การควบคุมเครื่องมือวัดผ่าน GPIB (IEEE 488)

Instrument Control via GPIB (IEEE 488)

โดย

นาย ชวลิต ฉลองกลาง

นาย ณัฐวุฒิ ชูลิขิต

โครงงานนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตรบัณฑิต

สาขาเทคโนโลยีโทรคมนาคม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2544

หัวข้อปริญญานิพนธ์	**การควบคุมเครื่องมือวัดผ่าน GPIB (IEEE 488)**
	Instrument Control via GPIB (IEEE 488)
โดย	นาย ชวลิต ฉลองกลาง รหัส 42015680
	นาย ณัฐวุฒิ ชูลิขิต รหัส 42015684
อาจารย์ที่ปรึกษา	อาจารย์ มนต์ชัย แช่มช้อย
ปริญญา	อุตสาหกรรมศาสตรบัณฑิต
สาขาวิชา	เทคโนโลยีโทรคมนาคม
ภาควิชา	เทคนิคอุตสาหกรรม
ปีการศึกษา	2544

คณะวิศวกรรมศาสตร์สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบังอนุมติให้นับปริญญานิพนธ์ ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาอุตสาหกรรมศาสตรบัณฑิต

คณะกรรมการสอบปริญญานิพนธ์

_____ประธานกรรมการ
(_____)
_____กรรมการ
(_____)
_____กรรมการ
(_____)
_____กรรมการ
(_____)
_____กรรมการ
(_____)

ลิขสิทธิ์ของคณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

หัวข้อปริญญานิพนธ์	**การควบคุมเครื่องมือวัดผ่าน GPIB (IEEE 488)**
	Instrument Control via GPIB (IEEE 488)
โดย	นาย ชวลิต ฉลองกลาง รหัส 42015680
	นาย ณัฐวุฒิ ชูลิขิต รหัส 42015684
อาจารย์ที่ปรึกษา	อาจารย์ มนต์ชัย แช่มช้อย
ปริญญา	อุตสาหกรรมศาสตรบัณฑิต
สาขาวิชา	เทคโนโลยีโทรคมนาคม
ภาควิชา	เทคนิคอุตสาหกรรม
ปีการศึกษา	2544

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้ นำเสนอเกี่ยวกับการควบคุมเครื่องมือวัดผ่านคอมพิวเตอร์ ภายใต้มาตรฐาน IEEE 488 โดยการอินเตอร์เฟซกันระหว่างคอมพิวเตอร์กับเครื่องมือวัด ผ่านการ์ด GPIB เพื่อใช้จำลองเครื่องมือวัด บนคอมพิวเตอร์ โดยใช้โปรแกรม HP VEE และ Matlab ในการควบคุมสเปกตรัมอนาไลเซอร์ และ เนทเวิร์กอนาไลเซอร์ เครื่องมือวัดสามารถรับค่าจากคอมพิวเตอร์ผ่านแป้นพิมพ์ และเมาส์ ซึ่งการสั่งงานผ่านคอมพิวเตอร์ ช่วยให้การเก็บข้อมูล การแสดงผล และการกระทำทางคณิตศาสตร์ต่อข้อมูล สามารถทำได้สะดวกยิ่งขึ้นโดยผ่านการโปรแกรม ซึ่งการใช้งานสามารถประยุกต์ได้หลากหลายยิ่งขึ้น

TITLE	**Instrument Control via GPIB (IEEE 488)**	
STUDENT	Mr. Shavalit Chalongklang	42015680
	Mr. Nuttawoot Chulikit	42015684
ADVISOR	Mr. Monchai Chamchoy	
DEGREE	Bachelor of Industrial Technology	
PROGRAMME	Telecommunication Technology	
DEPARTMENT	Industrial Technology	
ACADEMIC YEAR	2001	

ABSTRACT

This thesis presents instrument controlled by interfacing of computer via GPIB card by applying HP VEE program and Matlab which control Spectrum Analyzer and Network Analyzer respectively. For data input, the instrument will be received the data via computer's keyboard and mouse. This program will assist users to store, display, calculate, simulate data easily. This Project can be applied variously.

กิตติกรรมประกาศ

ปริญญานิพนธ์ฉบับนี้ ได้รับความช่วยเหลือในการให้คำแนะนำรวมถึงข้อมูลอย่างดียิ่งจาก อาจารย์ที่ปรึกษาปริญญานิพนธ์คือ อาจารย์ มนต์ชัย แช่มช้อย ซึ่งได้ให้คำแนะนำและการสนับสนุน ในการวิจัยมาด้วยดีตลอดการทำงาน

ขอขอบพระคุณบิดา มารดา ที่ช่วยให้กำลังใจ และให้การสนับสนุน ในทุกด้านรวมถึงในด้านทุนทรัพย์ระหว่างการทำโครงงานมาโดยตลอด ขอขอบคุณเพื่อนๆทุกคนที่ให้กำลังใจกับการทำโครงงานชิ้นนี้

นาย ชวลิต ฉลองกลาง

นาย ณัฐวุฒิ ชูลิขิต

สารบัญ

สารบัญ (ต่อ)

สารบัญรูปภาพ

ภาพ หน้า

สารบัญรูปภาพ(ต่อ)

ภาพ หน้า

สารบัญรูปภาพ(ต่อ)

ภาพ หน้า

สารบัญตาราง

บทที่ 1
บทนำ

ในปัจจุบันคอมพิวเตอร์เข้ามามีบทบาทในชีวิตของมนุษย์ค่อนข้างมากเพราะความที่คอมพิวเตอร์สามารถอำนวยความสะดวกให้กับการทำงานได้ค่อนข้างดีเยี่ยม และค่อนข้างแม่นยำ อีกทั้งการบันทึกหรือเก็บข้อมูลไว้ในคอมพิวเตอร์ก็ยังทำได้ง่ายและคงทน ซึ่งในงานทางด้านการสื่อสารนั้น เราได้นำเอาคอมพิวเตอร์มาร่วมใช้งานกับอุปกรณ์หลากหลายกับคอมพิวเตอร์ในที่นี้เรานำเอาคอมพิวเตอร์มาใช้งานร่วมกับ เครื่องมือวัดมาอินเตอร์เฟซกันโดยผ่านการ์ด GPIB ซึ่งการเชื่อมต่อระหว่างกันของอุปกรณ์ทั้งสองจะช่วยให้การควบคุมง่ายขึ้น อีกทั้งง่ายต่อการบันทึกผลการวัด หรือ การทดลอง

1. จุดมุ่งหมาย

เพื่อศึกษาการควบคุมการทำงานของอุปกรณ์เครื่องมือวัด โดยการควบคุมการทำงานของเครื่องมือวัดนั้นได้ทำการควบคุมผ่านทางการโปรแกรม โดยใช้คอมพิวเตอร์ ซึ่งสามารถที่จะแสดงผลผ่านทาง หน้าจอมอนิเตอร์ของคอมพิวเตอร์ เครื่องมือวัดสามารถรับค่าจากคอมพิวเตอร์ผ่านแป้นพิมพ์ และเมาส์ ซึ่งการสั่งงานผ่านคอมพิวเตอร์ ช่วยให้การเก็บข้อมูลและการแสดงผลของข้อมูลนั้นทำได้สะดวกมากยิ่งขึ้น

2. แนวความคิดและที่มา

จากการที่ได้มีโอกาสใช้เครื่องมือวัดไม่ว่าจะเป็น เครื่องสเปกตรัมอนาไลเซอร์ หรือเนทเวิร์กอนาไลเซอร์นั้น พบว่าการทำงานและการแสดงผลของเครื่องมือวัดทั้ง 2 ชนิดนั้นยังมีข้อจำกัดและขีดความสามารถอยู่พอสมควร จึงทำให้เกิดแนวความคิดที่ว่า หากนำคอมพิวเตอร์มาทำการต่ออินเตอร์เฟซกับเครื่องมือวัดแล้วนั้น น่าที่จะสามารถเพิ่มขีดความสามารถของการใช้งาน ได้ไม่มากก็น้อย อีกทั้งในปัจจุบันนี้คอมพิวเตอร์ ถือเป็นอุปกรณ์ที่มีส่วนสำคัญ กับเทคโนโลยีในทุกๆด้านอย่างกว้างขวางมาก เราจึงมีแนวความคิดที่ว่าหากเราสามารถควบคุมอุปกรณ์เครื่องมือวัด ผ่านทางคอมพิวเตอร์ได้ น่าจะอำนวยความสะดวก ในการใช้งานได้มากขึ้น และน่าจะทำให้เราใช้งานได้หลากหลายมากขึ้นด้วย ซึ่งการควบคุมการทำงานของอุปกรณ์เครื่องมือวัดผ่านทางโปรแกรมคอมพิวเตอร์ (Software) จะช่วยให้เราสามารถจดบันทึกและเก็บสถิติของผลการทดลองได้ง่ายขึ้น

3. ขอบเขตของโครงงาน

สามารถที่จะควบคุมเครื่องมือวัด โดยผ่านการ์ดอินเตอร์เฟซ GPIB ได้ โดยใช้การเขียนโปรแกรมคอมพิวเตอร์เพื่อทำการควบคุมและสั่งการ การกระทำใดๆของเครื่องมือวัด ให้ทำงานตามคำสั่ง และสามารถเก็บบันทึกผลของการทดลอง

4. โครงงานประกอบด้วยส่วนสำคัญ 2 ส่วนใหญ่ๆคือ

4.1 ส่วนที่เป็นฮาร์ดแวร์ (Hardware) คือส่วนของการ์ดอินเตอร์เฟซ (Interface Card) เครื่องคอมพิวเตอร์ และเครื่องมือวัด (Instrument) เครื่องสเปคตรัมอนาไลเซอร์ (Spectrum Analyzer) และ เครื่องเนทเวิร์กอนาไลเซอร์ (Network Analyzer)

4.2 ส่วนที่เป็นซอฟท์แวร์ (Software) คือส่วนที่ใช้ในการโปรแกรมโดยมีการ์ดอินเตอร์เฟซ (Interface Card) เป็นตัวเชื่อมโยงระหว่างคอมพิวเตอร์กับสเปคตรัมอนาไลเซอร์ (Spectrum Analyzer) และ เนทเวิร์กอนาไลเซอร์ (Network Analyzer)

รูปที่ 1 บล๊อกไดอะแกรม

บล๊อกไดอะแกรม (Block Diagram) ของโครงงานประกอบด้วยส่วนต่างๆดังรูป

- พีซี อินเตอร์เฟซ (PC Interface) ซึ่งจะทำหน้าที่แปลงข้อมูลดิจิตอล (Digital) ให้อยู่ในรูปแบบ (Format) ที่คอมพิวเตอร์สามารถนำไปใช้งานได้
- ซอฟท์แวร์ (Software) ควบคุมจะทำหน้าที่ในการควบคุมการทำงานของการ์ด อินเตอร์เฟซ (Card Interface) ให้ทำงานตามที่เรากำหนด
- เครื่องมือวัด (Instrument) เป็นอุปกรณ์ที่เราต้องการเขียนโปรแกรมเพื่อควบคุม

บทที่ 2

IEEE-488 (GPIB)

ในการเชื่อมต่ออุปกรณ์ภายนอกกับคอมพิวเตอร์ได้มีการนำเอามาตรฐานในการติดต่อ IEEE 488.1-1987 หรือที่นิยมเรียกกันว่า GPIB (General Purpose Interface Bus) ซึ่งมีลักษณะในการใช้งานที่เกี่ยวข้องกับทางด้านอิเล็กทรอนิกส์ และทางด้านเครื่องกล

GPIB จะมีลักษณะเป็นสัญญาณ ดิจิตอลโดยพื้นฐานที่สำคัญจะประกอบด้วย

- การส่งข้อมูลแบบขนาน มีขนาด 8 บิท มีความเร็วในการส่ง 1 เมกะไบท์ต่อวินาที (Mbytes/S)
- การใช้สายสัญญาณ 3 เส้น ในการแฮนด์เช็ค (Handshake) และใช้สายสัญญาณ 5 เส้น ในการควบคุม (Controller)

และได้มีการพัฒนามาเป็น IEEE 488.2-1992 โดยใช้กติกาการติดต่อแบบบัส มีการเซ็ทชนิดของข้อมูลและรูปแบบของข้อมูล

ในการใช้งานอุปกรณ์ GPIB เราสามารถที่จะใช้โปรแกรมมาควบคุมการทำงานของเครื่องมือวัดแต่ละตัวได้ ซึ่งในโครงงานนี้เราได้ใช้ โปรแกรม HP VEE สำหรับควบคุม

2.1 โครงสร้างของงาน IEEE-488

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

- Talker ทำหน้าที่เป็นตัวรับข้อมูล โดยในระบบเดียวกันสามารถมี Talker ได้หลายตัว แต่จะมีเพียงตัวเดียวเท่านั้นที่กำลังทำงานอยู่
- Listener ทำหน้าที่เป็นตัวรับข้อมูล โดยในระบบเดียวกันสามารถมี Listener ได้หลายตัว เช่นเดียวกัน แต่ Listener ทำการรับข้อมูล

อุปกรณ์ที่มี GPIB นั้นสามารถแบ่งตามหน้าที่ได้ดังนี้

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

3. หน้าที่เป็นทั้ง Talker และ Listener เช่น คอมพิวเตอร์, เครื่องมือวัดที่สามารถ ควบคุม ได้จากภายนอก เป็นต้น

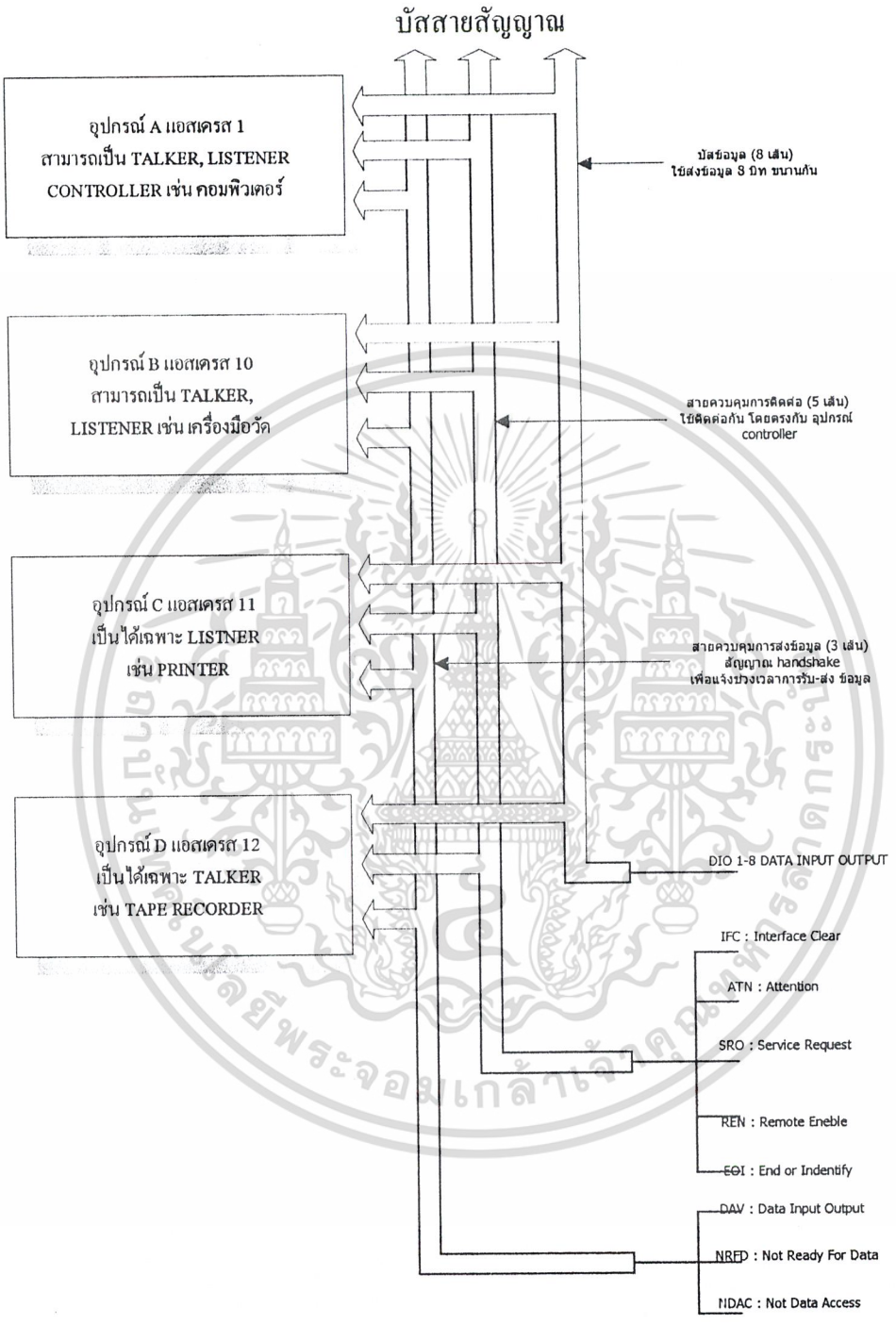
4. หน้าาที่เป็น Talker Listener และ Controller ในตัวเดียวกัน เช่น คอมพิวเตอร์ที่ทำหน้าที่ควบคุมระบบ

2.2 ขีดจำกัดของ IEEE-488

1. จำนวนอุปกรณ์ในระบบ (Talker, Listener, Controller) ที่ต่อกับสายสัญญาณ 1 เส้น จะต้องไม่เกิน 15 เครื่อง
2. สายเคเบิ้ลที่ต่อระหว่างอุปกรณ์ จะต้องยาวไม่เกิน 4 เมตร และความยาวรวมของสายเคเบิ้ล ในระบบจะต้องไม่เกิน 20 เมตร
3. ความเร็วในการส่งข้อมูลจะต้องไม่เกิน 1 Mb/sec (1 ล้านไบต์ต่อวินาที)
4. ต้องมีการจ่ายไฟให้อุปกรณ์มากกว่าครึ่งหนึ่งของระบบ

2.3 รายละเอียดเกี่ยวกับ IEEE-488

ลักษณะทางกายภาพ IEEE-488 นั้นคือ เป็นสายสัญญาณแบบ 24 เส้นขนานกัน และ มี ขั้วต่ออยู่ทางปลายทั้งสองของสาย เพื่อต่อกันเพื่อให้สายสัญญาณมีความยาวเพิ่มขึ้น ในจำนวนสายสัญญาณ 24 เส้น มีเพียง 16 เส้นเท่านั้น ที่ทำหน้าที่นำสัญญาณ ส่วนที่เหลืออีก 8 เส้นทำหน้าที่กราวนด์ (ground) และชีลด์ (shield)



รูปที่ 2.1 การแบ่งเส้นสายนำสัญญาณ

โดยจำนวนสายที่ใช้นำสัญญาณ 16 เส้นนั้นยังแบ่งได้เป็น 3 ประเภท ตามรูปที่ 2.1 คือ

1. บัสข้อมูล (Data Bus) จำนวน 8 สาย คือ

   DI01 – DI08

2. สายสัญญาณควบคุม (Control Line) จำนวน 5 สาย คือ

   IFC (Interface Clear)

   ATN (Attention)

   SRQ (Service Enable)

   REN (Remote Enable)

   EOI (End or Identify)

3. สายแฮนด์เชค (Hand Shake) 3 สาย คือ

   DAV (Data Valid)

   NRFD (Not Ready For Data)

   NDAC (Not Data Accepted)

2.4 ความหมายของสัญญาณต่างๆ ภายใน IEEE-488

ดังที่ได้กล่าวมาแล้วว่าสายสัญญาณต่างๆ ใน GPIB ได้แบ่งออกเป็น 3 กลุ่ม ในหัวข้อนี้จะอธิบายความหมายของสัญญาณต่างๆ ดังนี้

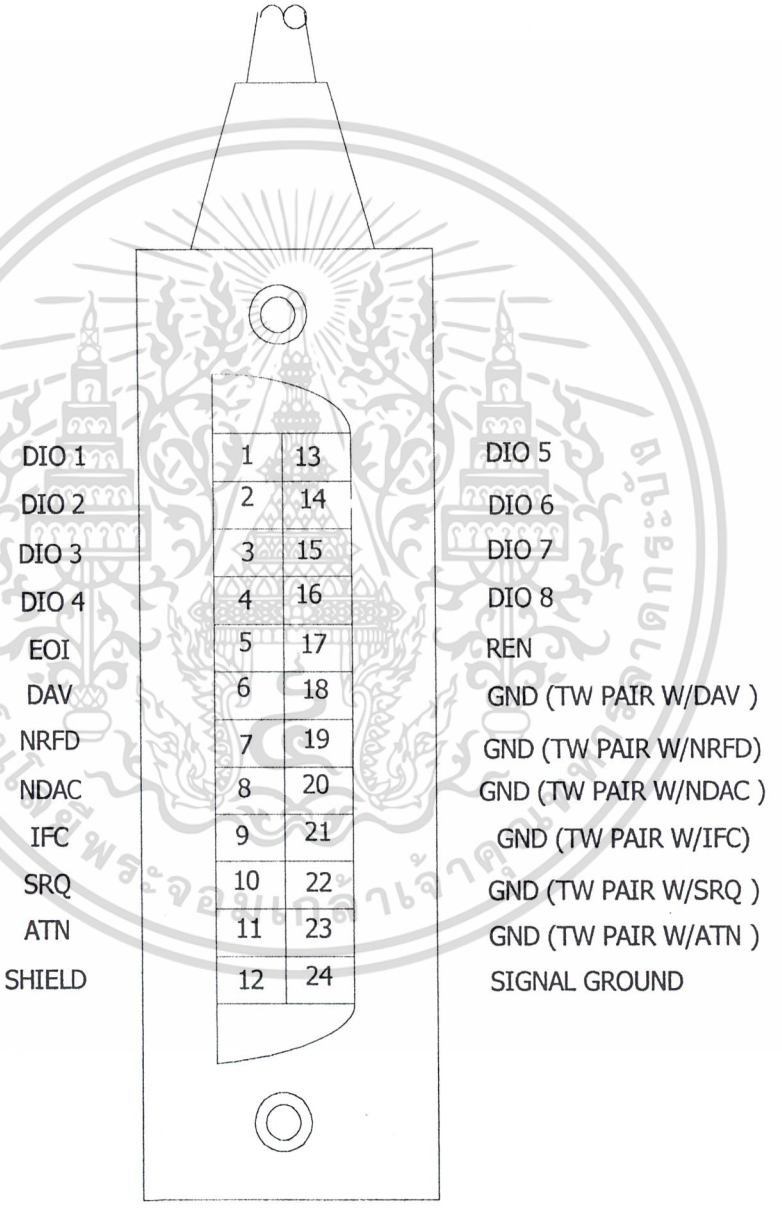
2.4.1 กลุ่มสัญญาณข้อมูล

DI01 – DI08 (Data Input/Output) สายสัญญาณทั้ง 8 เส้นนี้ ทำหน้าที่เป็นทางผ่านของข้อมูลในระบบ

2.4.2 กลุ่มสัญญาณควบคุมการเชื่อมต่อ (Interface)

