label20.Caption := 'KHz';
        '6' : Label20.Caption := 'MHz';
      end; { case Check Unit }
    end; { if}
  end; { else }

```

```
buf := "";
```

```
//buf := 'DAM;';
```

```
// Send(0,12,buf,5,1);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

buf := "";
buf := 'AMP?';
Send(0,12,buf,5,1);
Receive(0,12,buf,15,1);
msg := buf;
delete(msg,1,4);
delete(msg,6,5);
Label21.Caption := msg;
if buf[11] = '0' then Label22.Caption := 'V'
else Label22.Caption := 'mV';

ibonl(0,12);
FreeLibrary(Gpib32Lib);
end;

procedure TGEN.Button13Click(Sender: TObject);
var msg : String[15];
    A : Char;
begin
    loadDLL;
    SendIFC(0);
    buf := "";
    buf := 'STA?';
    Send(0,12,buf,5,1);
    Receive(0,12,buf,35,1);
    //A := buf[17];
    if Shape18.Brush.Color = clMaroon then
        begin Shape18.Brush.Color := clRed;
            Shape15.Brush.Color := clMaroon;

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

Shape16.Brush.Color := clMaroon;
Shape17.Brush.Color := clMaroon;
buf := "";
buf := 'DAM;';
Send(0,12,buf,5,1);
buf := "";
buf := 'AMP?;';
Send(0,12,buf,5,1);
Receive(0,12,buf,15,1);
msg := buf;
delete(msg,1,4);
delete(msg,6,5);
Label21.Caption := msg;
if buf[11] = '0' then Label22.Caption := 'V'
else Label22.Caption := 'mV';

buf := "";
buf := 'STA?;';
Send(0,12,buf,15,1);
Receive(0,12,buf,35,1);
case buf[17] of
    'R' : begin
        Shape11.Brush.Color := clMaroon;
        Shape12.Brush.Color := clMaroon;
        Shape19.Brush.Color := clMaroon;
        // buf := "";
        // buf := 'DFR;';
        // Send(0,12,buf,5,1);
        buf := "";
        buf := 'FRQ?;';

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Send(0,12,buf,5,1);
Receive(0,12,buf,15,1);
msg := buf ;
delete(msg,1,4);
delete(msg,8,5);
Label19.Caption := msg;
```

```
{ Check Unit }
```

```
case buf[13] of
  '0' : Label20.Caption := 'Hz';
  '3' : Label20.Caption := 'KHz';
  '6' : Label20.Caption := 'MHz';
end; { case Check Unit }
end; { case R }
```

```
'T' : begin
  Shape11.Brush.Color := clMaroon;
  Shape12.Brush.Color := clRed;
  Shape19.Brush.Color := clMaroon;
  // buf := "";
  // buf := 'DST';
  // Send(0,12,buf,5,1);
  // buf := "";
  buf := 'STT?';
  Send(0,12,buf,5,1);
  Receive(0,12,buf,15,1);
  msg := buf;
  delete(msg,1,4);
  delete(msg,8,5);
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Label19.Caption := msg;
```

```
{ Check Unit }
```

```
case buf[13] of
```

```
  '0' : Label20.Caption := 'Hz';
```

```
  '3' : Label20.Caption := 'KHz';
```

```
  '6' : Label20.Caption := 'MHz';
```

```
end; { case Check Unit }
```

```
end; { Case T }
```

```
'P' : begin
```

```
  Shape11.Brush.Color := clMaroon;
```

```
  Shape19.Brush.Color := clRed;
```

```
  Shape12.Brush.Color := clMaroon;
```

```
  //buf := ";
```

```
  //buf := 'DSP?';
```

```
  //Send(0,12,buf,5,1);
```

```
  buf := ";
```

```
  buf := 'STP?';
```

```
  Send(0,12,buf,5,1);
```

```
  Receive(0,12,buf,15,1);
```

```
  msg := buf;
```

```
  delete(msg,1,4);
```

```
  delete(msg,8,5);
```

```
  Label19.Caption := msg;
```

```
{ Check Unit }
```

```
case buf[13] of
```

```
  '0' : Label20.Caption := 'Hz';
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