1. IFC (Interface Clear) เป็นสัญญาณรีเซตหรือเคลียร์ระบบกำเนิดได้โดยตัวควบคุม (Controller) เท่านั้นเมื่ออุปกรณ์ในบัสได้รับสัญญาณเคลียร์นี้จะกลับคืนสู่สภาวะเริ่มต้นใหม่ซึ่งเป็น สภาวะแรกเริ่มก่อนกำหนดฟังก์ชันเหมือนแรกเปิดสวิตซ์
2. ATN (Attention) เป็นสัญญาณที่ถูกส่งโดยอุปกรณ์ที่เป็นตัวควบคุมเช่นเดียวกัน ใช้ในการส่งให้อุปกรณ์ทุกตัวในระบบเตรียมพร้อมเพื่อรอรับคำสั่งต่อไป
3. SRQ (Service Request) เป็นสัญญาณที่ถูกส่งจากอุปกรณ์ต่างๆเพื่อเป็นการบอกแก่ระบบว่าขณะนี้อุปกรณ์ดังกล่าวต้องการติดต่อจากตัวควบคุม

4. (Remote Enable) สัญญาณนี้ เป็นสัญญาณที่ถูกส่งมาจากตัวควบคุมเพียงตัวเดียว เท่านั้น เพื่อใช้สั่งให้อุปกรณ์ต่างๆ เปลี่ยนจาก REN โหมดที่ใช้งานปกติมาเป็นการควบคุมโดยตัวควบคุม แทน
5. EOI (End or Identify) เป็นสัญญาณที่ถูกส่งได้โดยอุปกรณ์ที่เป็นตัวควบคุม (Controller) หรืออุปกรณ์ ที่เป็นตัวส่ง (Talker) ก็ได้ ใช้สำหรับแสดงว่าข่าวสารที่ส่งเป็นชุดนั้นได้เสร็จสิ้นลงแล้ว



รูปที่ 2.2 ขั้วต่อของ GPIB และการจัดขาของสัญญาณต่างๆ

2.4.3 กลุ่มสัญญาณควบคุมการรับ-ส่งข้อมูล

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

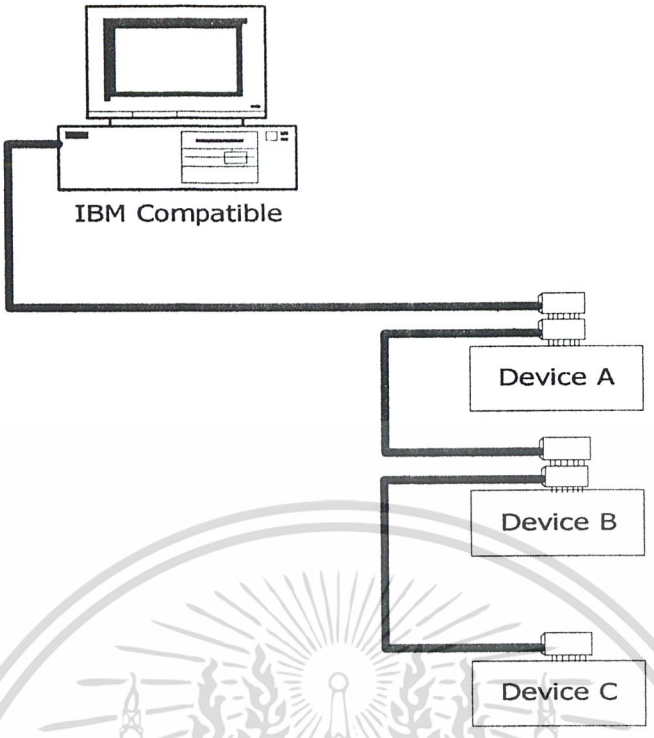
2. NRFD (Not Ready For Data) เมื่อสัญญาณนี้มีลอจิกเป็น "Low" จะเป็นการแสดงว่าในขณะนี้ระบบบัสยังไม่พร้อมที่จะรับข้อมูล เนื่องจากอุปกรณ์ในระบบยังไม่พร้อมหมดทุกตัว ซึ่ง สัญญาณเส้นนี้จะไม่เป็น "Hi" จนกว่าอุปกรณ์ทุกตัวให้ลอจิกที่เป็น "Hi" จนครบถ้วนแล้ว สัญญาณนี้มีประโยชน์ในกรณีที่อุปกรณ์ในระบบมีความเร็วแตกต่างกัน

3. NDAC (Not Data Accpeted) สัญญาณเส้นนี้เป็นสัญญาณที่ถูกควบคุมโดยอุปกรณ์ที่เป็นตัวรับ (Listener) โดยสัญญาณนี้จะมีลอจิกเป็น "Low" นะขณะที่อุปกรณ์ที่เป็นตัวรับกำลังเก็บ ข้อมูล จากสายข้อมูล (Data Bus) และจะเป็น "Hi" เมื่ออุปกรณ์นั้นได้ทำการอ่านข้อมูลเสร็จเรียบร้อยแล้ว
   โดยสัญญาณลอจิกที่ใช้ใน DATA BUS (D1-D8) ของ IEEE-488 มีลักษณะเป็นคอมพลีเมนต์ทั้งหมด คือ "1" เท่ากับ "Low" และ "0" เท่ากับ "Hi" ซึ่งตรงกันข้ามกับในวงจรที่เราคุ้นเคย

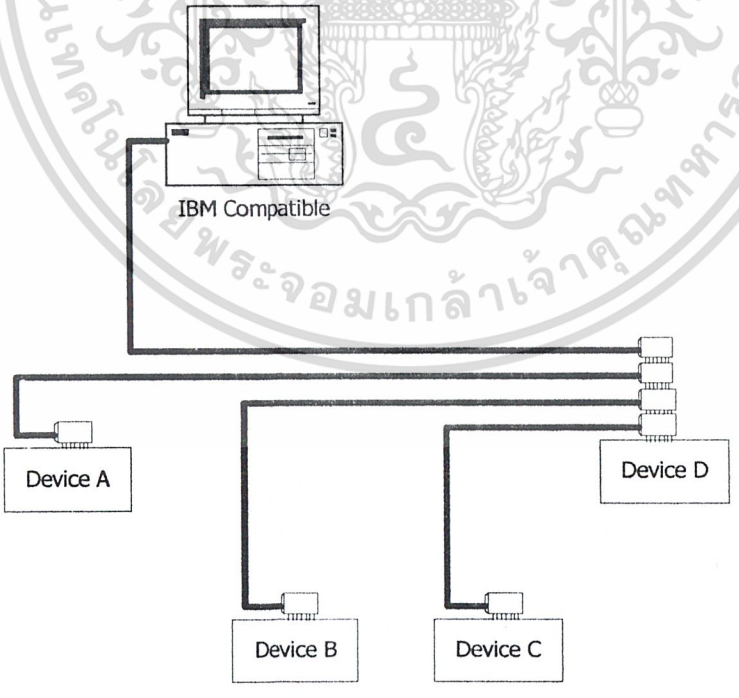
2.5 การเชื่อมต่ออุปกรณ์ต่างๆ ในระบบ IEEE-488 BUS

สำหรับการเชื่อมต่อระหว่างอุปกรณ์ต่างๆ ในระบบ IEEE-488 Bus นั้น มีอยู่ 2 วิธี คือ

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



รูปที่ 2.3 การเชื่อมต่อแบบเรียงต่อเนื่องกัน (Daisy Chain Cionfiguration)



รูปที่ 2.4 การเชื่อมต่อแบบกระจาย (Star Configuration)

2.6 ขบวนการแฮนด์เชค (Handshake Procedure)

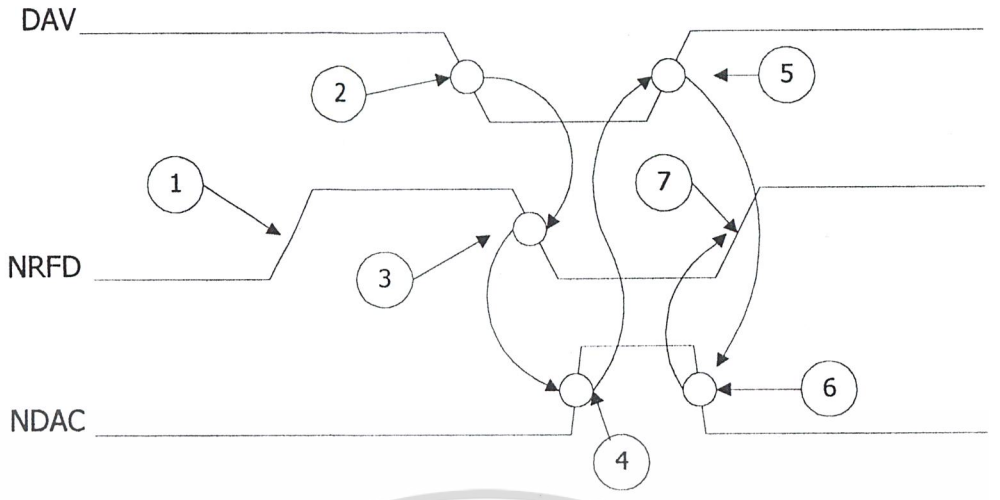
เมื่อมีการเชื่อมต่อระหว่างอุปกรณ์ต่างๆ ในระบบ ดังนั้น เมื่อระบบจะทำงานจึงต้องมีการตรวจสอบเสียก่อน ซึ่งการตรวจสอบนี้เราจะเรียกว่าขบวนการแฮนด์เชค (Handshake Procedure)

รูปที่ 2.6 ขบวนการแฮนด์เชค (Handshake Procedure)

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

การที่จะมีการสื่อสารระหว่างตัวส่งและตัวรับจะกระทำได้โดยการที่ตัวส่ง จะทำหน้าที่ส่งสัญญาณออกมาเพื่อให้ตัวรับทราบว่ามีข้อมูลเสร็จในสายสัญญาณและตัวรับก็จะส่งสัญญาณเพื่อให้ตัวส่งทราบว่าได้ทำการรับข้อมูลเป็นที่เรียบร้อยแล้ว

การส่งสัญญาณในการสื่อสารระหว่างตัวส่งและตัวรับนั้น จะส่งมาทางสายสัญญาณ 3 สาย ซึ่งสายสัญญาณดังกล่าวนั้นเป็นสายสัญญาณในกลุ่มควบคุมการรับ-ส่งข้อมูลคือ NRFD (Not Ready For Data), DAV (Data Valid) และ NDAC เป็นสัญญาณที่ถูกส่งโดยตัวรับโดยตัวรับจะใช้สัญญาณ NDAC เพื่อชี้ให้เห็นว่าพร้อม หรือ ไม่พร้อมที่จะข้อมูล

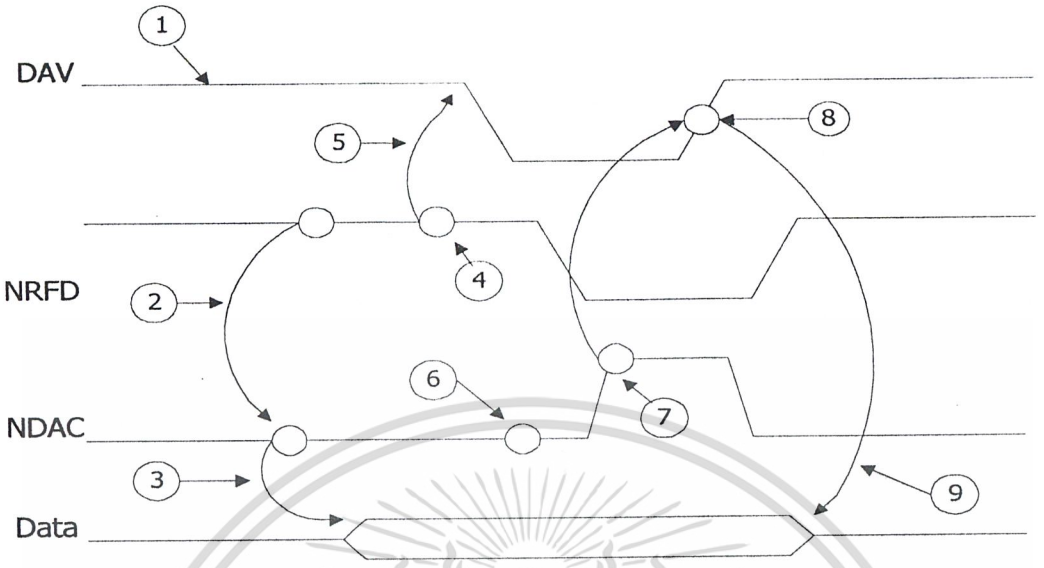


NOTE : The timing show is relative

รูปที่ 2.7 timing diagram ของขบวนการแฮนด์เชค

ขบวนการแฮนด์เชคเริ่มขึ้นเมื่อ ตัวส่งมีข้อมูลที่จะถ่ายทอดลงสายสัญญาณไปยังตัวเริ่มรับเริ่มด้วย การที่ตัวรับจะทำให้สัญญาณ NRFD เป็น “Hi” (มีค่าเป็น 0) นั่นคือ เป็นการบอกให้ทราบว่าตัวรับ พร้อมที่จะให้ตัวส่งทำการถ่ายข้อมูลลงในสายสัญญาณแล้ว (ในจุดที่ 1) ตัวส่งจะทำการถ่ายข้อมูล ลงในสายสัญญาณ DI01-DI08 หลังจากทำการรอเวลาเพื่อให้ตัวส่งถ่ายข้อมูลและตัวส่งจะทำให้ สัญญาณ DAV มีลอจิกเป็น “Low” (มีค่าเป็น 1) เพื่อที่จะบอกให้ทราบว่าขณะนี้ได้มีข้อมูลในสาย สัญญาณทั้ง 8 เส้นแล้ว (ในจุดที่ 2) ต่อมาเมื่อผู้รับทราบว่ามีข้อมูลอยู่ในบัสก็จะทำให้สัญญาณ NRFD ให้มีค่าเป็น “Low” เพื่อบอกให้ตัวส่งทราบว่ายังไม่พร้อมที่จะให้ตัวส่งทำการส่งข้อมูลชุดต่อไป และตัวรับพร้อมที่จะเก็บข้อมูลจากสายสัญญาณ (ในจุดที่ 3) หลังจากที่ตัวรับทำการเก็บข้อมูลจาก สายสัญญาณเสร็จแล้ว ตัวรับจะทำให้สัญญาณ NDAC ให้มีสถานะเป็น “Hi” เพื่อบอกให้ทราบว่าได้ ทำการเก็บข้อมูลเสร็จแล้ว ตัวรับจะทำให้สายสัญญาณ NDAC ให้มีสถานะเป็น “Hi” เพื่อบอกให้ทราบว่าได้ ทำการเก็บข้อมูลเสร็จแล้ว (ในจุดที่ 4) เมื่อตัวส่งตรวจพบว่าสัญญาณ NDAC มีลอจิกเป็น “Hi” ก็จะทำให้สัญญาณ NDAC ให้เป็น “Low” ทำให้ข้อมูลในบัสถูกกำจัดออกไป หลังจากนั้นก็จะทำให้สัญญาณ NRFD ให้เป็น “Hi” เพื่อบอกให้ทราบ ว่าพร้อมที่จะรับข้อมูลชุดต่อไปแล้วขบวนการแฮนด์เชคจึงเสร็จสิ้นลง และจะเริ่มขึ้นใหม่เมื่อมีสัญญาณ ชุดใหม่เข้ามา

2.7 ขบวนการแฮนด์เชคของตัวควบคุมซึ่งทำหน้าที่เป็นตัวส่ง

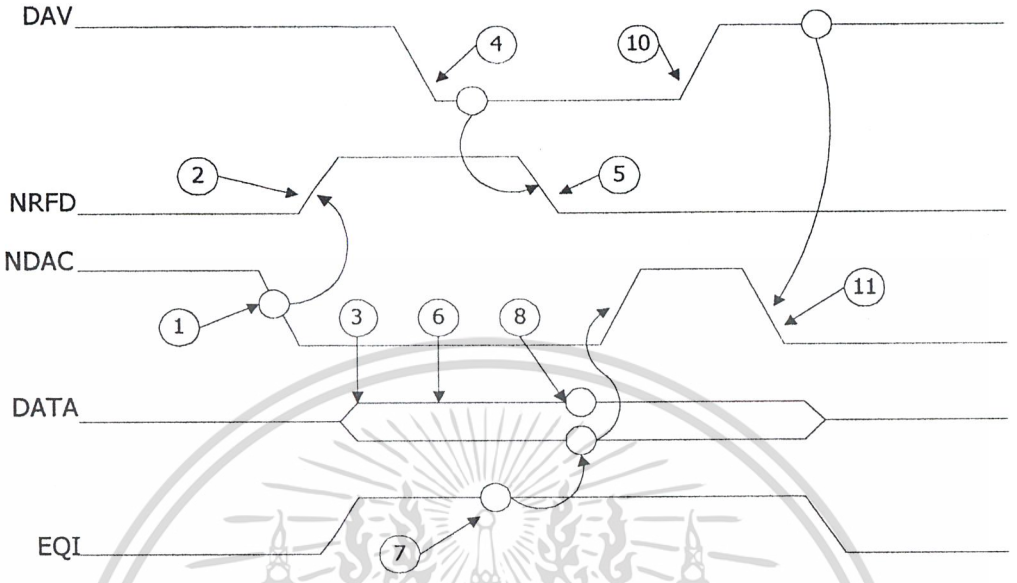


รูปที่ 2.8 Timming Diagram ของขบวนการแฮนด์เชคของตัวควบคุมซึ่งทำหน้าที่เป็นตัวส่ง

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

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

2.8 ขบวนการแฮนด์เช็คของตัวควบคุมซึ่งทำหน้าที่เป็นตัวรับ



รูปที่ 2.9 timing diagram ของขบวนการแฮนด์เช็คของตัวควบคุมซึ่งทำหน้าที่เป็นตัวรับ

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

ต่อมาตัวควบคุมจะทำการเซตค่าสัญญาณ NRFD ให้เป็น "Hi" เพื่อที่จะให้ตัวส่งทราบว่าขณะนี้ตัวควบคุมพร้อมที่จะรับข้อมูลแล้ว (ในจุดที่ 2) เมื่อสัญญาณ NRFD มีค่าเป็น "Hi" และสัญญาณ NDAC มีค่าเป็น "Low" แล้ว ตัวส่งจะทราบทันทีว่าทำการส่งข้อมูลลงมาในบัสข้อมูล ได้แล้ว (ในจุดที่ 3) หลังจากนั้นตัวควบคุมจะทำการรอเวลาให้ตัวส่งทำการส่งข้อมูลลงในบัสข้อมูลให้เสร็จ โดยจะทำการแฮนด์เชค หากไม่เกินกำหนดเวลาจะทำการตรวจสอบเวลาว่าเกินเวลาที่ต้องคอยแล้วหรือยังหากยังไม่เกินเราก็จะทำการแฮนด์เชคต่อไปจนกระทั่งหมดเวลาหรือจนกว่าสัญญาณ DAV เป็น "Low" หรือไม่หากไม่ก็จะตรวจสอบเวลาว่าเกินเวลาที่ต้องรอแล้วหรือยัง หากว่ายังไม่เกินเวลาก็จะทำการเชคต่อไปจนกระทั่งหมดเวลาหรือจนกว่าสัญญาณ DAV จะมีค่าเป็น "Low"

เมื่อสัญญาณ DAV เป็น "Low" แล้วตัวควบคุมจะตอบรับ โดยการส่งสัญญาณ NRFD มีค่าเป็น "Low" (ในจุดที่ 5) นั่นเป็นการบอกว่าตัวควบคุมพร้อมที่จะเริ่มเก็บข้อมูล (ในจุดที่ 6) หลังจากนั้นตัวควบคุมจะทำการตรวจสอบสัญญาณ EOI เพื่อตรวจสอบว่าข้อมูลที่ตัวส่งทำการส่งลงมา นั้น

หมดหรือยัง (ถ้าตัวส่งทำการส่งข้อมูลลงในบัสหมดแล้วจะทำการตั้งค่าสัญญาณ EOI ให้เป็น "Low") (ในจุดที่ 7) หากสัญญาณ EOI เป็น "Low" ตัวควบคุมจะเซตสถานะว่า ขณะนี้ตัวส่งได้ทำการส่งข้อมูลลงมาในบัสทั้งหมดแล้ว หากสัญญาณ EOI เป็น "Hi" ก็ไม่ทำการเซตสถานะ หลังจากนั้นตัวควบคุมจะทำการเป็นข้อมูลในบัส (ในจุดที่ 8) แล้วทำให้สัญญาณ NDAC มีค่าเป็น "Hi" เพื่อบอกให้ทราบว่าตัวควบคุมได้ทำการอ่านข้อมูลเสร็จเรียบร้อยแล้ว (ในจุดที่ 9) เมื่อตัวควบคุมทำให้สัญญาณ NDAC เป็น "Hi" แล้วตัวส่งจะลบข้อมูลในบัสข้อมูล โดยการทำให้สัญญาณ DAV เป็น "Hi" (ในจุดที่ 10) ซึ่งก่อนหน้านี้ตัวควบคุมจะทำการตรวจสอบอยู่แล้วว่าสัญญาณ DAV เป็น "Hi" หรือยัง

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

2.9 คำสั่งใช้งานของ GPIB

การสั่งการต่างๆเพื่อกำหนดหน้าที่การทำงานและกำหนดฟังก์ชั่น เช่น กำหนดช่วงการวัด โหมดการวัด หรือ อื่นๆ แก่เครื่องวัดที่ตกอยู่เหล่านี้นั้น ตัวควบคุมจะเป็นตัวกำหนดการส่งรหัสคำสั่งไปที่อุปกรณ์โดยผ่าน DI1-DI6 รหัสคำสั่งนี้จะถูกส่งไปในช่วงที่สายสัญญาณ ATN เป็น LOW

คำสั่งสำหรับกำหนดหน้าที่การงานต่างๆ ตามมาตรฐานของ GPIB มีอยู่ด้วยกัน 128 คำสั่ง ดัง แสดงในตารางที่ 2.1 ต่อไปนี้ โดยแบ่งเป็น 5 กลุ่มคำสั่ง

รหัสที่ใช้ในระบบ GPIB บัสนั้นใช้ร่วมกัน ทั้งรหัสข้อมูลและรหัสคำสั่ง นั่นคือรหัสเดียวกัน มีความหมายได้ 2 อย่าง คือ เมื่อ ATN เป็น LOW จะหมายถึงรหัสคำสั่ง แต่ ATN เป็น HIGH รหัสนี้จะแทนข้อมูลที่เป็น ASCII แทน ซึ่งในตารางได้แบ่งความหมายออกเป็น 2 คอลัมน์

ตารางที่ 2.1 คำสั่งการใช้งานของ GPIB

MSO	0		1		2		3		4		5		6		7	
LSO	ASCII	MSG	ASCII	MSG	ASCII	MSG	ASCII	MSG	ASCII	MSG	ASCII	MSG	ASCII	MSG	ASCII	MSG
0	NUL		OLE		SP	0	0	16	@	0	P	16	:		p	
1	SOH	GTL	OC1	LLO	!	1	1	17	A	1	Q	17	a		q	
2	STX		OC2		~	2	2	18	B	2	R	18	b		r	
3	ETX		OC3		#	3	3	19	C	3	S	19	c		s	
4	EOT	SDC	OC4	DCL	S	4	4	20	D	4	T	20	d		t	
5	ENO	PPC	NAK	PPU	%	5	5	21	E	5	U	21	e		u	
6	ACX		SYN		&	6	6	22	F	6	V	22	f		v	
7	BEL		ETB		,	7	7	23	G	7	W	23	g		w	
8	ZS	GET	CAN	SPE	(	8	8	24	H	8	X	24	h		x	
9	HT	TCT	EM	SPO	)	9	9	25	I	9	Y	25	I		y	
A	LF		SUB		"	10	:	26	J	10	Z	26	j		z	
B	VT		ESC		+	11	;	27	K	11	[	27	k		{	
C	FF		FS		'	12	<	28	L	12	\	28	l		.	
D	CR		GS		-	13	=	29	M	13	]	29	m		}	
E	SOH		RS		.	14	>	30	N	14	^	30	n		-	
F	SI		US		/	15	?	UNL	O	15	_	UNT	o		DEL	
	addressed command		universal command		listen address				talk address				secondary command			

Primary Command Group (PCG)

1. กลุ่มคำสั่งเจาะจงจุดหมาย (addressed command group) เป็นคำสั่งที่ส่งไปยังอุปกรณ์ที่เป็นตัวส่งหรือเป็นตัวรับที่กำหนดไว้ล่วงหน้าแล้ว คำสั่งนี้จะประกอบด้วย

GTL (got to local) สั่งให้อุปกรณ์กลับสู่สภาพการควบคุมปกติด้วยมือ

SDC (select device clear) สั่งให้อุปกร์เคลียร์ตัวเองสู่สภาพเริ่มต้นใหม่

PPC (paralled poll confiqure) เป็นคำสั่งสำหรับทำการจัดสายสัญญาณของการทำการจัดสรรสายสัญญาณของการทำกระบวนการตรวจสอบสภาพอุปกรณ์โดยวิธีขนานโดยใช้กับกลุ่มคำสั่งรอง

GET (group excute trigger) ใช้คำสั่งเริ่มต้นการทำงานของอุปกรณ์ทีละหลายตัว

TCT (take control) เป็นการกำหนดให้อุปกรณ์ตัวส่งทำหน้าที่เป็นตัวควบคุม

2. กลุ่มคำสั่งครอบคลุม (universaal command group) เป็นคำสั่งที่ส่งไปยังอุปกรณ์ทุกตัวที่ต่ออยู่ในบัสประกอบด้วย

LLO (local lockout) เป็นการสั่งให้อุปกรณ์ทุกตัวกลับไปสู่สภาวะเริ่มต้น

PPU (parallel poll uncinfiqure) ใช้ยกเลิกกระบวนการตรวจสอบสภาพแบบขนานทั้งหมด

SPE (serial pol enable) เปลี่ยนโหมดของการตรวจสอบสภาพเป็นอนุกรมในโหมดนี้จะเป็นการส่งสถานะของเครื่องแทนการส่งข้อมูล

SPD (serial poll disable) เปลี่ยนโหมดของการตรวจสอบแบบอนุกรม

3. กลุ่มคำสั่งกำหนดอุปกรณ์ตัวรับ (listener address group) เป็นคำสั่งสำหรับกำหนดให้อุปกรณ์เป็นตัวรับ ตารมรหัสหมายเลขจาก 0-30 และมีคั่ง UNT (untalker) สำหรับยกเลิก

4. กลุ่มคำสั่งกำหนดอุปกรณ์ตัวส่ง (talker address group) เป็นคำสั่งสำหรับกำหนดให้อุปกรณ์เป็นตัวส่ง ตามรหัสหมายเลขจาก 0-30 และมีคั่ง UNL (unlisten) สำหรับยกเลิกเช่นกัน

คำสั่งกลุ่มที่ 1 ถึง 4 นั้น จัดเป็นกลุ่มคำสั่งหลักที่มีความหมายตายตัวยังมีคำสั่งอีกกลุ่มที่ขึ้นอยู่กับการกำหนดภายหลังนั่นคือ กลุ่มคำสั่งรอง

5. กลุ่มคำสั่งรอง (secondary command group) เป็นคำสั่งที่กำหนดรายละเอียดย่อยของอุปกรณ์แต่ละตัวที่ต่ออยู่ในระบบให้มีการทำงานอย่างไร ตามจุดประสงค์ใช้งานของเครื่องมือนั้น เช่นเดียวกับการปรับปุ่มต่างๆ ด้วยมือนั่นเอง คำสั่งรองนี้จะตามหลังคำสั่งหลัก คือจะใช้หลังจากอุปกรณ์ต่างๆ ถูกกำหนดวางตัวในระบบเรียบร้อยแล้ว

คำสั่งต่างๆที่กล่าวไป ซึ่งใช้ในการกำหนดสภาวะการทำงานของอุปกรณ์ แต่ละสภาวะที่กำหนดไปนั้นเป็นอย่างไร และมีจุดประสงค์เพื่ออะไร ดังต่อไปนี้

Device clear / Interface Clear

Device clear ใช้ในการทำให้อุปกรณ์ที่ต่ออยู่ในบัสกลับไปสู่สภาวะเริ่มต้น ยังไม่มีการกำหนดฟังก์ชั่นใดๆ สภาวะเริ่มต้นนี้จะแตกต่างกันไปแล้วแต่ว่าอุปกรณ์นั้นออกแบบไว้อย่างไร Device clear มีอยู่ 2 ลักษณะ คือ เคลียร์หมดทุกตัวที่ต่ออยู่ (DCL) กับการเคลียร์เจาะจงอุปกรณ์ตัวใด ตัวหนึ่ง (SDC)

แต่ว่าในการเคลียร์อุปกรณ์ให้อยู่ในสภาวะเริ่มต้นนั้นไม่ได้หมายความว่าInterface function ของ GPIB จะถูกเคลียร์อุปกรณ์ให้ไปอยู่ในสภาวะเริ่มต้นด้วยแต่อย่างใด Interface function คือ สภาพการ interface ที่ได้กำหนดไว้ในระบบประกอบด้วยฟังก์ชั่นต่างๆ ดังแสดงในตารางที่ 2.2

ตารางที่ 2.2 Interface Function

ฟังก์ชั่น	สัญลักษณ์	การกลับสู่จุดเริ่มต้นโดย IFC
Source handshake	SH	ได้
Acceptor handshake	AH	ได้
Talker or enlarge talker	T or TE	ได้
Listener or enalarge litener	L or LT	ได้
Service request	SR	ไม่ได้
Remove / local	RL	ไม่ได้
Parallel poll	PP	ไม่ได้
Device clear	DC	ได้
Device trigger	DT	ได้
Controller	C	ได้

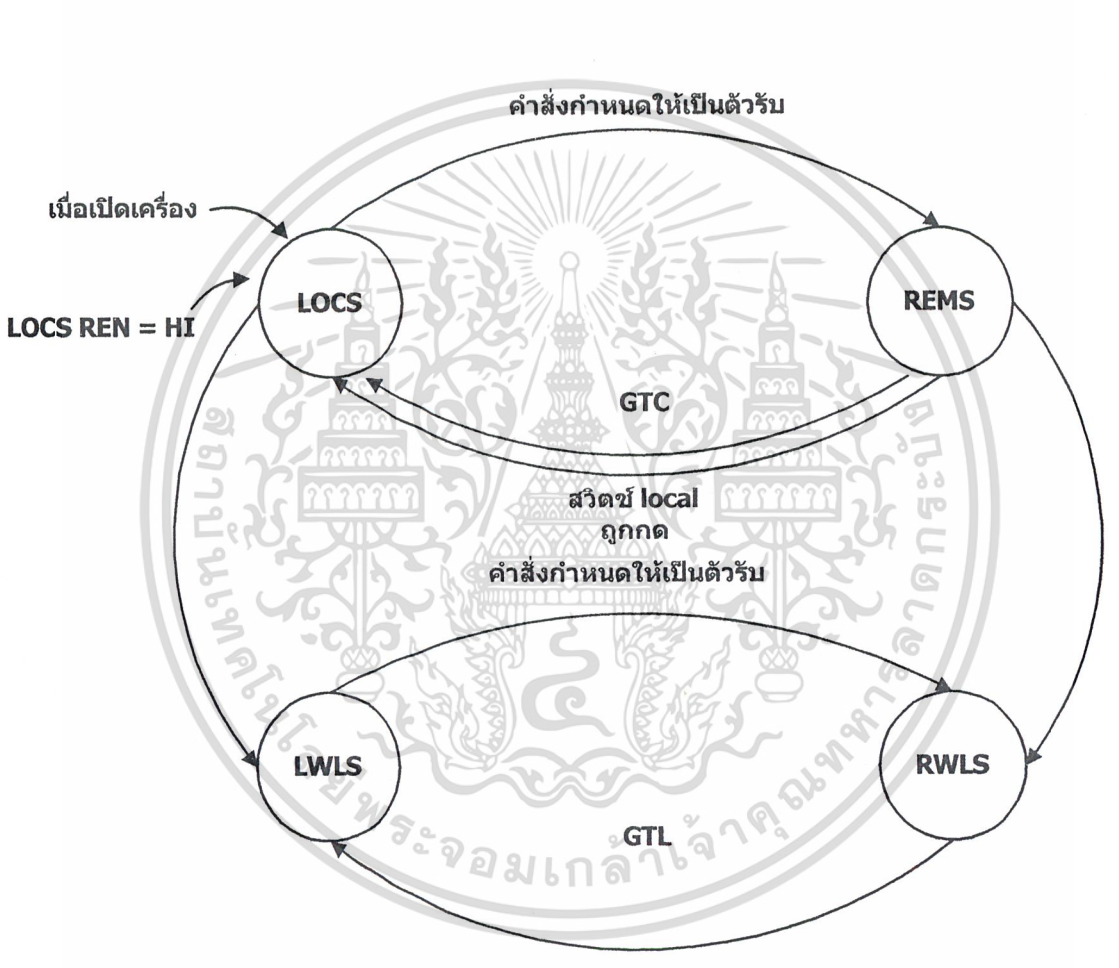
Remote / Local

Remote เป็นการกำหนดให้อุปกรณ์ที่อยู่ในระบบเช่น เครื่องมือวัดให้อยู่ในการควบคุมของอุปกรณ์ ตัวเชื่อมตัวอื่นแทน ซึ่งปุ่มปรับต่างๆ บนหน้าปัดเครื่องจะไม่มีผลต่อการทำงาน ส่วน Local เป็นการ ควบคุมการทำงานของเครื่องมือวัดด้วยปุ่มปรับบนหน้าปัดตามปกติ

การใช้งาน remote มีประโยชน์ในแง่ที่ขณะที่ตัวการควบคุมเช่น คอมพิวเตอร์กำลังติดต่ออุปกรณ์ ตัวนั้นอยู่ หากไม่มีการตัดการควบคุมโดยปุ่มปรับบนหน้าปัดออก ถ้ามีใครมาปรับแต่งก็จะทำให้อุปกรณ์ ตัวนั้นอยู่ หากไม่มีการตัดการควบคุมโดยปุ่มปรับบนหน้าปัดออก ถ้ามีใครมาปรับแต่งก็จะทำให้การ ทำงานผิดพลาดไปได้ การทำงานของ GPIB ใน remote and loacl มี 4 ลักษณะดังนี้

1. LOCS ก็คือ local นั่นเอง เป็นสภาพการควบคุมโดยปุ่มหน้าปัดตามปกติ จะอยู่ในสภาพนี้เปิดตอนเครื่องหรือ REN เป็น HIGH หรือเมื่อได้รับคำสั่ง GTL
2. REMS คือ remote หมายถึงการตัดการควบคุมโดยปุ่มหน้าปัดออก จะเกิดขึ้นเมื่อ REN เป็น LOW และจะถูกล็อกไว้เว้นแต่ว่าสวิตช์ local ที่ตัวอุปกรณ์จะถูกเปลี่ยนไปตำแหน่ง loacl

3. RWLS เป็นสภาพ remote ที่ถูกล็อกเอาไว้เช่นกัน แต่ว่าจะตัดการควบคุมตรงสวิตช์ local ที่ตัวอุปกรณ์ออกไป สภาพ remote โดย RWLS จึงมีความสำคัญสูงกว่า REMS อย่างไรก็ตามยังถูกยกเลิกได้ด้วยคำสั่ง LLO
4. LWLS มีสภาพเช่นเดียวกับ local แต่จะแตกต่างกันตรงที่สภาพ local โดย LWLS นี้ เมื่อได้รับคำสั่งกำหนดอุปกรณ์ตัวรับจะเปลี่ยนไปอยู่ในสภาพแบบล็อกหรือ RWLS ทันทีในการที่จะมาที่สภาพ LWLS นี้ได้ก็มี 2 กรณีคือ เมื่ออยู่ในสภาพ local ธรรมดา (LOCS) แล้วได้รับคำสั่ง LLO หรืออยู่ใน RWLS แล้วได้รับคำสั่ง GTL



รูปที่ 2.10 Loop การทำงานของ GPIB ใน Remote/Local

2.10 การขอบริการและการตรวจสอบ (Service Request and Polling)

เมื่อตัวควบคุมได้รับ SRQ เป็น LOW จะให้อุปกรณ์ส่งข้อมูลแสดงสถานะการทำงาน ซึ่งมีอยู่ 2 วิธีคือ

1. การตรวจสอบแบบอนุกรม ซึ่งมีขั้นตอนดังนี้

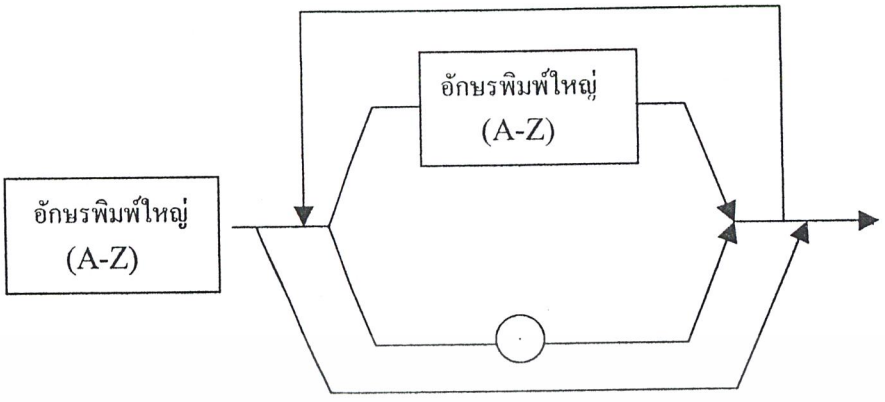
- ATN ถูกดึงเป็น LOW หลังจากได้รับ LOW จากสัญญาณ SRQ
- คำสั่ง UNL ถูกส่งไปยังอุปกรณ์
- ตัวควบคุมจะแจ้งรหัสตัวรับของตน และกำหนดรหัสตัวส่งอุปกรณ์ ที่จะตรวจสอบไป ที่บัส
- ตามด้วยคำสั่ง SPE และสาย ATN กลายเป็น HI ซึ่งอุปกรณ์ที่ถูกเรียกจะส่งข้อมูลแสดงสถานะออกมา 1 ไบต์ โดยบิตที่ 7 จะเป็นตัวชี้ว่าอุปกรณ์เส้นเป็นตัวขอบริการ ถ้าใช่จะเป็น LOW ส่วนบิตอื่นๆ ก็ใช้บอกข้อมูลอื่นๆ ซึ่งมิได้กำหนดเฉพาะ
- สาย ATN ถูกดึงเป็น LOW อีกที เพื่อส่งคำสั่งยกเลิกการตรวจสอบคือ SPD
- จากนั้นคำสั่ง UNT ก็ถูกส่งไปยังอุปกรณ์เพื่อยกเลิกการเป็นตัวส่ง ซึ่งถ้าหาก SQR ยังคงเป็น LOW อยู่ ก็จะมีการตรวจสอบไปยังอุปกรณ์ตัวอื่นๆ ต่อไปตามขั้นตอนเดิม

2. การตรวจสอบแบบขนาน สามารถทำได้เร็วกว่าแบบอนุกรมทั้งนี้เพราะสามารถอ่านข้อมูลเพียงไบต์เดียวก็สามารถรู้ได้ทันที ว่าอุปกรณ์ตัวใดเป็นผู้ขอบริการ

2.11 รูปแบบของข้อมูล

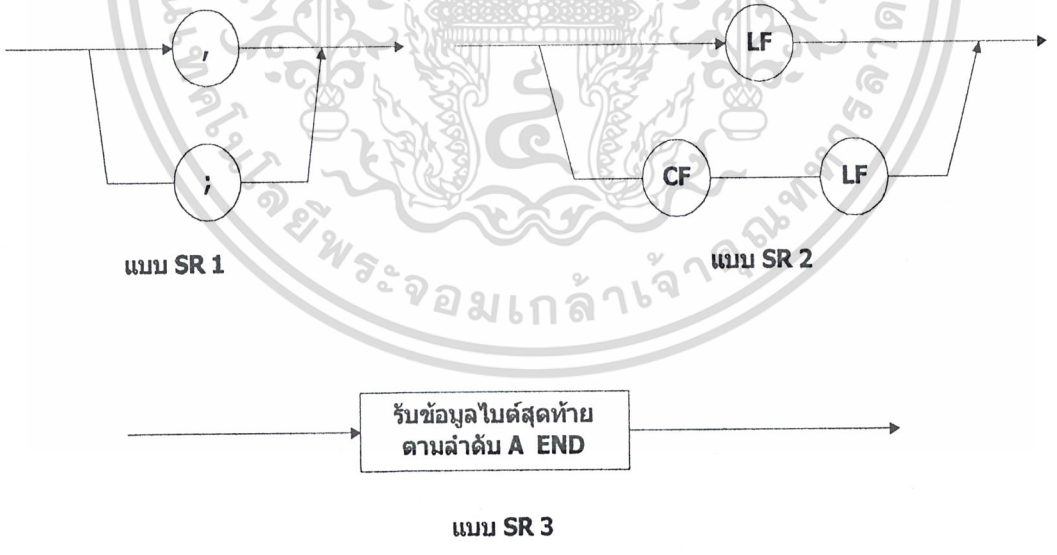
โดยทั่วไปข้อมูลจากอุปกรณ์ (device message) แบ่งออกได้เป็น 3 ส่วนดังแสดงในรูปที่ 2.11 อันได้แก่ส่วนหัว (HR) ซึ่งจะอยู่ส่วนหน้าสุด เป็นตัวบอกชนิดข้อมูล ส่วนประกอบ HR แสดงในรูปที่ 10 จะเห็นว่าประกอบด้วยตัวอักษรภาษาอังกฤษตัวพิมพ์ใหญ่ช่องว่างที่เป็นเว้นวรรค ( ) ปกติจะมี อักษรประมาณ 1-3 ตัว

รูปที่ 2.11 รูปแบบของข้อมูล

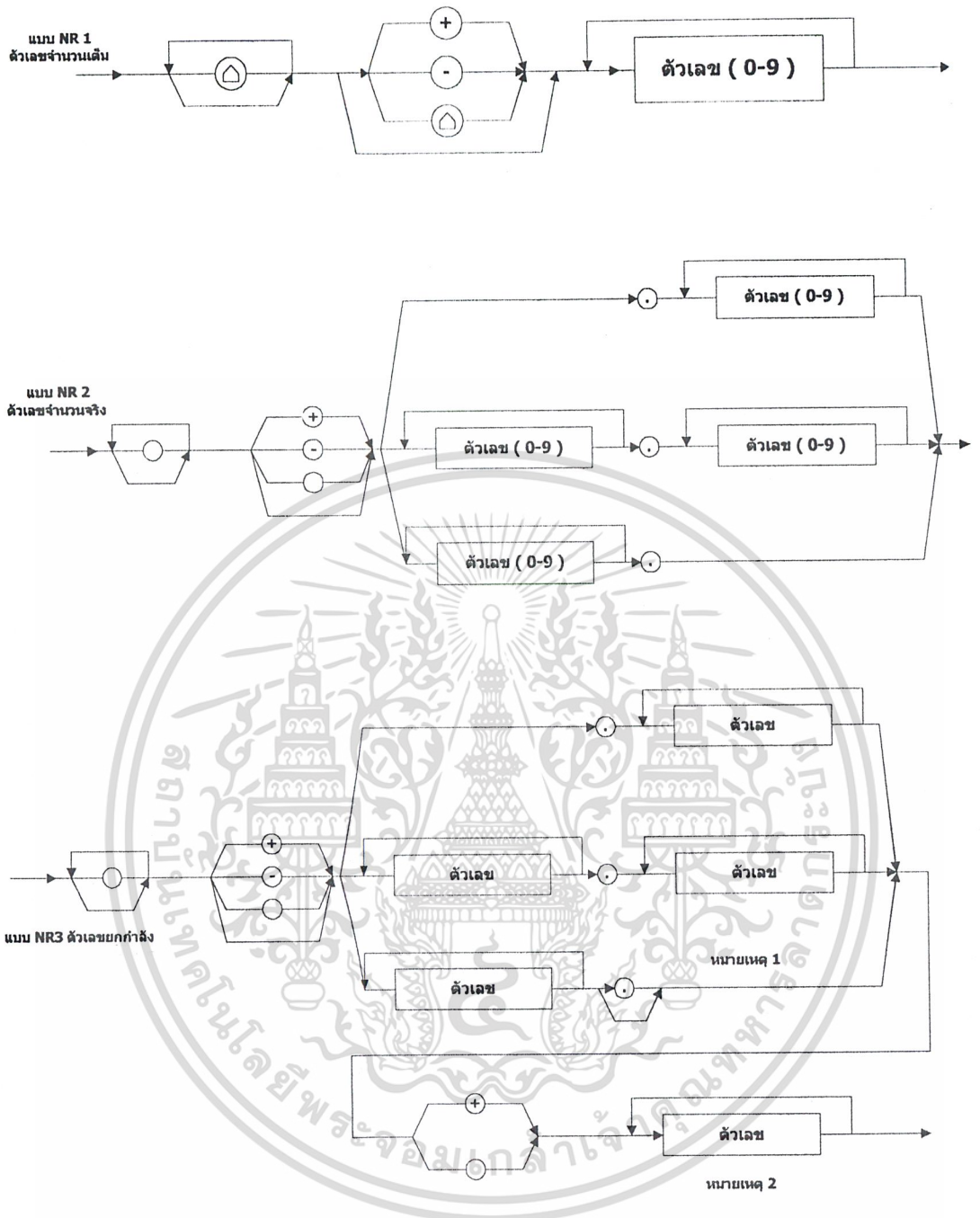


รูปที่ 2.12 รูปแบบชอง HR

ส่วนที่ 2 คือ เนื้อหาข้อมูล (NR) ซึ่งใช้แสดงค่าตัวเลข มีอยู่ 3 แบบ คือ NR1, NR2 และ NR3 ดังแสดงในรูปที่ 2.14 ส่วนท้ายของ NR ยังอาจมีตัวอักษรแสดงหน่วยตามมา