        '3' : Label20.Caption := 'KHz';
        '6' : Label20.Caption := 'MHz';

    end; { case Check Unit }

    end; { Case P }

    end; { Case }

    end; { if }

    ibonl(0,12);

    Freelibrary(Gpib32Lib);

end;

procedure TGEN.Button12Click(Sender: TObject);
var msg : String[15];
    A : Char;
begin
    loadDLL;
    SendIFC(0);
    buf := "";
    buf := 'STA?';
    Send(0,12,buf,5,1);
    Receive(0,12,buf,35,1);
    //A := buf[17];
    if Shape17.Brush.Color = clMaroon then
        begin Shape17.Brush.Color := clRed;

            Shape15.Brush.Color := clMaroon;
            Shape16.Brush.Color := clMaroon;
            Shape18.Brush.Color := clMaroon;

            buf := "";
            buf := 'DOF';
            Send(0,12,buf,5,1);

            buf := "";
            buf := 'OFS';

```

```
Send(0,12,buf,5,1);
Receive(0,12,buf,15,1);
msg := buf;
delete(msg,1,4);
delete(msg,5,5);
Label21.Caption := msg;
if buf[10] = '0' then Label22.Caption := 'V'
else Label22.Caption := 'mV';
```

```
buf := "";
buf := 'STA?';
Send(0,12,buf,15,1);
Receive(0,12,buf,35,1);
case buf[17] of
  'R' : begin
    Shape11.Brush.Color := clMaroon;
    Shape12.Brush.Color := clMaroon;
    Shape19.Brush.Color := clMaroon;
    // buf := "";
    // buf := 'DFR?';
    // Send(0,12,buf,5,1);
    buf := "";
    buf := 'FRQ?';
    Send(0,12,buf,5,1);
    Receive(0,12,buf,15,1);
    msg := buf;
    delete(msg,1,4);
    delete(msg,8,5);
    Label19.Caption := msg;
```

```
{ Check Unit }
```

```
case buf[13] of
```

```
'0' : Label20.Caption := 'Hz';
```

```
'3' : Label20.Caption := 'KHz';
```

```
'6' : Label20.Caption := 'MHz';
```

```
end; { case Check Unit }
```

```
end; { case R }
```

```
'T' : begin
```

```
Shape11.Brush.Color := clMaroon;
```

```
Shape12.Brush.Color := clRed;
```

```
Shape19.Brush.Color := clMaroon;
```

```
// buf := ";
```

```
// buf := 'DST';
```

```
// Send(0,12,buf,5,1);
```

```
// buf := ";
```

```
buf := 'STT?';
```

```
Send(0,12,buf,5,1);
```

```
Receive(0,12,buf,15,1);
```

```
msg := buf;
```

```
delete(msg,1,4);
```

```
delete(msg,8,5);
```

```
Label19.Caption := msg;
```

```
{ Check Unit }
```

```
case buf[13] of
```

```
'0' : Label20.Caption := 'Hz';
```

```
'3' : Label20.Caption := 'KHz';
```

```

        '6' : Label20.Caption := 'MHz';
    end; { case Check Unit }
end; { Case T }

'P' : begin
    Shape11.Brush.Color := clMaroon;
    Shape19.Brush.Color := clRed;
    Shape12.Brush.Color := clMaroon;

    //buf := "";
    //buf := 'DSP?';
    //Send(0,12,buf,5,1);
    buf := "";
    buf := 'STP?';
    Send(0,12,buf,5,1);
    Receive(0,12,buf,15,1);
    msg := buf;
    delete(msg,1,4);
    delete(msg,8,5);
    Label19.Caption := msg;

    { Check Unit }

case buf[13] of
    '0' : Label20.Caption := 'Hz';
    '3' : Label20.Caption := 'KHz';
    '6' : Label20.Caption := 'MHz';
end; { case Check Unit }

end; { Case P }

end; { Case }

end; { if }

ibonl(0,12);

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า  
ไม่ว่ากรณีใดๆ ทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหาและต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



Freelibrary(Gpib32Lib);

end;

procedure TGEN.SpeedButton3Click(Sender: TObject);

begin

SETG.SHOW;

end;

procedure TGEN.FormActivate(Sender: TObject);

begin

Gen.Show;

end;

end.