รูปที่ 2.13 สัญญาณแบ่งข้อมูลแต่ละชุด



รูปที่ 2.14 เนื้อหาข้อมูล

ส่วนที่ 3 คือ สัญญาณแบ่งข้อมูลแต่ละชุด (SR) ดังแสดงในรูปที่ 2.14 โดย SR1 ใช้แสดงการต่อเนื่องของข้อมูล (ข้อมูลยังมีต่อ) SR2 และ SR3 แสดงการสิ้นสุดของข้อมูล แต่ SR3 เป็นการบอกการเสร็จสิ้นข้อมูลทั้งจากการวัด

บทที่ 3

HP VEE

HP VEE คือโปรแกรมถูกพัฒนาขึ้นโดยบริษัท Hewlett-Packard ซึ่ง HP VEE ย่อมาจาก Hewlett-Packard Visual Engineering Environment ซึ่งก็คือโปรแกรมในรูปแบบของ ภาษาวิชวล โปรแกรมมิ่ง ซึ่งในความต้องการขั้นแรกของทางบริษัท Hewlett-Packard นั้นต้องการที่จะให้โปรแกรมช่วยให้ง่ายต่อการติดต่อสื่อสารเครื่องมือวัด และให้สะดวก ต่อการควบคุม เครื่องมือวัด

3.1 ทำไมเราต้องศึกษาเกี่ยวกับ HP VEE

- เพิ่มประสิทธิภาพ ในการพัฒนาการโปรแกรมมิ่ง เพื่อช่วยให้ง่ายต่อการควบคุมเครื่องมือวัด
- ช่วยให้เราประหยัดเวลาในการโปรแกรมมิ่ง
- โปรแกรม HP VEE ถูกพัฒนาขึ้นมาให้ง่ายต่อการใช้งาน โดยได้รวมเอาส่วนดี โปรมแกรมในรูปลักษณะ อักขระ และ โปรแกรม C, C++, Visual Basic, Pascal, Fortran, HP BASIC. มารวมกัน
- ใช้ HP VEE ได้ในหลากหลายรูปแบบ ซึ่งสามารถรันหรือใช้งานบน เครื่องคอมพิวเตอร์ โดยทั่วไป และยังใช้ได้กับหลากหลายระบบปฏิบัติการเช่น MS Windows, Windows 95, Windows NT
- ใช้ HP VEE ประยุกต์ในการใช้งานการเขียนโปรแกรมได้อย่างหลากหลาย ใช้สำหรับทดสอบ ใช้กับเครื่องมือวัด ใช้แสดงผล และใช้ควบคุม ในทางวิทยาศาสตร์ และ ทางด้านการศึกษา HP VEE ใช้ในการควบคุม GPIB, VXI, Serial, GPIO, PC Plug-in, และ เครื่องมือ LAN และกว่านั้นในการอินเตอร์เฟซ เราสามารถ อินเตอร์เฟซได้ในทางตรง โดยใช้ไดร์ฟเวอร์ ของเครื่องมือวัด

3.2 การใช้งาน HP VEE Development Environment โดยทั่วไป

HP VEE (Hewlett-Packard Visual Engineering Environment) เป็นโปรแกรมวิชวล ที่ถูกสร้างขึ้นเพื่อลดเวลาการทำงาน โปรแกรมนี้ถูกสร้าง โดยการเชื่อมต่อกับไอคอน เข้าด้วยกันบนหน้าจอมอนิเตอร์ การแสดงผลของโปรแกรม HP VEE นี้จะคล้ายคลึงกับบล๊อกไดอาแกรม หรือ อาจจะรันโปรแกรมในลักษณะคล้ายกับ C หรือ HP BASIC แบบฝึกหัดในบทนี้จะแสดงให้เห็นถึงลักษณะของการใช้งานของโปรแกรมนี้ ซึ่งไม่เพียงแต่ประสิทธิภาพของมันเท่านั้น แต่มันยังใช้ง่าย และยังเพลิดเพลินอีกด้วย

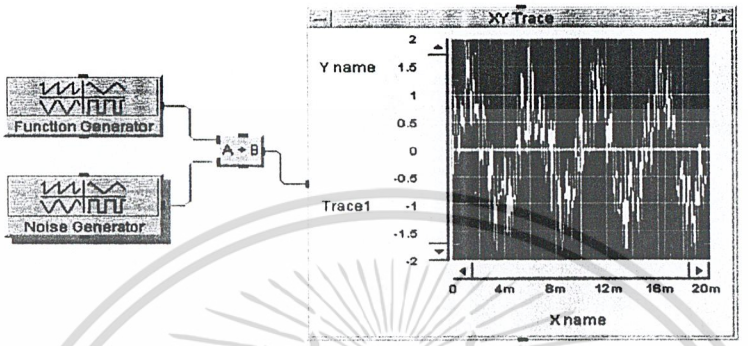
ในบทนี้เราจะเรียนรู้เกี่ยวกับ HP VEE ดังต่อไปนี้

- ส่วนตัวประกอบต่างๆ
- การเลือก Menu
- การ Save, Start.exit โปรแกรม HP VEE
- Help
- การใช้ Object
- Input & Output pin
- การเชื่อมต่อ Object เพื่อสร้างโปรแกรม
- การสร้าง Userobject

3.3 การโปรแกรมด้วย HP VEE

สามารถดูตัวอย่างโปรแกรม HP VEE ได้จาก รูป 3.1 ภาพนี้แสดงถึง รายละเอียดของโปรแกรมซึ่งติดต่อ ระหว่างไอคอนทั้งหมด ซึ่งคล้ายคลึงกับโค๊ดของภาษาโดยทั่วไป Object นี้มีหลาย Function ด้วยกัน เช่น I/O operation, การวิเคราะห์, และ การแสดงผล Object ทั้งหมดนี้จะมี Input & Output pin คงที่ไม่เปลี่ยนแปลง ข้อมูลของ Input pin อยู่ทางซ้าย ข้อมูล Output pin อยู่ทางขวา ลำดับของ pin คงไม่เปลี่ยนแปลง ข้อมูลของ Input ในแต่ละ Object จะแสดงเป็นรูปลักษณะของ Icon หรือในการเปิดรูปลักษณะ ดังรูปที่ 3.1 Object ที่ได้ทำการวิเคราะห์และตรวจสอบข้อมูลจะแสดงเป็น Icon และ Object ที่โปรแกรมตัวอย่าง Test 3 ข้อมูลและขอบเขตเป็น Object ที่เราสามารถแสดงผลเป็นภาพ วิวเปิดขนาดใหญ่ ซึ่งจะมีขนาดใหญ่และมีรายละเอียดมากกว่า ใน Object ก็จะประกอบด้วยโปรแกรมย่อยๆซึ่งสามารถแสดงให้เห็นหรือเปลี่ยนแปลงได้ เราสามารถเริ่มโปรแกรมนี้ได้โดยคลิกที่ Start Test 3. โดย Object นี้จะเริ่มทำการวัด Object ซึ่งจะเก็บรวบรวม ข้อมูลที่เราทดสอบไว้ และส่งไปยัง Object ถัดไปเพื่อวิเคราะห์ข้อมูล ไม่เพียงแต่วิเคราะห์ข้อมูลที่ต้องการทดสอบเท่านั้น แต่ยังสามารถนำข้อมูลความเข้าและออก หรือส่งข้อมูลและ ขอบเขตของการทดสอบไปยังที่แสดงผล และสามารถก๊อบปี้ และ บันทุกข้อมูลที่เราทดสอบในรูปของ ไฟล์ข้อมูลและในรูปของ ฐานข้อมูล เราจะเห็นว่า มันง่ายในการใช้งานโปรแกรมนี้ เพื่อใช่สร้าง ใช้วัด ใช้วิเคราะห์ทดสอบข้อมูลเป็น Object ที่เราสามารถสร้างและจัดเป็นพวกได้เอง HP VEE โปรแกรมนี้จะมีเอกสารที่เป็นมาตรฐานและใช้การได้จริง เราสามารถบันทึกข้อมูลของการโปรแกรมในไฟล์ซึ่งเราสามารถ เปิด เปลี่ยนแปลง และ รัน ใน HP VEE ได้ประโยชน์เพิ่มเติมของโปรแกรม HP VEE คือมันใช้เวลาสั้นๆเพียงไม่กี่นาที ในการสร้างระบบเชื่อมต่อ (ในตัวอย่าง รูป 3.1 ใช้เวลา 90 วินาที)

ในรูป 3.1 แสดงแผงควบคุมของโปรแกรม ในรูปที่ 3.1 เพียงแต่เริ่มที่ Object การแสดงผลก็จะปรากฏขึ้นเพื่อทำการทดสอบ ซึ่งง่ายต่อการใช้งาน (ซึ่งเหมือนกับเมนู และ ทูลบาร์ไอคอน ทั่วไป) เรายังสามารถป้องกันการเปลี่ยนแปลง ในสิ่งที่ไม่ต้องการได้

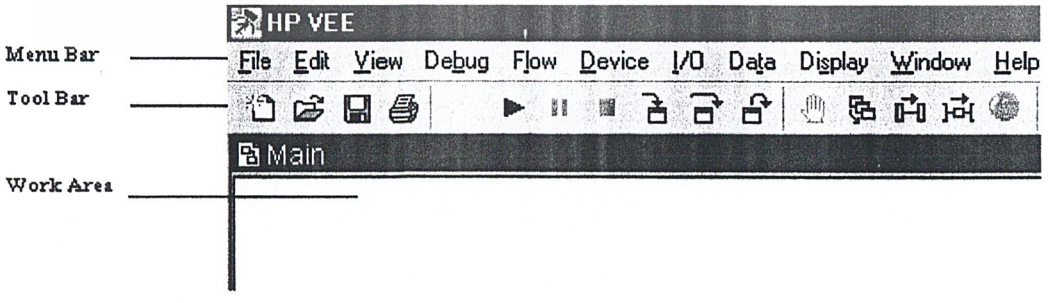


รูปที่ 3.1 การโปรแกรมมิ่งโดย HP VEE

ในบทที่จะสอนเกี่ยวกับการเริ่มต้น และ หยุดการใช้งาน HP VEE และการนำมาใช้ในการเรียนได้อย่างไร รวมทั้งทางกลไกของ Object เมื่อรับจากเมนูและติดต่อ Object เข้าด้วยกันเพื่อสร้างโปรแกรม ขั้นแรกให้ดูที่ส่วนประกอบพื้นฐานของ HP VEE

3.4 ส่วนประกอบของ HP VEE

หลังจากที่เราได้ทำการติดตั้งโปรแกรม HP VEE และเริ่มใช้งานโปรแกรม HP VEE คุณจะเห็นว่า HP VEE development environment จากรูปที่ 3.2 HP VEE จะมีแผงอยู่ด้านบนดังนั้น เราสามารถเลือกโปรแกรมจากส่วนประกอบหนึ่ง ไปยังอีกอันหนึ่งได้อย่างง่ายดาย



รูปที่ 3.2 หน้าต่างของ HP VEE

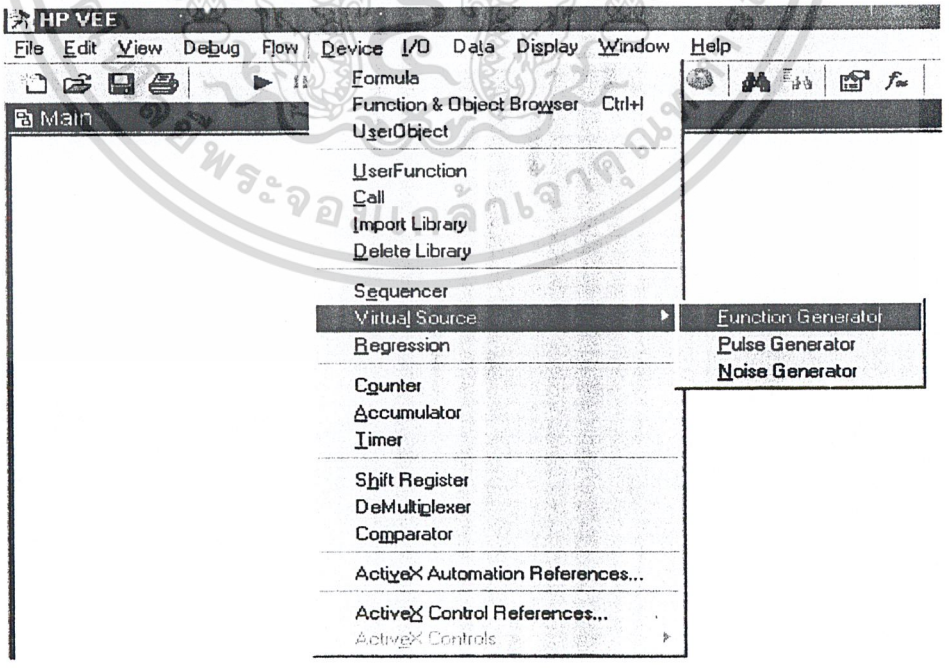
- เมนูบาร์ ประกอบด้วยเมนูคำสั่ง และ ไอคอน ที่จะสร้างโปรแกรม
- ปุ่มแสดงทูลบาร์ จะเป็นแป้นพิมพ์ลัด ที่มี การทำงานเหมือนทั่วๆไป (เพียงแต่เลื่อนเมาส์ ไปยังปุ่มของฟังก์ชั่น ต่างๆ)
- พื้นที่การทำงานของโปรแกรม เป็นที่ ที่เราสามารถสร้างโปรแกรมของเราได้ด้วยไอคอน

❑ **การใช้งานเมนู Menus**

มาดูตัวอย่างที่จะอธิบายเกี่ยวกับการทำงานของเมนู การทำงานจะเหมือนกันทั้งหมด ขั้นแรก เราจะต้องเปิดเมนู และเลือกที่ Object จากเมนูย่อย และนำมาวางที่พื้นที่การทำงานของโปรแกรม หลังจากนั้น เราก็ทำการเปิด 'Object Menu' และเลือกวิธีใช้ (Help) เพื่อหาว่าเราจะประสาน Object อย่างไร (HP VEE object จะมีลักษณะเน้นเป็นสีสันที่เด่นชัด ถ้าเพียงคลิกหรือเคลื่อนเมาส์ไปที่มัน)

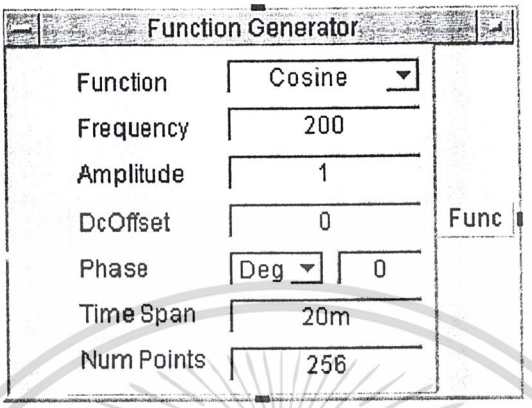
ข้อควรจำคลิกแล้วกดค้าง หมายความว่า เราควรจะกดคลิกซ้ายและ ลากลงมาจนถึงที่ ที่ต้องการคลิก หมายถึง กดคลิกซ้ายอย่างรวดเร็ว ถ้าเราต้องการใช้คลิกขวา เราจะได้ คำแนะนำขึ้นมา ถ้าเรามีเมาส์ ที่มี 3 ปุ่ม เราไม่จำเป็นต้องใช้ปุ่มกลางใน HP VEE

1. คลิกแล้ว กดค้างที่ Device เพื่อที่จะเปิดเมนู โดยเมนูจะเลื่อนลงมาให้เลือก
2. เลื่อนเมาส์ ไปยังเมนูย่อย (Virtual Source) และเลื่อนไปทางขวาจะพบ Function Generator และปล่อยปุ่มเมาส์ตาราง Function Generator จะปรากฏขึ้น



รูปที่ 3.3 คลิก Device => Virtual Source จะได้ผลดังรูป

3. เลื่อน Function Generator ไปยังตรงกลางของพื้นที่ของงาน และ คลิกเพื่อวาง Object โดย Object จะแสดงให้เห็นเมื่อเราวางมัน เหมือนในรูป 3.4



รูปที่ 3.4 Function Generator Object

เลือก Device -> Virtual Source => Function Generator, และวางในส่วนกลางของพื้นที่ขอโปรแกรมนั้น

4. เมื่อเปิด Function Generator ของ Object เมนูโดยการคลิกเมาส์ตามแนวนอนของบาร์ บนมุมซ้ายมือของ Object เราจะเห็นว่า Object เมนูบาร์ และการเปิดเมนู ซึ่ง Object ของเมนู จะเปิดแบบนี้เหมือนกันหมด

เพื่อความสะดวกที่จะเปิด Object เมนูเราใช้เมาส์ชี้ไปที่ Object และคลิก ขวาและใช้ Start Object เมนูจะขึ้นมาที่จุดที่เมาส์ชี้ Function Generator โดย Object จะเลียนแบบการทดสอบข้อมูล ซึ่งเราสามารถใช้มันเพื่อสร้างลูกสัญญาณ เช่น sine, cosine, square, triangle, +ramp,

- ramp, ซึ่งดีพอๆกับการใช้ CD ถ้าเราต้องการที่จะรู้มากกว่านี้เพียงแต่เราเปิดไปที่เมนู เลือกวิธีใช้

❑ การใช้งาน Object

การลบ Object จากพื้นที่การทำงานของโปรแกรม ให้เราใช้ตัวอย่าง Object โดยใช้ Function Generator จากเมนู Device แล้วนำมาใส่ในพื้นที่การทำงานของโปรแกรม
เลื่อนเมาส์ไปยัง Object แล้วกด Ctrl-d หรือเปิดเมนู Object แล้วเลือก cut

❑ การวางและการลบ Object

หลังจาก Object ได้ถูกลบเราสามารถนำกลับมา (undo) ได้ โดยให้คลิก Edit -> Paste หน้าจอ Object ก็จะปรากฏขึ้นมาให้วาง Object แล้วคลิกเพื่อลบออกถ้า Object มีเส้นอยู่ด้วย เส้น เส้นนั้นก็จะยังคงอยู่ การทำในรูปแบบนี้เหมือนกับการ Undo ในโปรแกรมอื่นๆ แต่ที่ไม่มีการเรียกว่า Undo

เพราะมันไม่สามารถประยุกต์ใช้ในโปรแกรม HP VEE ทั้งหมด (มันสามารถทำงานเพื่อ Object เป็นกลุ่มๆ นั้นได้ถูกลบออกไปแล้ว)

- **การจำลองสำเนา หรือ การจำลอง Object**

การกระทำซ้ำของ Object จะต้องเปลี่ยนบางสิ่งบางอย่างเช่นขนาดหรือเปลี่ยนชื่อทำได้โดยการเปิดเมนู Object แล้วเลือก Clone หน้าจอ Duplicate Object จะปรากฏขึ้น แล้วจากนั้นทำการเลื่อนเมาส์ ไปยังที่ ที่ต้องการและคลิกวาง Object นั้น

- **การเคลื่อนย้าย Object**

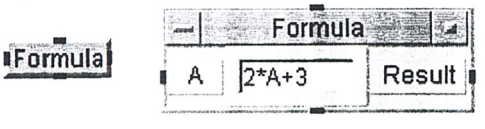
การเคลื่อนย้าย Object เลือก File -> Now เพื่อเริ่ม File ใหม่และทำการล้างพื้นที่ของโปรแกรมก่อนที่จะเริ่มต้นทำการโปรแกรมงานอันใหม่ ถ้าจำเป็น ถ้างานทำงานแล้วยังต้องการให้มีอยู่ ดูตัวอย่างการแสดงผลของ Object จากตัวอย่าง เลือก Display -> Waveform(time) แล้ววางไว้ทางขวาของพื้นที่ของโปรแกรม จากนั้นเปิดเมนู Object แล้วเลือก Move เมาส์จะเปลี่ยนเป็นสี่เหลี่ยมขนาดเล็กที่เป็นกากบาท เลื่อนเมาส์ไปที่ Object แล้วกดค้างที่ คลิกซ้าย (หรือที่เรียกว่าลากเมาส์) ขณะที่เราเคลื่อนย้าย Object เพื่อนำมาวางในที่ ที่เราต้องการ ปล่อยคลิกที่กดค้างไว้เพื่อวาง Object

- **การแก้ไขและเปลี่ยนแปลงชื่อของ Object**

เราสามารถแก้ไขเปลี่ยนชื่อของ Object ได้โดยเปิด Object เมนูแล้วเลือก Edit Properties หน้าจอ Properties จะปรากฏขึ้นและจะมีข้อความหัวเรื่องขึ้นมาด้วยพิมพ์ข้อความแล้วคลิกOK

- **การปิดเปิด Object ระหว่างไอคอน และ ลักษณะ Open views**

การเปิด Object ไอคอนและลักษณะ Open view โดย HP VEE จะแสดงผล Object ในรูปของภาพไอคอน หรือเปิดวิว ดังในรูป 3.5 ภาพไอคอนจะปรากฏในที่ว่างในพื้นที่ของการโปรแกรมและจะทำให้โปรแกรมของเราสามารถอ่านได้ง่าย Open view จะมีรายละเอียดมากและสามารถแก้ไข ภายใน Object ได้ ในการเปิดภาพไอคอน ต้องคลิกที่จุดทางขวาบนของหัวเรื่อง ของแถบ Object ถ้าจะกลับไปยัง Open view ก็คลิกสองครั้งที่ไอคอน โดยเมนูหรือ Object สามารถย่อหรือ ขยายเป็นเท่าเดิมได้



รูปที่ 3.5 ไอคอน และโอเพนวิว

- **การปรับขนาดของ Object**

เลื่อนเมาส์ไปที่ด้านมุมล่างซ้ายของ Object จนคุณจะเห็นสัญลักษณ์เป็นรูป มุมซ้ายแล้วคลิก แล้วจากนั้นลากไปยังขนาดที่ต้องการ หรือ เปิดเมนู Object แล้วคลิก size เราจะเห็นมุมทางด้านซ้ายบนของ Cursor เลื่อนจากมุมด้านซ้ายไปยังขนาดที่ต้องการของมุมทางขวาล่างแล้ว คลิก เราสามารถเลือกขนาดของ Object เมื่อเราเริ่มเลือกมันจากเมนู หลังจากนั้นเราวาง Object เพียงแต่คลิกแล้วลากไปยังจุดที่ต้องการของมุมขวาล่างของ Object

- **การเลือกใช้ (Select) และ การไม่ใช้ (Deselect) Object**

Object จะถูกเลือก (Select) เมื่อเราคลิกมัน HP VEE ทำให้เกิดเงาเกิดขึ้นหลังที่เราได้เลือก Object ถ้าจะไม่เลือกใช้ (Deselect) ก็เพียงแต่เลื่อนเมาส์ไปยังพื้นที่ของโปรแกรมแล้วคลิก ซึ่งการใช้คำว่า select แทน choosing item ก็มีความหมายชัดเจนอยู่แล้วว่า เมื่อเราเลือก Object เพื่อให้ HP VEE ทำบางอย่างกับ Object นั้น เช่น ถ้าเราเลือก Object แล้ว คลิก Edit -> Cut จะทำให้ Object นั้นหายไป

- **การเลือกใช้งานหลายๆ Object**

เมื่อเราต้องการทำ Object หลายๆ Object เช่นเลือก Edit -> Cut or Edit -> Move Object หรือทำได้โดยเลือก Edit -> เลือก Object Cursor จะกลายเป็นลักษณะมุมขวาหลังจากนั้นคลิกแล้วลากรูปสี่เหลี่ยมผืนผ้าจะขยายออกตามแต่ขนาดที่ต้องการ โดยถ้า Object ไม่สามารถเป็นรูปสี่เหลี่ยมผืนผ้า เราสามารถเลือก Object ได้หลายๆแบบโดยกด Ctrl แล้วคลิกเลือกที่ Object ที่ต้องการ โดยสังเกตว่าแป้นพิมพ์ลัด (Shortcut) จะอยู่ในเมนู 'LB' ซึ่งมันก็คือคลิกซ้ายดังนั้น Ctrl-LB ก็คือแป้นพิมพ์ลัดเพื่อเลือก Object Menu Item

- **การแก้ไข Object**

คลิก Edit ที่แถบเมนูแล้วเลือกการทำงานที่เราต้องการ จากนั้นเลื่อนเมาส์ไปที่ใดที่หนึ่งบนพื้นที่ของโปรแกรม แล้วคลิกขวา หน้าจอ Edit Menu จะปรากฏขึ้นมา โดยในแต่ละ Menu ที่ปรากฏขึ้นมาในลักษณะสีที่ต่างกันเช่น cut, copy, clone ในเมนู Edit จะปรากฏขึ้นต่างสีกันกับเมนูทั่วไป จนกระทั่ง Object จะถูกทำให้ดูเด่นในพื้นที่ของโปรแกรม

- **การสร้าง Data Line ระหว่าง Object**

คลิกที่ Data Output Pin ของ Object แล้วคลิก Data Input Pin (Line จะปรากฏขึ้นหลังจาก Pin หนึ่งไปยังอีก Pin หนึ่ง)

- **การเคลื่อนย้ายพื้นที่ของการโปรแกรม (Work Area)**

ก่อนอื่นต้องแน่ใจว่ามีอย่างน้อย 1 ไอคอนในพื้นที่ของโปรแกรม จากนั้นเราเลื่อนเมาส์ไปที่ใดก็ได้ทางด้านหลังของพื้นที่ของโปรแกรม แล้วกดค้างที่คลิกซ้าย และ เราเลื่อนไปยังพื้นที่ของการโปรแกรม ตรงไหนก็ได้ โดย Scroll Bar จะปรากฏขึ้นถ้าโปรแกรมของเรานั้น มีขนาดใหญ่เกินพื้นที่ของการโปรแกรม

- **การเชื่อมต่อระหว่าง Object โดย Pin และ จุดต่อ (Terminals)**

เราสามารถสร้างโปรแกรม HP VEE โดยการเชื่อมต่อ Object เข้าด้วยกันกับ Data Line ซึ่ง Line จะเชื่อมต่ออยู่กับส่วนย่อย (Pin)ใน Object เราจะติดต่อ ข้อมูลทางด้านอินพุท และ ข้อมูลทางด้าน เอาท์พุท, Pin มีไว้เพื่อรับส่งข้อมูลระหว่าง Object ต่อ Object ในลำดับ Pin ที่เชื่อมกันอยู่ เราสามารถเลือกได้ถ้าเชื่อมต่อกันแล้ว Pin จะส่งงานผ่านไปยังด้านบนถึงด้านล่างของ พื้นที่ของโปรแกรม ซึ่งจุดต่อ จะง่ายต่อการแสดงผล Open view ของข้อมูลย่อย (Data Pin) มันจะรับส่งรายละเอียดข้อมูล เช่น ชื่อของจุดต่อ ชนิดและค่าของข้อมูล (การแสดงผลของข้อมูลทำได้โดย คลิกสองครั้งที่ Terminal)

- **การเพิ่มข้อมูลทางด้านอินพุทไปยัง Object**

เปิดเมนู Object เลือก Add Terminal => Data Input ถ้า Data Input มีฟังก์ชั่นจำกัด เหมือนกับเครื่องมือไดร์เวอร์ เช่นเราจะได้เมนูของฟังก์ชั่นนี้มาจุดต่อ จะมีชื่อเป็น A, B, C, ... โดยแป้นพิมพ์ลัดทำได้โดยเลื่อนเมาส์ไปยังจุดต่อตรงพื้นที่อินพุท แล้วกด Ctrl-a

- **การลบข้อมูลทางด้านอินพุท หรือ เอาท์พุทที่ถูกเชื่อมต่อ**

เปิดเมนู Object แล้วเลือก Delete Terminal => Input เลือก Input เพื่อ Delete แล้วคลิก OK โดยแป้นพิมพ์ลัดทำได้โดยเลื่อน เมาส์ไปที่จุดต่อแล้วกด Ctrl-d ซึ่งแป้นพิมพ์ Ctrl-d จะไวถ้ามันอยู่ที่พื้นที่จุดต่อ มันจะทำการลบจุดต่อ ถ้ามันอยู่ที่ Object มันจะลบ Object นั้นออกไป

- **การเชื่อมต่อ Object เพื่อสร้างโปรแกรมขึ้น**

ต่อไปนี้จะเป็นตัวอย่างง่ายๆ หลักการทำงานโดยทั่วไปจะเหมือนกันหมดทุกๆโปรแกรม โดยโปรแกรมนี้จะง่ายต่อการสร้างและแสดงผลโดยการสุ่มตัวเลขซึ่งจะให้เราฝึกในการเชื่อมต่อ

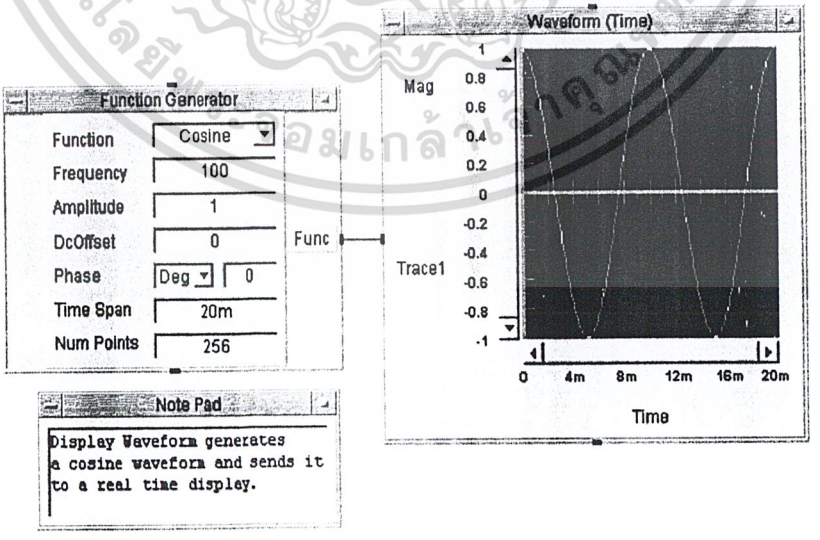
Object และ เอกสาร ซึ่งตัวอย่างนี้ทำขึ้นเพื่อสอนเกี่ยวกับหลักการของโปรแกรม แต่จะไม่มีเกี่ยวกันทางด้านปฏิบัติ

3.5 ตัวอย่างการโปรแกรม ที่ 1

โปรแกรมนี้จะให้คุณสมบัติมากขึ้นในการสร้างโปรแกรม HP VEE เราจะสร้าง ลูกคลื่น Cosine และแสดงผลมัน

ขั้นตอนการโปรแกรม

1. ขั้นแรกให้เอกสาร โปรแกรมที่เลือก Display -> Notepad และวางมันที่ทางด้านกลางส่วนบนของพื้นที่ของโปรแกรม คลิกที่พื้นที่สำหรับเขียนข้อความควรจำเอาไว้ Cursor และ Enter การแสดงผลจะมีลูกคลื่น Cosine แสดงขึ้นและส่งมันไปยัง Real Time Display เราอาจจะเปลี่ยนแปลงขนาดที่ Notepad นั้น ขึ้นอยู่กับหน้าจอของเราเอง (เปิดเมนู แล้วเลือก Size เลื่อน Cursor ให้มีรูปล่างเป็นมุมขวาที่มุมล่างของทางขวาของ Object แล้วคลิก)
2. เลือก Device -> Virtual Source -> Function Generator และวางไว้ที่ทางด้านซ้ายของพื้นที่ของโปรแกรมและ Edit Frequency ถึง 100
3. เลือก Display -> Waveform(time) และวางไว้ทางด้านขวาของพื้นที่ของโปรแกรม
4. คลิกที่ Function Generator ข้อมูลทางด้านรอยต่อเอาท์พุท จากนั้นเลื่อนเมาส์ไปยังข้อมูลทางด้านรอยต่ออินพุทของลูกคลื่น Waveform(time) Object แล้วคลิกอีกครั้ง
5. คลิกรันที่ทูลบาร์ (ข้างใต้เมนู Debug) โปรแกรมของเราจะเหมือนในรูปที่ 3.4



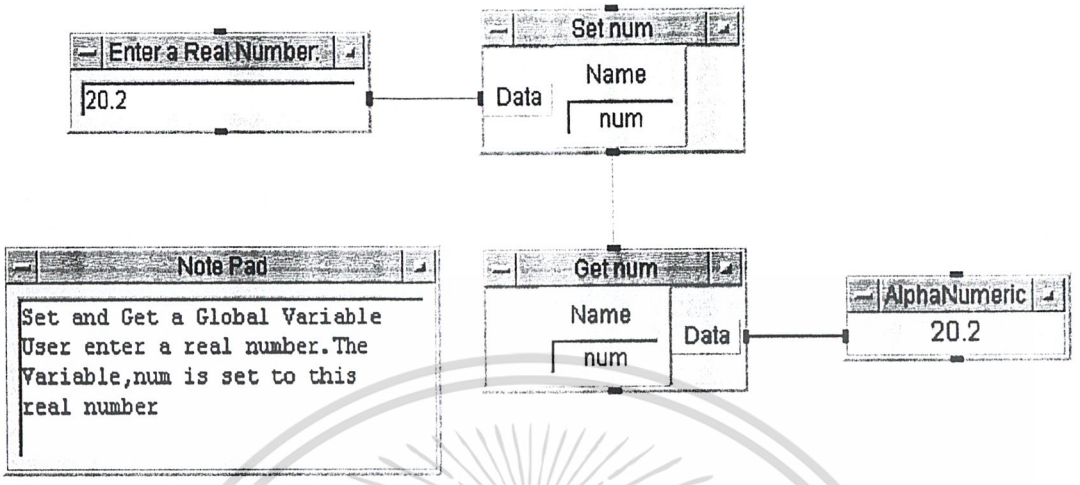
รูปที่ 3.6 โปรแกรมแสดงผลลูกคลื่น Cosine

3.6 ตัวอย่างการโปรแกรม ที่ 2

โปรแกรมนี้จะให้เราฝึกเพิ่มขึ้นเกี่ยวกับพื้นฐานทางกลไกของการสร้าง HP VEE และแนะนำให้เข้าใจง่ายขึ้นเกี่ยวกับตัวแปร ซึ่งเราสามารถตั้ง Object ที่เข้าใจง่ายได้เพื่อสร้างตัวแปรที่สามารถเรียกคืนได้ ตัวอย่างจะใช้ตัวเลขเป็นจำนวนจริง

1. เลือก Display -> Notepad และวางไว้ที่ด้านบนของส่วนกลางของพื้นที่ของโปรแกรม แล้วคลิกที่มุมซ้ายบนของ Edit Area เพื่อให้มี Cursor แล้ว Enter เพื่อตั้งค่าแล้วนำตัวแปรมา Enter จำนวนจริงตัวแปรมา Enter จำนวนจริงตัวแปร num จะตั้งให้เป็นจำนวนจริงดังนั้น num นั้นจะถูกเรียกอีกครั้งแล้วจึงจะแสดงผล
2. เลือก Data -> Constant -> Real -> แล้ววางไปทางซ้ายของพื้นที่โปรแกรม เปิดเมนู Object แล้วตรวจสอบที่วิธีใช้ (Help)
3. เปิด Real Object แล้วเลือก Edit Properties เปลี่ยนหัวเรื่องที่พร๊อมพ์ (Prompt) แล้ว Enter ที่ Real แล้วจากนั้นจึงคลิกที่ OK โดยเราจะใช้ Object ค่าคงที่ เพื่อให้ข้อมูลทางด้านอินพุท จะได้เปลี่ยน หัวเรื่องที่พร๊อมพ์ (Prompt) ได้ง่าย นี่เป็นเทคนิคทั่วๆไป ในการใช้งานอินพุท เราสามารถใช้ Data -> Dialog Box -> Real input ยังสามารถคลิกสองครั้งที่แถบหัวเรื่องเพื่อที่จะทำการแก้ไข (Edit Properties Dialog Box)
4. เลือก Data -> Variable -> Set Variable แล้ววางไปทางขวาของ Real Object จากนั้นคลิกสองครั้งที่ Global A จะมีลักษณะเน้นเป็นสีสันที่เด่นชัด เกิดขึ้นมาให้เรา Enter num ซึ่งมันหมายความว่าจะ Enter จำนวนจริงใน Real Object เมื่อคลิกที่ปุ่ม Run จำนวนดังกล่าวนั้นจะถูกตั้งค่าโดย Global Variable num
5. การเชื่อมต่อข้อมูลรอยต่อทางด้านอินพุท ของ Real Object ไปยังข้อมูลรอยต่อทางด้านอินพุท ของการตั้งค่า Global
6. เลือก Data -> Variable -> Get Variable แล้ววางไว้ข้างใต้ของ Set Variable Object แล้วเปลี่ยนตัวแปรเป็น num
7. เชื่อมต่อ Set Variable ข้อมูลรอยต่อทางด้านอินพุท ไปยังลำดับขั้นของ Get Variable โดยตัวแปรอย่างง่ายนี้เรา สามารถตั้งค่าได้ก่อนที่เราจะใช้มัน เราจึงใช้ Pin ซึ่งเป็นไปตามลำดับเพื่อให้แน่ใจว่าตัวแปร num นั้นได้ถูกตั้งค่าแล้ว ก่อนที่เราจะแก้ไขมันด้วย Get Global
8. เลือก Display -> AlphaNumeric และวางทับไว้ทางด้านขวาของ Get Variable ของ Object นั้น
9. เชื่อมต่อ Get Varible ข้อมูลรอยต่อทางด้านเอาท์พุท ไปยัง AlphaNumeric ข้อมูลรอยต่อทางด้านอินพุท

10. Enter จำนวนจริงแล้วคลิกปุ่ม Run ที่บูลบาร์



รูปที่ 3.7 ผลตัวอย่างการโปรแกรมมิ่งที่ 2

- **Conserving Screen Space**

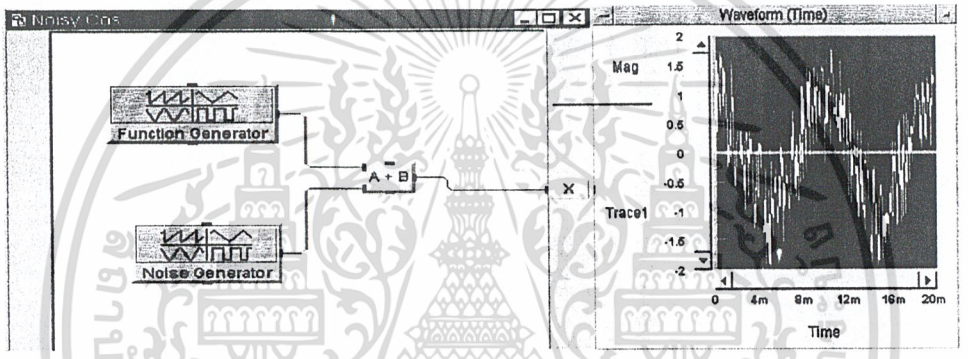
ที่เป็น Object ในเมนู Device เรียกว่า Userobject ซึ่งจำเป็นสำหรับ HP VEE environmeny ในพื้นที่โปรแกรม หลักมันเป็นส่วนย่อย ที่เราตั้งขึ้นมาเหมือน Object อื่นๆใน HP VEE ถ้าเรานำส่วนโปรแกรมของเราใส่ลงไปใน Userobject มันจะเป็นลักษณะหน้าจอเปล่า เราสามารถนำ UserObject มาอธิบายหน้าที่ของมันได้

3.7 ตัวอย่างการโปรแกรม ที่ 3

ตัวอย่างโปรแกรมนี้จะชี้ให้เห็นว่าการใช้งานเกี่ยวกับ UserObject เราสามารถเพิ่มลูกคลื่น Cosine ไปยังลูกคลื่น noisy ใน UserObject เรียกว่าNoisy Cos การแสดงผลจะปรากฏในรูปของคลื่น

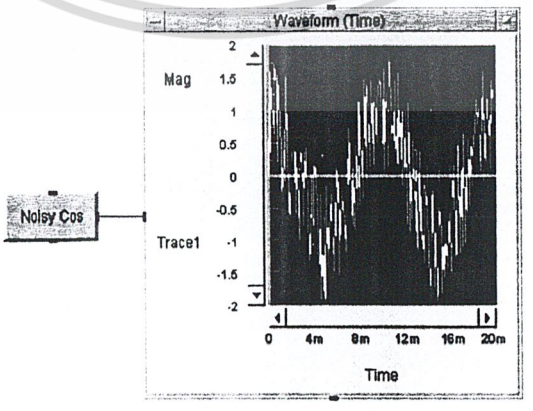
1. เลือก Device -> Userobject แล้ววางทางด้านซ้ายของพื้นที่โปรแกรม ถ้าจะเปลี่ยนหัวเรื่องก็เปิดเมนู UserObject อีกครั้งแล้วเลือก Add Terminal -> Data Output UserObject ตอนนี้มีชื่อเรียกว่า Noisy Cos และมันจะมีข้อมูลรอยต่อทางด้านเอาท์พุท กับ ทางเลือก Label X
2. เลือก Device -> Virtual Source -> Function Generator วางลงใน Noisy Cos. มีค่า 1000 ในความถี่ทางด้านอินพุท แล้วพิมพ์ 100 แทน ไอคอน Function Generator โดยการคลิกที่ปุ่ม Icon ทางด้านมุมบนขวาของ Object
3. เลือก Device -> Virtual Source -> Noise Generator แล้ววาง ลงใน Function Genenrator ในUserobjectแล้วเปิดวิธีใช้(Help)เพื่อหาว่ามันเป็นอะไรและลักษณะของ Noise Generator

4. เลือก Math -> + - * / -> a+b แล้ววางลงทางขวาของ Generator ใน Userobject ใช้วิธีใช้ (Help) เพื่อดู + เชื่อมต่อข้อมูลรอยต่อทางด้านเอาท์พุทของ Generator ทั้งสอง ที่ รอยต่อขาเข้า บน Object + HP VEE ตอนนี้ก็จะเพิ่มลูกคลื่น 2 อันเราต้องการส่งผลของเคลื่อนไปยัง Userobject ข้อมูลรอยต่อทางด้านเอาท์พุท
5. เชื่อมต่อข้อมูลรอยต่อทางด้านเอาท์พุท ของ Object a+b ไปยัง Userobject ข้อมูลทางด้านเอาท์พุท X ตำแหน่งของไอคอน จะอยู่ทางด้านบนซ้ายของ Noisy Cos
6. เลือก Display -> waveform(time) และวางไว้ทางขวาของ Noisy Cos Userobject เชื่อมต่อ Noisy Cos ข้อมูลทางด้านขาออกเอาท์พุท ไปยัง waveform(time) ข้อมูลทางด้านขาเข้าอินพุท เปิดเมนู Object waveform(time) แล้วเลือก Edit Properties ด้านล่างของ Layout เลือก Graph only แล้วคลิก OK



รูปที่ 3.8 โปรแกรมรวม Noisy และ Cosine เข้าด้วยกัน

7. คลิกที่รันแล้วจะแสดงผลให้เห็นตามรูปข้างบน
8. เมื่อรวมทำการย่อ ไอคอน Noisy ที่เป็น UserObject จะเป็นดังรูปที่ 3.9 ซึ่งการใช้ UserObject ช่วยให้เราสะดวกและจัดหมวดหมู่ของการโปรแกรมได้ดีขึ้นอีกด้วย



รูปที่ 3.9 Noisy Cosine โปรแกรมหลังจากที่ได้ทำการย่อเป็นไอคอน

- **การเพิ่มโปรแกรม**

เราจะทำการแบ่งแยกโครงสร้าง ภายใต้โปรแกรม ให้ง่ายขึ้นไปอีกในการประยุกต์ใช้กับโปรซีเยอร์ ต่างๆของโปรแกรม เพื่อที่จะสร้างหัวเรื่องของโปรแกรม

1. คลิกที่ File -> Edit Properties เลือก PulseProgram ในส่วนของหัวเรื่องจากนั้นคลิกที่ OK

- **สำหรับสร้างทดสอบและปรับปรุงการใช้งาน Procedure**

โดยการเก็บ Procedure ทดสอบไว้เป็น UserObject โดยเราสามารถทดสอบไอคอนทั้งหมดที่อยู่ภายใต้หน้าจอมอนิเตอร์ อย่างไรก็ตามเราสามารถ เอาโปรแกรมนี้ร่วมกับโปรแกรมอื่นๆ ได้อย่างง่าย เพราะว่ามันง่ายที่เราจะเชื่อมต่อระหว่าง โปรแกรมและท้ายสุดนี้ UserObject ยังง่ายต่อการปรับแต่งฟังก์ชั่น ในฟังก์ชั่นไหนก็ตามเราสามารถเรียกแบบฟอร์มของฟังก์ชั่น หรือเบอร์ ตำแหน่งออกมาได้จากโปรแกรมอีกที

1. เลือก Device => UserObject และวางทับไปที่พื้นที่ของการโปรแกรม (Work Area) ได้ทันที
2. ขยายขนาดของ UserObject โดยเลือกคลิกที่ตัวปุ่มที่มีข้อความว่า maximize bottom ซึ่งอยู่มุมบนด้านขวาของ Object (ถ้าเราคลิกบนตัวไอคอนด้านบน โดยการคลิกสองครั้งที่ไอคอน มันจะทำการ maximize ให้เราได้เอง)

- **การตั้งค่าของ Pulse Generator**

เราจะใช้แหล่งจ่ายเป็นในลักษณะวิชวล (virtual source) โดยเราเรียกเอาแหล่งจ่ายได้จากการจำลองข้อมูล บนเครื่องมือวัด ซึ่งในที่นี้เราจะดัดแปลงการติดต่อสื่อสาร กับ ค่าจริงๆที่ได้จากการวัด โดยเราต้องเพิ่มข้อมูลรอยต่อทางด้านอินพุท ซึ่งรับเอาค่าความถี่ โดยตั้งค่า Object ให้เป็นจำนวนเต็ม

1. เลือก Device => Virtual Source => Pulse Generator, และวางมันลงไปทางด้านขวาของกล่องทางด้านอินพุท
2. ทำการเปิด Pulse Generator Object เมนูและเลือกคลิกที่ Add Terminal => Data Input ซึ่งค่าที่ได้ในกล่องรายละเอียดนี้ จะมีข้อมูลรอยต่อทางด้านอินพุทเกิดขึ้น เราอาจจะเพิ่ม หรือ เปลี่ยนแปลง ฟังก์ชั่นได้ตามต้องการบน Pulse Generator. ซึ่งความถี่ที่เลือกใช้ในตอนนี้นั้นจะถูกเน้นเป็นสีสันให้เห็นชัดเจน ดังนั้นให้เราคลิก OK โดย HP VEE จะเพิ่มจุดต่อขึ้น
3. ต่อทางด้านบนของข้อมูลรอยต่อทางด้านเอาท์พุท ซึ่งก็คือข้อมูลจำนวนเต็มทางด้านขาเข้า (Integer Input) ซึ่ง Object ดังกล่าวจะเชื่อมต่อกับ Pulse Generator ซึ่งเป็นข้อมูลจากรอยต่อทางด้านอินพุท
4. ทำการรันโปรแกรมในส่วนนี้ โดยใส่ค่าตัวแปรที่ต่างๆ ออกไปดู ซึ่งใส่ในกล่องทางด้านอินพุท ซึ่งเราดูได้จากรูปที่ 3.4

ส่วนมากเราจะเลือก File -> Exit ที่จะ stop HP VEE เราจะอาจจะไม่จำเป็นที่จะใช้เทคนิคดังกล่าว แต่บางครั้งเราต้องทำอะไรบางอย่างที่สร้างสรรค์มากๆและ HP VEE ก็ไม่จำเป็นที่จะต้องใช้เมาส์และคีย์บอร์ด

1. กด Ctrl-Alt-Delete แล้ว windows จะมี Option ขึ้นมาเพียงแต่ทำตามคำแนะนำ ใน window (สำหรับ MS Windows) หรือ คลิก End Task ใน Win95

ใน Window NT, คุณต้อง lock application แต่คุณยังคงต้องใช้เมาส์ในการควบคุม ดังนั้น

คลิกสองครั้งที่ด้านหลังแล้วเลือก End Task จาก Pop-Up Window ใน Unix เราต้องการ 'Kill' ขั้นตอนการทำงาน

1. Enter ps-ef | grep vee ใน HP-UX หรือ Solaris (หรือ ps-avx|grep vee ใน SunOs) ที่ Prompt เพื่อระบุจำนวนขั้นตอนการทำงาน เราจะเห็นเส้นของ veetest ที่ End จำนวนที่คุณ Login จะทำงานตามจำนวนที่คุณต้องการในกรณีนี้ Bobh<number>...veetest
2. Enter Kill-9<number> เพื่อหยุดการทำงานของ HP VEE คุณสามารถ Enter veetest เพื่อเริ่มการทำงานได้อย่างง่ายดาย

❑ **เราได้เรียนรู้เกี่ยวกับ HP VEE ดังนี้**

ซึ่งเราสามารถทำงานต่างๆ ในแต่ละหัวข้อต่อไปนี้ได้

- ระบุ แถบเมนูหลัก, ปุ่มโปรแกรมการทำงาน, และ พื้นที่ของโปรแกรม
- การเลือกเมนูจากเมนหลัก และ Object เมนู
- เปิด Object เมนูได้ 2 ทาง
- การบันทึกงาน,การออกจาก HP VEE, การเริ่มต้นในการทำการโปรแกรม
- สามารถทำงานเกี่ยวกับ Object เช่นการเคลื่อนย้าย, เปลี่ยนชื่อ, การจัดการไอคอน, การขยาย, การปรับแต่งขนาด, การเลือก และ ไม่เลือกใช้ Object, การลบ
- ระบุ ข้อมูลและลำดับ Pin ของ Object และอธิบายจุดประสงค์
- ตรวจสอบ Terminal และเปลี่ยนชื่อ
- เคลื่อนย้ายพื้นที่ของโปรแกรม, และการล้าง
- อธิบาย UserObject และจุดประสงค์
- เขียนโปรแกรมเล็กๆใน UserObject
- ย่อ,Restore UserObject
- การรัน และการบันทึก โปรแกรม
- ตั้งค่าตัวแปรและแก้ไข

5. ไอคอน Pulse Generator เราสามารถที่จะเคลื่อนย้าย หรือ ปิดมันไปได้ เพื่อทำให้ง่ายขึ้นจากการดูกล่องของหน้าจอมอนิเตอร์

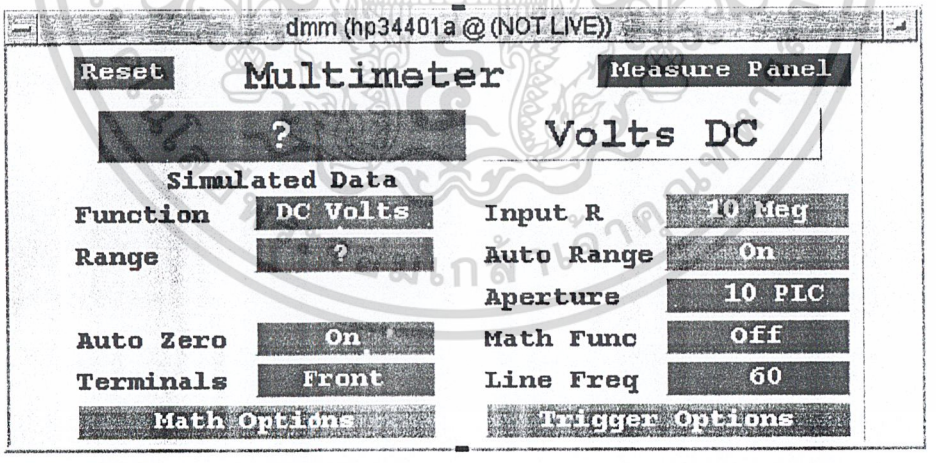
3.8 การใช้งาน HP VEE กับเครื่องมือวัด

กับ HP VEE เราสามารถใช้งานมัน เพื่อพัฒนาตัวของโปรแกรม โดยปราศจากการต่อเครื่องมือวัดโดยตรงเข้ากับเครื่องคอมพิวเตอร์เพื่อใช้งาน สำหรับวัตถุประสงค์ของขั้นตอนนี้ก็เพื่อที่จะทำโครงร่างของมัลติมิเตอร์ (HP 3478A) สำหรับใช้งานกับเครื่องวัดที่มี ไดร์เวอร์อยู่ในลิสท์อยู่ ดังนั้นเราจึงสามารถเรียกมาแสดงผลได้อย่างง่ายดาย ถ้าเพียงแต่เราต้องการเพิ่มเครื่องมือวัดลงไปในโครงร่างของโปรแกรม

3.9 การใช้งาน HP VEE กับ Multimeter

เป็นตัวอย่างของการใช้งาน HP VEE กับเครื่องมือวัดมัลติมิเตอร์ โดยปราศจากเครื่องมือวัดจริง เหมือนเป็นการจำลอง มัลติมิเตอร์ขึ้นมา

1. เลือก I/O => Other Instrument ... โดยเราทำการเลือกรายชื่อของเครื่องมือวัด เมื่อเราได้เลือกว่าจะจำลองเครื่องมือวัดชนิดใดขึ้นมาแล้วใน ให้เราเลือกคลิกที่ Panel Driver และจากภาพที่แสดงข้างล่างคือ ภาพที่เราได้ทำการจำลองเครื่องมือวัดมัลติมิเตอร์ออกมา



รูปที่ 3.10 การจำลองมัลติมิเตอร์

เราสังเกตุได้ว่ามีการปรับแต่งและตั้งค่าต่างๆ เหมือนกับมัลติมิเตอร์จริง ซึ่งนี้ก็คือตัวอย่างหนึ่งของการจำลองเครื่องมือวัดด้วย HP VEE

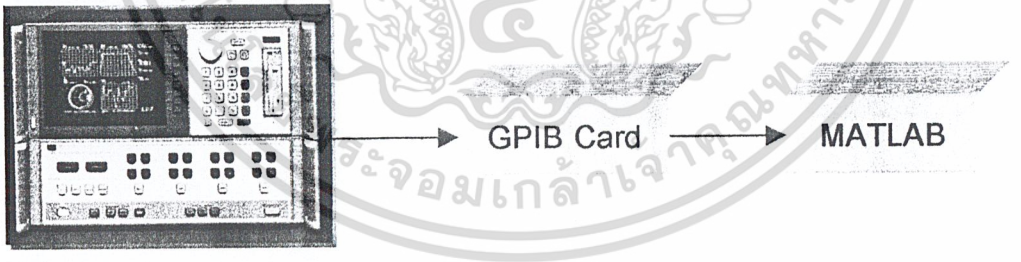
บทที่ 4

MATLAB

Matlab เป็นภาษาคอมพิวเตอร์ชั้นสูง (High-Level Language) สำหรับการคำนวณทางเทคนิคที่ประกอบด้วยการคำนวณเชิงตัวเลข กราฟิกที่ซับซ้อน และการจำลองแบบเพื่อให้มองเห็นภาพพจน์ได้ง่ายและชัดเจน ชื่อของ Matlab ย่อมาจาก Matrix Laboratory เดิมโปรแกรม Matlab ได้เขียนขึ้นจากโพรเจ็กที่ชื่อ Linkpack และ Eispack

Matlab ได้พัฒนาด้วยการแก้ปัญหาที่ส่งมาจากหลายๆ ผู้ใช้เป็นระยะเวลาหลายปี จึงทำให้โปรแกรม Matlab มีฟังก์ชันต่างๆ ให้เลือกใช้มากมาย ในบางมหาวิทยาลัยได้ใช้โปรแกรม Matlab เป็นหลักในหลักสูตรพื้นฐานในการศึกษาทางด้านคณิตศาสตร์ วิศวกรรม และวิทยาศาสตร์แขนงต่างๆ ตลอดจนในด้านอุตสาหกรรม ได้ใช้โปรแกรม Matlab เป็นเครื่องมือสำหรับใช้ในงานวิจัยพัฒนาและวิเคราะห์

โปรแกรม Matlab จะมีกล่องเครื่องมือที่ใช้ในการหาคำตอบเรียกว่า toolbox โดยโปรแกรม Matlab จะมี toolbox ในแต่ละสาขา เช่น การประมวลผลสัญญาณ (signal processing toolbox) โครงข่ายประสาท (naural networks toolbox) ฟัซซี่ลอจิก (fuzzy logic toolbox) เวฟเลท (wavelet toolbox) การติดต่อสื่อสาร (communication toolbox) สถิติ (statistic toolbox) และด้านอื่นๆอีกมากมาย ภายใน toolbox ก็ยังมีฟังก์ชันต่างๆที่เกี่ยวข้องกับการแก้ปัญหาในสาขานั้นๆ ให้เลือกประยุกต์ใช้งานเป็นจำนวนมาก



รูปที่ 4.1 บล๊อกไดอะแกรม

จากรูปที่ 4.1 แสดงถึงบล๊อกไดอะแกรมของการอินเตอร์เฟซ กันระหว่างเครื่องมือวัด เนทเวิร์กอนาไลเซอร์กับคอมพิวเตอร์โดยใช้โปรแกรม Matlab ในการควบคุม โดยผ่านการ์ด GPIB สำหรับเป็นตัวอินเตอร์เฟซ

เนื่องจากโปรแกรม Matlab เป็นโปรแกรมที่มีโครงสร้างลักษณะเหมือนภาษา C ทำให้ผู้ใช้ที่มีพื้นฐานของภาษา C สามารถใช้งานได้ง่ายขึ้น และโปรแกรม Matlab นั้นเป็นโปรแกรมในเชิงคณิตศาสตร์ จึงทำให้การนำผลหรือค่าที่ได้จากเครื่องมือวัด (Instrument) มากระทำการเก็บค่า หรือกระทำการใดๆในเชิง คณิตศาสตร์นั้นจะทำได้ง่ายกว่าโปรแกรมอื่นๆ

4.1 อาร์เรย์และการดำเนินการ เกี่ยวกับอาร์เรย์

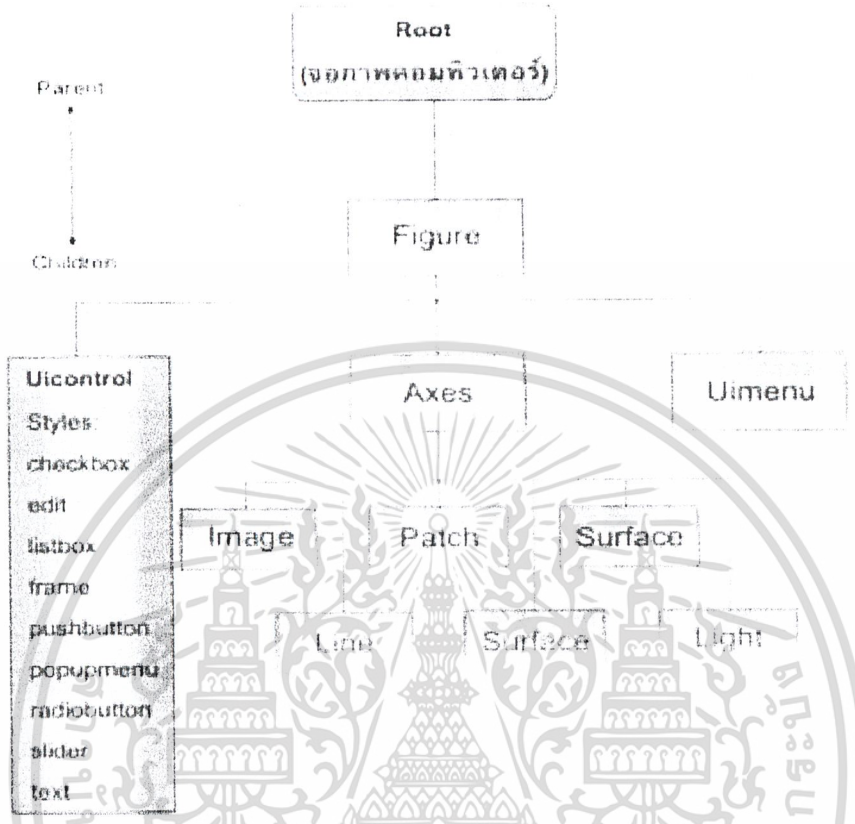
บ่อยครั้งข้อมูลมักจะถูกจัดเรียงในรูปแถวลำดับซึ่งมีอยู่ในรูปเซตของสมาชิกที่มีหนึ่งตัวหรือมากกว่า เรียกว่า อาร์เรย์ (Array) เราเรียกแถวลำดับ 1 มิติ หรืออาร์เรย์ 1 มิติว่า เวกเตอร์ และเรียกแถวลำดับ 2 มิติหรืออาร์เรย์ 2 มิติว่าเมตริกซ์ เวกเตอร์และเมตริกซ์จะมีสมาชิกเป็นจำนวนจริงหรือเรียกจำนวนจริงนี้ว่า สเกลาร์ (scalar)

โปรแกรม Matlab จะทำการแก้ปัญหาเกี่ยวกับข้อมูลที่เป็นตัวแปรค่าต่างๆ ซึ่งค่าตัวแปรอาจเป็นเมตริกซ์ เวกเตอร์ หรือสเกล่าร์ก็ได้ขึ้นอยู่กับรูปแบบของปัญหา โปรแกรม Matlab มีพื้นฐานมาจากเมตริกซ์โดยการกระทำฟังก์ชันต่างๆมักจะเกี่ยวข้องกับเมตริกซ์ ดังนั้นเมตริกซ์จึงเป็นส่วนที่สำคัญมากต่อโปรแกรม Matlab

อาร์เรย์หรือเมตริกซ์จะใช้แก้ปัญหาในการคำนวณเชิงตัวเลขที่เป็นสเกลาร์ที่มีจำนวนมาก แต่ต้องการคำนวณในครั้งเดียวโดยไม่ต้องกระทำหลายๆครั้งที่ทำให้สูญเสียเวลา

4.2 การเชื่อมต่อกับผู้ใช้ทางกราฟิก

การเชื่อมต่อกับผู้ใช้ทางกราฟิก (Graphic User Interface : GUI) เป็นวิธีการเชื่อมต่อกันระหว่างผู้ใช้และคอมพิวเตอร์โดยคอมพิวเตอร์จะถูกใช้งานก็ต่อเมื่อผู้ใช้ได้ป้อนข้อมูลที่ต้องการผ่านทางคีย์บอร์ด เมาส์ trackball, drawing pad และไมโครโฟนอย่างใดอย่างหนึ่งให้กับคอมพิวเตอร์ คอมพิวเตอร์จะแสดงตัวอักษรและกราฟิกต่างๆ บนจอภาพ การเชื่อมต่อกับผู้ใช้ทางกราฟิก (GUI) จะสร้าง object ต่างๆที่ใช้สำหรับการติดต่อหรือใช้งานร่วมกันก็คือ หน้าต่าง, ไอคอน, ปุ่ม, กรอบ, เมนู, popup และตัวอักษรต่างๆ ซึ่งในการโปรแกรมเพื่อต้องการควบคุมเครื่องมือวัดในโครงงานนี้นั้นได้นำเอา การเชื่อมต่อกับผู้ใช้ทางกราฟิก (GUI) มาใช้เป็นส่วนสำคัญในการเขียนควบคุมเครื่องมือวัด ลักษณะของตัวโปรแกรมเองก็เป็นในรูปแบบของ การเชื่อมต่อกับผู้ใช้ทางกราฟิก (Graphic User Interface : GUI)



รูปที่4.2 ลักษณะของโครงสร้างฟังก์ชันที่ใช้ในการติดต่อกับผู้ใช้ (GUI)

4.2.1 Handle Graphic

Handle Graphic เป็นพื้นฐานอยู่บนแนวความคิดที่ว่าทุกๆสิ่งบนหน้าต่างรูปภาพของโปรแกรม Matlab จะเป็นวัตถุ (Object) เนื่องจากทุกๆ object ที่อยู่บนจอภาพหรือหน้าต่างรูปภาพจะมีเอกลักษณ์เฉพาะตัว (unique idedntifier) ซึ่งเรียกว่า Handle และแต่ละวัตถุมี properties ที่สามารถปรับปรุงแก้ไขได้ตลอดเวลา ทุกอย่างที่สร้างโดยใช้คำสั่งทางกราฟิกจะเรียกว่า graphics object

ชนิดของ Control Objects ชนิดของ Control Objects ที่ Matlab ได้เตรียมไว้ ด้วยการเลือก โดยวางตำแหน่งของ pointer แล้วจึงกด mouse (สำหรับ MS-Windows ปุ่มซ้าย) มีดังต่อไปนี้

- **Push Buttons** เป็น screen objects เล็กๆ ขนาดตามที่เรากำหนด สามารถที่จะเขียนสัญญลักษณ์ใดๆ ตามที่เราต้องการ โดยการเลือกและกดไปที่ mouse ณ ตำแหน่งที่ปรากฏจะทำให้ Matlab ปฏิบัติตามที่กำหนดไว้ล่วงหน้า
- **Check Boxes** Check boxes สามารถให้ผู้ใช้เลือกได้มากกว่าหนึ่งทางเลือก ซึ่งในแต่ละทางเลือก คล้ายกับสวิทช์ ปิด/เปิด แสดงให้เห็นว่าในแต่ละทางเลือกนั้น อยู่ตำแหน่ง on หรือ off และจะ on เมื่อ box นั้นถูกเลือก และ off เมื่อ box นั้นไม่ถูกเลือก check boxes แต่ละอันไม่ขึ้นแก่กัน สามารถเลือก และไม่เลือกจำนวนเท่าใดก็ได้ เมื่อเลือกในแต่ละ box Matlab จะปฏิบัติตามหน้าที่แต่ละ box นั้นๆกำหนดไว้ล่วงหน้า
- **Radio Buttons** คล้ายกับ check boxes แต่สามารถเลือกได้หนึ่งทางเลือกเท่านั้น หากเลือกทางเลือกใหม่ ทางเลือกใหม่นั้นจะอยู่ในสภาวะ on ทางเลือกเก่าจะอยู่ในสภาวะ off และ Matlab จะปฏิบัติตามที่ทางเลือกนั้นๆ กำหนดไว้ล่วงหน้า
- **Sliders** ให้ผู้ใช้เลือกค่าที่อยู่ภายในย่านนั้น เป็นอุปกรณ์ทางอนาลอก แสดงให้เห็นค่าที่เลือกทาง รูปภาพ สามารถเปลี่ยนได้ด้วยการเลื่อนตัวชี้ (indicator) ค่าที่เปลี่ยนไปจะทำให้ Matlab ปฏิบัติการตามที่กำหนดไว้ล่วงหน้า slider ประกอบด้วย 3 ส่วน คือ
  1. ส่วนที่แสดงย่านของค่าที่เลือกได้ กดที่ส่งนั้นจะทำให้ค่าเปลี่ยนไปประมาณ 10%
  2. ตัวเลื่อนที่แสดงตำแหน่งสัมพันธ์กับค่าที่เลือก
  3. ลูกศรที่ปลายทั้งสองข้างของ indicator กดที่ปุ่มเหล่านี้ จะทำให้ค่าเปลี่ยนไป ประมาณ 1 %
- **Pop-up Menus** ให้ผู้ใช้สามารถเลือกสิ่งที่อยู่ในเมนูได้ โดย pop-up menus นี้ จะไม่อยู่บน menubar แต่จะอยู่บน window รูปภาพ ตามขนาดและตำแหน่งที่กำหนด เมื่อเลือกสิ่งที่อยู่ในเมนูนี้ จะทำให้ Matlab ปฏิบัติตามสิ่งที่เลือก
- **Static Text** แสดง text ที่ต้องการให้ปรากฏหนึ่งบรรทัด โดยทั่วไปใช้เป็นสัญญลักษณ์ในการรวมกลุ่มของ controls อื่นๆ เพื่อให้ผู้ใช้ทราบ หรือแสดงค่าที่สัมพันธ์กับ silder ผู้ใช้ไม่สามารถเปลี่ยนแปลง text ที่ปรากฏนี้ได้ด้วยการปฏิสัมพันธ์ โดยปกติ static text controls นี้จะใช้กับ sliders และ frames
- **Editable Text** เป็น text ที่เปลี่ยนแปลงได้ โดยผู้ใช้สามารถใส่ค่า string เพื่อใช้ในการ application ต่างๆ ค่านี้สามารถใส่เพียงบรรทัดเดียว หรือหลายๆบรรทัดได้ ผู้ใช้สามารถเปลี่ยนแปลง หรือลบทิ้งได้
- **Frames** ทำให้สามารถที่จะล้อมกรอบบริเวณที่ต้องการ ภายใน window รูปภาพ โดยปกติมักจะรวม control ต่างๆที่อยู่ในลักษณะที่ใช้ร่วมกันอยู่ใน frame เดียวกัน

4.2.2 ฟังก์ชัน uicontrol

การควบคุม object ทางภาพ คือ การที่เราเคลื่อน point บนจอภาพ ด้วย mouse หรือสิ่งอื่นๆ ไปยัง object ทางภาพนั้น และ activate ณ ตำแหน่งทาง pointer ที่ชี้นั้น ทำให้เกิด การปฏิบัติการตามที่ได้กำหนดไว้ล่วงหน้า ณ object นั้นๆ ซึ่งเราสามารถสร้าง control ต่างๆได้หลายลักษณะ

K = uicontrol('PropertyName',PropertyValue,...)

K = uicontrol(kfig,'ชื่อลักษณะ',ตัวลักษณะ,...)

Uicontrol สร้างปฏิสัมพันธ์กับผู้ใช้ ตามลักษณะ และตัวลักษณะเป็นคู่ๆไป หากมี argument ตัวแรกนอกเหนือจากนี้ (เช่น kfig) argument ตัวแรกนี้จะเป็น parent ของรูป object ที่ uicontrol จะนำไปแสดงลง

- **ชื่อลักษณะ Call Back ตัวลักษณะ string**

เป็นปฏิบัติการควบคุม ตัวลักษณะนี้ใช้ได้ถึงแม้จะเป็น Matlab expression ใดๆที่ถูก syntex รวมทั้งชื่อของ M-file หรือฟังก์ชัน

- **ชื่อลักษณะ Children ตัวลักษณะ Handles**

เป็น Children object uicontrol ต่างๆ ไม่มี children ดังนั้นลักษณะนี้จะให้ empty matrix เสมอ

- **ชื่อลักษณะ Fore Ground Color ตัวลักษณะ Colorspec**

เป็นสีของ text ให้สีของ text ที่แสดงบน uicontrol object

- **ชื่อลักษณะ Horizontal Alignment ตัวลักษณะ left – center – right**

เป็นการจัดตามแนวนอนของ label string บอกให้ text ที่เขียนลงไปอยู่ ณ ตำแหน่ง ใดบน contrrol

- **ชื่อลักษณะ Max ตัวลักษณะ สเกลา**

เป็นค่าสูงสุดที่กำหนด ลักษณะนี้กำหนดค่าสูงสุดที่ให้เป็นไปได้ในลักษณะของค่า control สำหรับ radio button และ check box จะทำหน้าที่เป็นสวิทช์ เปิด/ปิด ซึ่งใช้แทนลักษณะการตั้งของค่าเมื่อ uicontrol อยู่ที่ตำแหน่ง on สำหรับ popup menu จะเป็นการใช้จำนวนการเลือกที่กำหนดไว้ใน popup menu นั้นมากที่สุดที่จะเป็นไปได้ สำหรับ slider ลักษณะค่านี้คือค่าสูงสุดใน slider ที่จะสามารถเลือกได้ default คือ หนึ่ง

- **ชื่อลักษณะ Min ตัวลักษณะ สเกลา**

เป็นค่าต่ำสุดที่กำหนด ลักษณะค่าต่ำสุดที่ให้เป็นไปได้ ในลักษณะของค่า control สำหรับ radio button และ check box จะทำหน้าที่เป็นสวิทช์ เปิด/ปิด ซึ่งใช้แทนลักษณะการตั้งค่า เมื่อ uicontrol อยู่ที่ตำแหน่ง off สำหรับ slider ลักษณะค่านี้ คือค่าสูงสุดใน slider ที่จะสามารถเลือกได้ default คือ ศูนย์

- **ชื่อลักษณะ Parent ตัวลักษณะ handle**

เป็น Handle ของ object's parent ของ uicontrol object คือ object รูปที่ uicontrol แสดงลักษณะในขณะนั้น เราสามารถตั้งลักษณะนี้ได้โดยกำหนด handle ของ parent object แรก ในฟังก์ชัน uicontrol (โดยไม่มีคำว่า parent) เช่น

uicontrol (1, 'Style', 'pushbutton', ...)

หากตัวแรกนั้นเป็น gef ฟังก์ชัน gef นี้ จะให้ handle กลับไปยังรูปปัจจุบัน

- **ชื่อลักษณะ Position ตัวลักษณะ เวกเตอร์ 4 ค่า (1xw4)**

เป็นตัวกำหนดตำแหน่งของ object ด้วยการกำหนดขนาดและตำแหน่งของ contrrol ที่ปรากฏบน window รูปภาพขนาดและตำแหน่งกำหนดโดยในเวกเตอร์ 1x4 ดังนี้

rec = [ซ้าย ล่าง กว้าง สูง]

ซึ่งซ้ายและล่าง เป็นการกำหนดระยะจากมุมซ้ายล่างของ window รูปภาพไปยังมุมซ้ายด้านล่างของ control object ตามลำดับ ส่วนกว้างและสูงเป็นการกำหนดความกว้างและสูงของขนาดของสี่เหลี่ยมที่จะปรากฏเป็นตัว control รายละเอียดของหน่วยที่สามารถใช้ได้อยู่ในสัญญลักษณ์ชื่อ Units

- **ชื่อลักษณะ String ตัวลักษณะ String**

สัญลักษณ์ control เป็นการกำหนดสัญญลักษณ์ให้ปรากฏบนตัว control ต่างๆ เช่น push buttons, radio buttons, check boxes และ popup menus สำหรับหลายๆทางเลือกบน popup menu หรือ editable text ป้อนแต่ละ string แยกด้วยเครื่องหมาย 1 (หรือ) ไปเรื่อยๆ ตามลำดับ และให้ทั้งหมดอยู่ภายใต้เครื่องหมาย quotes ('...|...') สำหรับ editable text ลักษณะของ string กำหนดได้โดยการพิมพ์ inout จาก user ส่วน sliders จะไม่มีการแสดงสัญญลักษณ์

- **ชื่อลักษณะ Style ตัวลักษณะ pushbutton / radiobutton / check box / slider/edit / popupmenu**

ชนิดของ uicontrol object ลักษณะนี้เป็นการกำหนดชนิดของ control ที่เราต้องการจะสร้าง รายละเอียด

- **ชื่อลักษณะ Type ตัวลักษณะ string (read only)**

ชนิดของ graphic object ลักษณะนี้เป็นการกำหนดชนิดของ graphics object สำหรับ uicontrol object string ของ type นี้ คือ 'uicontrol'

- **ชื่อลักษณะ Unitse ตัวลักษณะ pixels / normalized / inches / centimeters / pointes**

หน่วยของการวัด ลักษณะนี้กำหนดหน่วยที่ใช้ โดยชื่อลักษณะ Position ทุกๆ หน่วย จะเริ่มต้นวัดจากมุมซ้ายล่างของ window รูปภาพ หน่วย normalized จะกำหนดมุมซ้ายล่างเป็นจุด (0 , 0) และมุมขวาบนจุด (1.0 , 1.0) inches, centimeters, และ points เป็นหน่วย absolute (หนึ่งจุดเท่ากับหนึ่งส่วนเจ็ดสิบสองนิ้ว)

- **ชื่อลักษณะ UserData ตัวลักษณะ เมตริกซ์**

ข้อมูลที่กำหนดโดยผู้ใช้ ข้อมูลนี้สามารถเป็นเมตริกซ์บนจุดใดๆ ที่เราต้องการให้มีส่วนร่วมกับ object ที่สร้างขึ้น object ไม่สามารถที่จะใช้ข้อมูลนี้ เราสามารถนำมาใช้ได้ด้วยการใช้ฟังก์ชัน get

- **ชื่อลักษณะ Value ตัวลักษณะ สเกลา**

ค่าปัจจุบันของ control ค่าที่เป็นไปได้ของ control นี้ขึ้นอยู่กับชนิดของ control

- Radio buttons และ check boxes ตั้ง Value ไว้ที่ Max (ปกติเป็น 1) เมื่ออยู่ในสภาวะ on (เมื่อปุ่มถูกกด) และ Min (ปกติปิดเป็น 0) เมื่ออยู่ในสภาวะ off (เมื่อปุ่มถูกปล่อย)
- Sliders ตั้ง Value ไว้ที่ตัวเลขที่กำหนด โดย slider bar ซึ่งสัมพันธ์กับขอบเขตที่ ตั้งไว้ด้วย Min และ Max
- Popup menus ตั้ง Value ไว้ที่ดัชนีของสิ่งที่ถูกเลือก
- Push buttons และ editable controls ไม่ต้องใช้ชื่อลักษณะนี้

เราสามารถกำหนดชื่อลักษณะ Value นี้ได้ด้วยการปฏิสัมพันธ์โดยใช้ mouse หรือ กำหนดฟังก์ชัน set การเปลี่ยนแปลงสามารถเห็นได้โดยตรง

- **ชื่อลักษณะ Visible ตัวลักษณะ on / off**

ความสามารถในการมองเห็น object ลักษณะนี้กำหนด object ที่สร้างขึ้นให้ปรากฏบนจอภาพหรือไม่

4.2.3 ฟังก์ชันและการสร้าง Menus

การสร้าง menu สามารถทำได้ด้วยฟังก์ชัน uimenu ตามลักษณะดังนี้

h = uimenu ('Property Name', Property Value, ...)

หรือ submenu ด้วย

hsub = uimenu (h, 'Property Name', Property Value,...)

ฟังก์ชัน uimenu นี้จะสร้าง menu ให้ปฏิบัติตามที่กำหนดไว้ล่วงหน้า ซึ่ง menu ที่สร้างนี้จะอยู่ส่วนบนของรูป ใน menu bar แต่ละทางเลือกใน menu สามารถที่จะเป็น menu ด้วยตัวเองได้ นั่นคือมี submenu การสร้าง submenu ด้วยการส่ง handle ไปยัง parent menu โดย argument ตัวแรก หากไม่มี menu นั้นกลายเป็น top-level menu ชื่อลักษณะและตัวลักษณะเป็นไปได้ดังนี้

- **ชื่อลักษณะ Accelerator ตัวลักษณะ (character)**

ใช้เรียกให้ปฏิบัติการโดยตรงได้จาก keyboard โดยไม่ต้องเลื่อน mouse ไปที่ menu แล้วกดเรียก key ที่ต้องใช้กด กำหนดไว้ล่วงหน้าตามคำสั่งบนเครื่อง PC การใช้งาน คือกด Control key ค้างไว้ แล้วกดตัวอักษะที่ระบุ

- **ชื่อลักษณะ CallBack ตัวลักษณะ string**

ใน menu ปฏิบัติตามที่กำหนดไว้ล่วงหน้า ตามลักษณะ ไม่ว่าจะเป็น Matlab expression ใดๆ ที่ถูก syntex รวมทั้งชื่อของ M-file หรือฟังก์ชัน

- **ชื่อลักษณะ Children ตัวลักษณะ เวกเตอร์ของ handels**

เป็น children object ของ uimenu คือ uimenu อื่นๆที่เป็น submenu

- **ชื่อลักษณะ Enable ตัวลักษณะ on / off**

เป็นสภาวะให้เลือก ได้ / ไม่ได้ ลักษณะจะควบคุมให้เลือกสิ่งใน menu ได้หรือไม่ได้ หากอยู่ในสภาวะที่ให้เลือกไม่ได้ สัญญลักษณ์บน menu จะเป็นสีอ่อน แสดงว่าให้เลือกไม่ได้

- **ชื่อลักษณะ Label ตัวลักษณะ string**

  เป็นสัญญลักษณ์ที่เขียนขึ้นบน object

- **ชื่อลักษณะ Parent ตัวลักษณะ handle**

  เป็น handle ของ object ที่เป็น parent ซึ่ง parent สำหรับ uimenu เป็นรูป หรือ uimenu ซึ่งหากต้องการสร้าง submenu จะต้องระบุลักษณะนี้ไว้

- **ชื่อลักษณะ Position ตัวลักษณะ scalar**

  เป็นตำแหน่ง menu สัมพัทธ์กำหนดการวาง menu items หากเป็น top-level menu จะวางจากซ้ายไปขวาบน menu bar ตามค่าที่ระบุไว้ในลักษณะ Position แต่ละ item ภายใน menu ที่กำหนดไว้ จะวางจากบนลงล่างตามค่าที่ระบุไว้ในลักษณะ Position

- **ชื่อลักษณะ Separator ตัวลักษณะ on / off**

  จะลากเส้นตามแนวนอนคั่นระหว่าง menu item ด้านบน เมื่ออยู่ในสภาวะ on ไม่ใช้กับ top-level menu

- **ชื่อลักษณะ Type ตัวลักษณะ string (read only)**

  ชนิดของ graphic object สำหรับ uimenu object type คือ string "uimenu"

- **ชื่อลักษณะ User Data ตัวลักษณะ เมตริกซ์**

  ข้อมูลที่กำหนดโดยผู้ใช้ ข้อมูลนี้สามารถเป็นเมตริกซ์บนจุดใดๆ ที่เราต้องการให้มีส่วนร่วมกับ object ที่สร้างขึ้น object ไม่สามารถที่จะใช้ข้อมูลนี้ เราสามารถนำมาใช้ได้ด้วยการใช้ฟังก์ชัน get

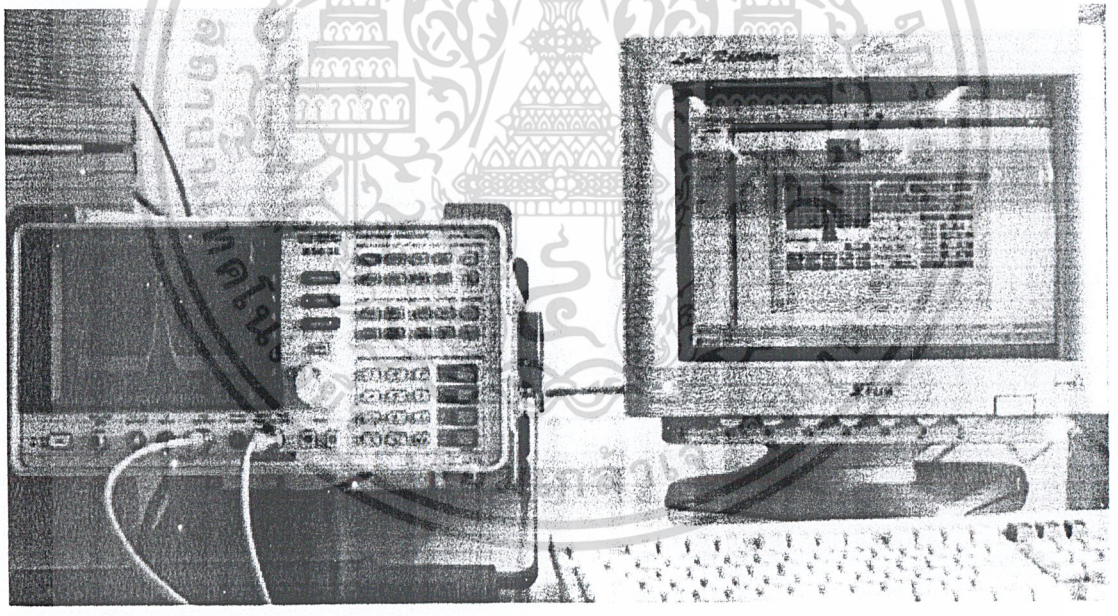
- **ชื่อลักษณะ Visible ตัวลักษณะ on / off**

  ความสามารถในการมองเห็น object ลักษณะนี้กำหนด object ที่สร้างขึ้นให้ปรากฏบนจอภาพ หรือไม่

บทที่ 5
การโปรแกรมและผลการทดลอง

การทดสอบใช้งานโปรแกรม HP VEE ที่เขียนขึ้นเพื่อควบคุม เครื่องสเปคตรัมอนาไลเซอร์ และโปรแกรม Matlab สำหรับควบคุมเครื่องเนทเวิร์กอนาไลเซอร์ โดยอุปกรณ์ที่ใช้ในการทดลองมีดังนี้

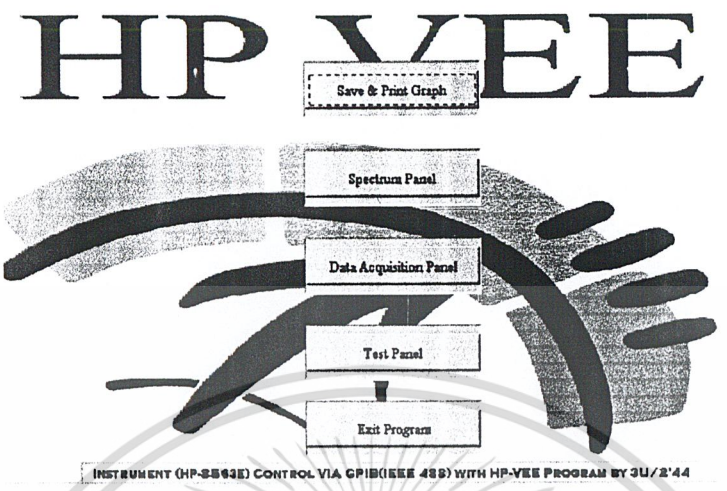
- เครื่องคอมพิวเตอร์
- การ์ด GPIB
- เครื่อง Spectrum Analyzer
- เครื่อง Network Analyzer
- เครื่องพิมพ์ (Printer)



รูปที่ 5.1 การติดต่อระหว่างเครื่องมือวัดกับคอมพิวเตอร์

รูปที่ 5.1 แสดงให้เห็นว่าคอมพิวเตอร์สามารถที่จะติดต่อกับเครื่องมือวัดได้ และยังสามารถที่จะดึงข้อมูลจากเครื่องสเปคตรัมอนาไลเซอร์ได้อีกด้วย โดยการทำงานเป็นในลักษณะของการจำลองเครื่องสเปคตรัมอนาไลเซอร์ไว้ใช้งานบนคอมพิวเตอร์ โดยสามารถสั่งงานและควบคุมได้

5.1 โปรแกรมควบคุม Spectrum Analyzer ด้วย HP VEE

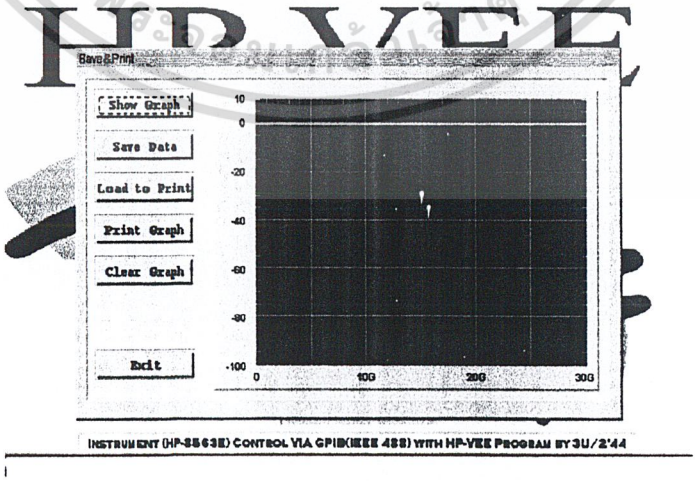


รูปที่ 5.2 เมนูหลักของโปรแกรม

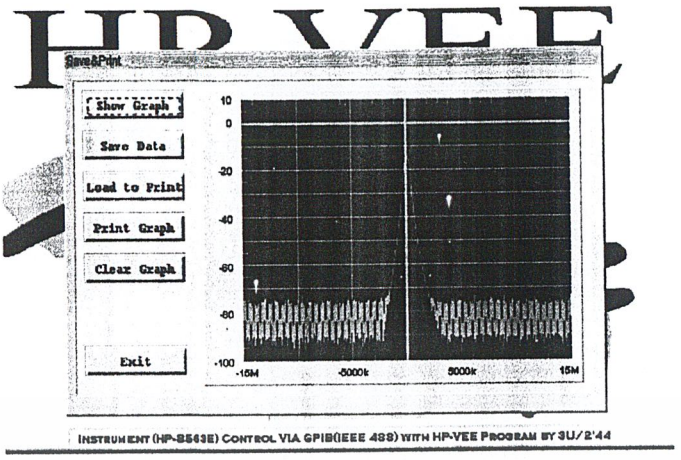
จากรูปที่ 5.2 เป็นรูปที่แสดงถึงเมนูหลักของโปรแกรมซึ่งประกอบด้วย เมนูที่ใช้ในการเลือกฟังก์ชันการทำงานของโปรแกรมเพื่อควบคุมเครื่องสเปคตรัมอนาไลเซอร์ ซึ่งในที่นี้ประกอบด้วย 4 ส่วนคือ Save & Print Graph, Spectrum panel, Data Acquisition Panel, Test Panel โดยทั้ง 4 ส่วนจะมีฟังก์ชันย่อยๆสำหรับใช้งาน โดยสามารที่จะเลือกใช้งานโดยการเรียกใช้ Call Function

- **Save & Print Graph**

  คือโหมดของการใช้งานเพื่อสำหรับดึงข้อมูลจากเครื่องสเปคตรัมอนาไลเซอร์โดยเฉพาะ

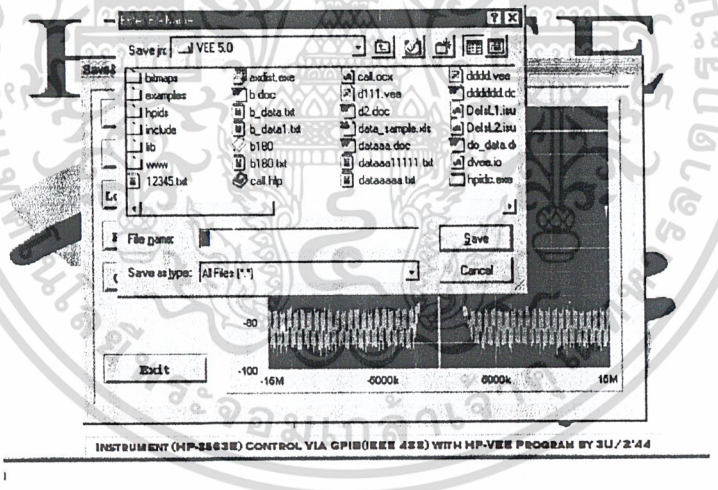


รูปที่ 5.3 โหมด Save & Print Graph



รูปที่ 5.4 สัญญาณที่ได้จากสเปคตรัมอนาไลเซอร์

โหมดนี้ผู้ใช้สามารถที่จะสั่งให้โปรแกรมดึงผลของข้อมูลมาแสดงบนคอมพิวเตอร์ได้ โดยการจำลองพล็อตรูปของสัญญาณจากเครื่องสเปคตรัมอนาไลเซอร์บนคอมพิวเตอร์ และสามารถเลือกที่

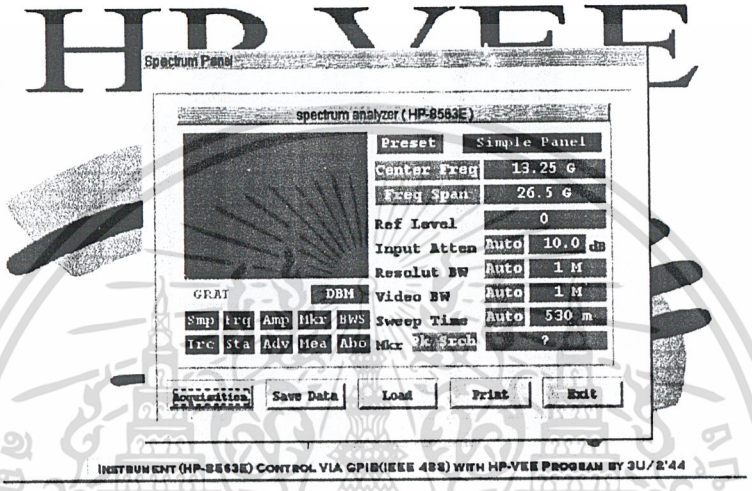


รูปที่ 5.5 การบันทึกข้อมูลสัญญาณ

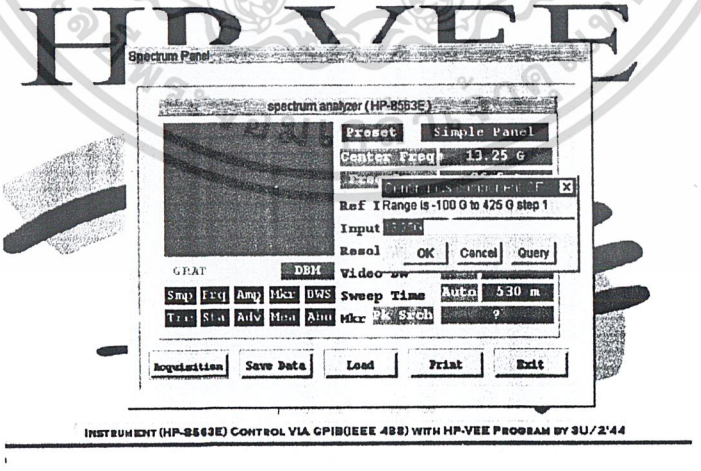
จะบันทึกข้อมูลของสัญญาณไว้ในรูปของแฟ้มเอกสารต่างๆได้ หรือหากต้องการที่จะพิมพ์สัญญาณผ่านเครื่องพิมพ์ก็สามารถทำได้ในโหมดนี้เช่นกัน

- **Spectrum Panel**

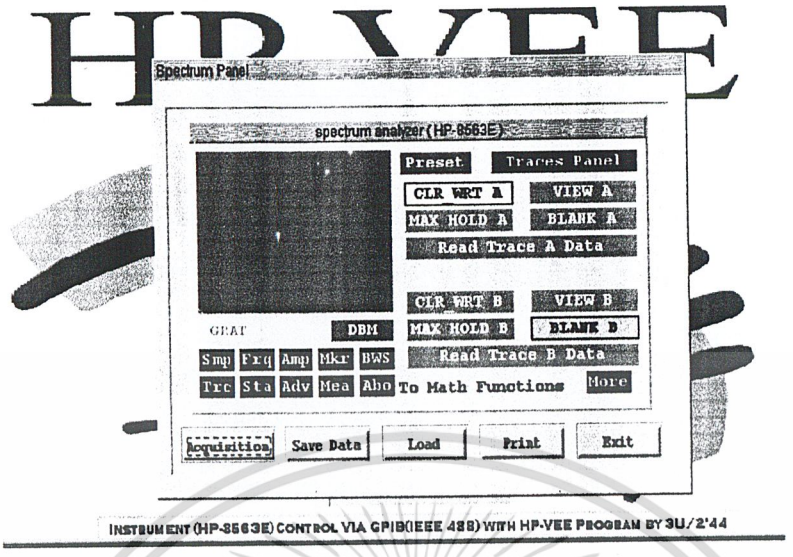
คือโหมดของการควบคุมและสั่งงานสเปคตรัมอนาไลเซอร์ โดยมีฟังก์ชันสำหรับใช้ควบคุมเหมือนกับตัวสเปคตรัมอนาไลเซอร์เอง ในโหมดนี้เราสามารถทำการตั้งหรือเซตค่าต่างๆของสเปคตรัมได้ ซึ่งในรูปที่ 5.4 และรูปที่ 5.5 นั้นแสดงอยู่ในฟังก์ชัน Smp ซึ่งเป็นส่วนของการตั้งค่าต่างๆเช่น Center Freq ซึ่งจากรูปป้อนค่าอยู่ที่ 13.25G แบะ Freq Span ตั้งค่าที่ 26.5G



รูปที่ 5.4 โหมด Spectrum Panel

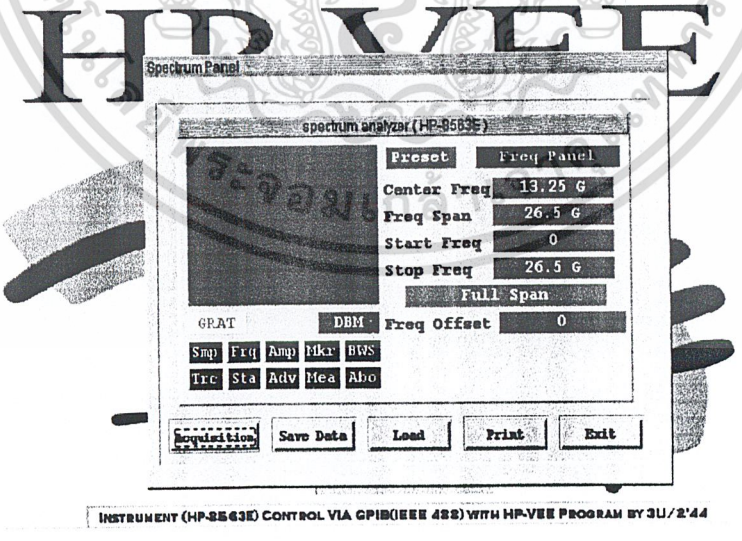


รูปที่ 5.5 การป้อนค่าในโหมด Spectrum Panel



รูปที่ 5.6 ฟังก์ชัน Traces Panel

รูปที่ 5.6 เป็นการแสดงในฟังก์ชัน Traces Panel ซึ่งมีฟังก์ชันในการใช้งานต่างๆ หาก ต้องการเข้ามาในฟังก์ชันของการทำงานนี้นั้น สามารถเลือกเข้ามาได้โดยการคลิกเมาส์ที่ Trc

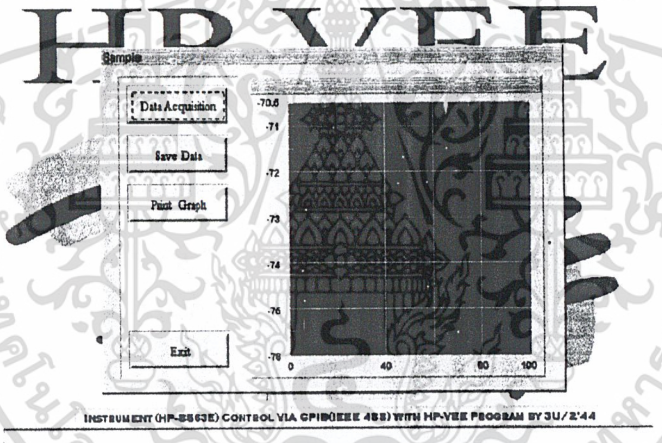


รูปที่ 5.7 ฟังก์ชัน Freq Panel

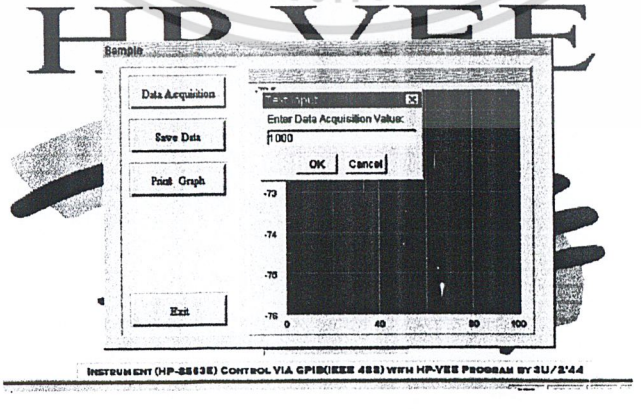
รูปที่ 5.7 เป็นการแสดงในฟังก์ชัน Freq Panel ซึ่งมีฟังก์ชันในการใช้งานต่างๆ หากต้องการเข้ามาในฟังก์ชันของการทำงานนี้นั้น สามารถเลือกเข้ามาได้โดยการคลิกเมาส์ที่ Frq ซึ่งในฟังก์ชันนี้นั้นเราสามารถที่จะตั้งค่าของความถี่ในรูปแบบของการวัดต่างๆได้ตามต้องการ

- **Data Acquisition Panel**

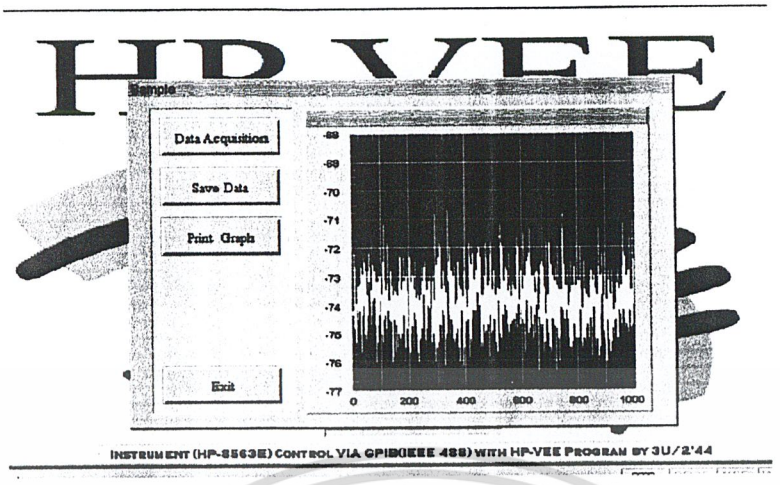
เป็นโหมดที่ได้ทำการโปรแกรมขึ้นเมื่อเพื่อประยุกต์และเพิ่มขีดความสามารถในการใช้งานเครื่องสเปคตรัมอนาไลเซอร์ ซึ่งโดยปกติแล้วเครื่องสเปกตรัมอนาไลเซอร์จะแสดงผลในรูปสัญญาณอยู่ในรูปของโมนความถี่ ซึ่งในโหมนี้นั้นเราได้ทำการประยุกต์ให้โปรแกรมประมวลผลและทำการวิเคราะห์และแสดงผลของสัญญาณในรูปโดเมนเวลา และในโหมดนี้เราสามารถนำข้อมูล หรือ ค่าที่ได้จากประมวลผลมาบันทึก เป็นไฟล์ในรูปแบบเอกสาร หรือจะแสดงผลเป็นกราฟได้ตามต้องการ โดยเราสามารถที่จะระบุได้ว่าต้องการให้ โปรแกรมทำการประมวลผลเป็นจำนวนทั้งหมดกี่ค่า



รูปที่ 5.5 โหมด Data Acquisition

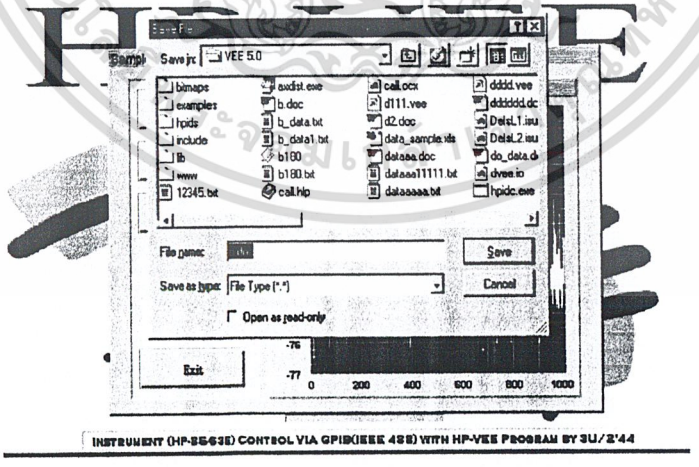


รูปที่ 5.6 การป้อนค่าในโหมด Data Acquisition Panel



รูปที่ 5.7 กราฟที่ได้จากการประมวลผล

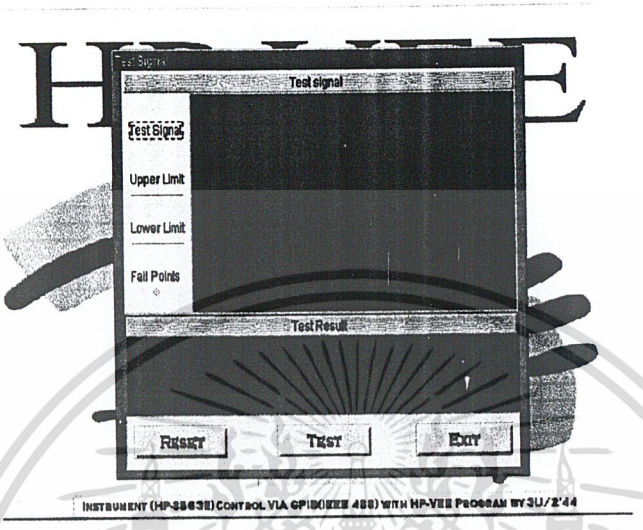
สำหรับรูปที่ 5.6 นั้นเป็นตัวอย่างของการป้อนค่าในโหมด Data Acquisition Panel จากภาพนั้นเป็นการป้อนค่าจำนวน 1000 ค่า เมื่อทำการป้อนค่าเพื่อให้โปรแกรมทำการประมวลผลแล้วนั้น โปรแกรมจะทำการประมวลผลวิเคราะห์และแสดงผลในโดเมนเวลาจำนวน 1000 ค่า ซึ่งภาพหรือกราฟของสัญญาณแสดงให้เห็นดังรูปที่ 5.7 หากเราต้องการเก็บบันทึกผลของการทดลองก็สามารถทำได้ด้วยการเลือกที่ฟังก์ชัน Save Data ดังแสดงในรูปที่ 5.8 และหากต้องการสั่งพิมพ์ผ่านเครื่องพิมพ์ให้เลือกที่ฟังก์ชันPrint Graph โปรแกรมก็จะสั่งพิมพ์ภาพของสัญญาณผ่านเครื่องพิมพ์



รูปที่ 5.8 การบันทึกผลของการทดลอง

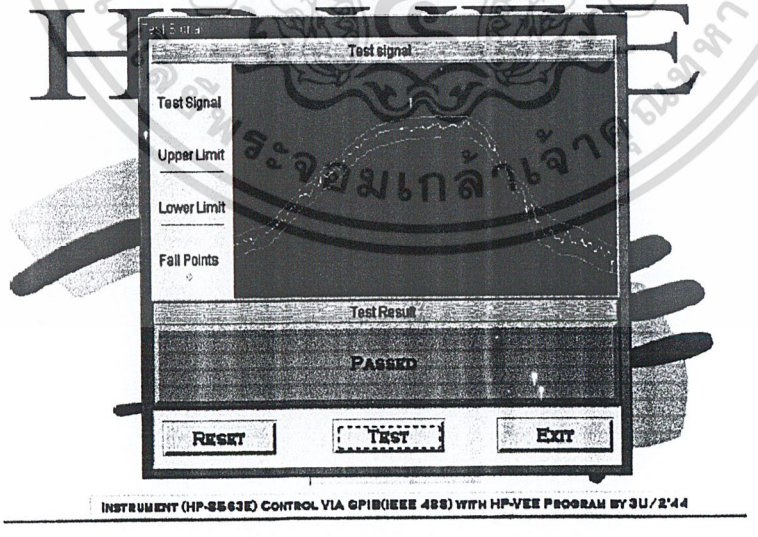
- **Test Panel**

คือโหมดของการตรวจสอบคุณลักษณะของสัญญาณ หรือโหมดของการตรวจสอบคุณภาพของสัญญาณที่ได้ทำการวัดว่าอยู่ในขอบเขตตามที่ต้องการหรือที่ได้กำหนดไว้หรือไม่ ซึ่งในโหมดนี้ได้

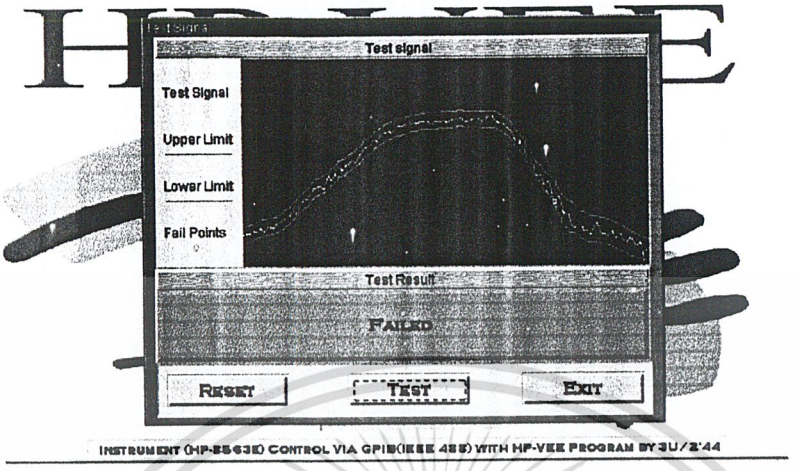


รูปที่ 5.9 โหมด Test Panel

แสดงให้เห็นว่าการควบคุมเครื่องมือวัดผ่านการโปรแกรมนั้น สามารถที่จะเพิ่มขีดความสามารถต่างๆในการใช้งานเครื่องมือวัดได้จริงซึ่งแสดงดังรูปที่ 5.9



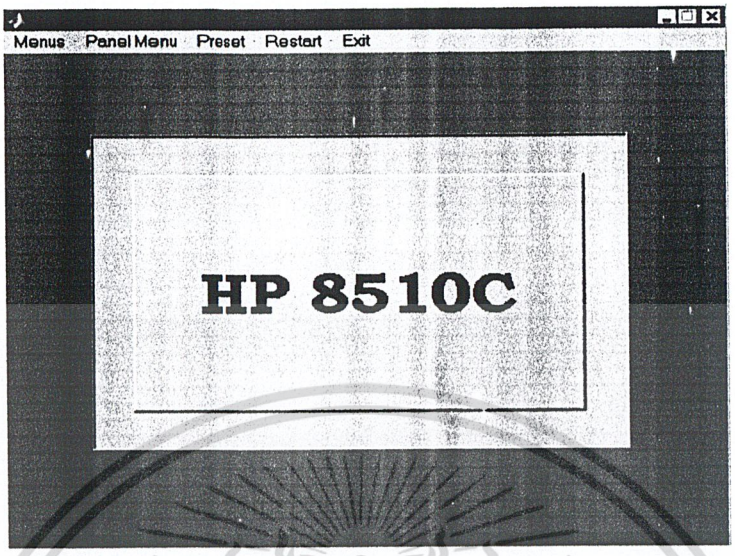
รูปที่ 5.10 ผลจากตรวจสอบ 'PASSED'



รูปที่ 5.11 ผลจากตรวจสอบ 'FAILED'

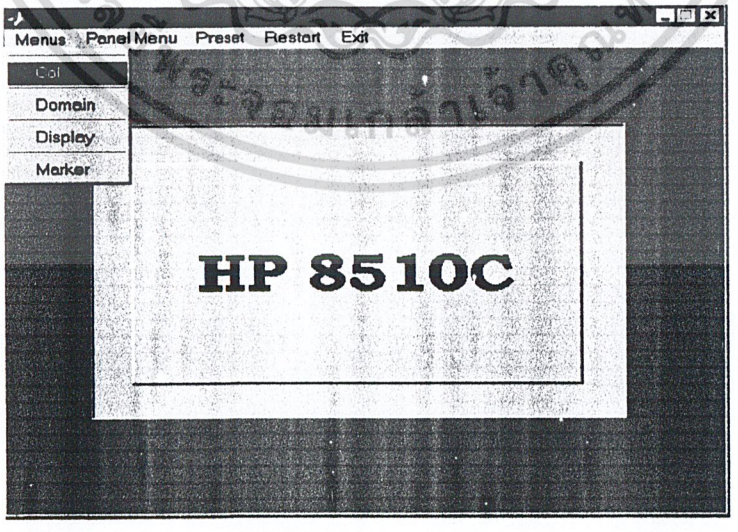
จากรูปที่ 5.10 และรูปที่ 5.11 แสดงให้เห็นถึงผลของการตรวจสอบอุปกรณ์แบนด์พาสว่าอยู่ในขอบเขตที่ต้องการหรือไม่ สำหรับรูปที่ 5.10 แสดงให้เห็นว่าสัญญาณที่นำเข้ามาเพื่อตรวจสอบนั้นผ่านการตรวจสอบโดยโปรแกรมได้ประมวลผลและแสดงผลออกมาด้วยคำว่า 'PASSED' และรูปที่ 5.11 ได้แสดงให้ทราบว่าสัญญาณที่นำมาตรวจสอบนั้นไม่ผ่านการตรวจสอบ โดยโปรแกรมได้ประมวลผล และแสดงผลด้วยคำว่า 'FAILED'

5.2 โปรแกรมควบคุม Network Analyzer ด้วย Matlab

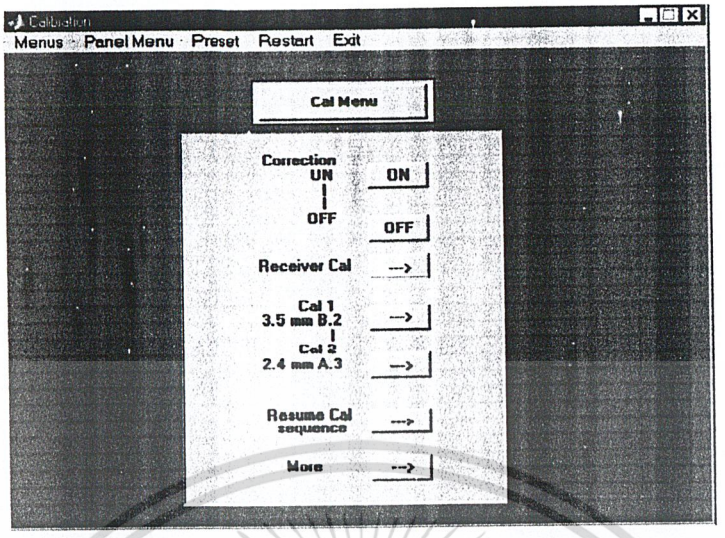


รูปที่ 5.12 การจำลอง Network Analyzer ด้วย Matlab

จากรูปที่ 5.12 แสดงลักษณะของโปรแกรมที่เป็นการจำลองเครื่องเนทเวิร์กอนาไลเซอร์ โดยจากรูปแสดงลักษณะของโปรแกรมเมื่อเริ่มการทำงาน โดยเราสามารถเลือกฟังก์ชันที่ต้องการจะใช้งานได้ที่เมนูบาร์ จากรูปจะมีเมนูคือ Menus, Panel Menu, Reset, Restart และ Exit ซึ่งในเมนูต่างๆของโปรแกรมก็จะมีฟังก์ชันย่อยๆให้เลือกในการใช้งานอยู่อีก จากรูปที่ 5.12 เลือกที่ Menus จะมีฟังก์ชันย่อยคือ Cal, Domain, Display, Marker

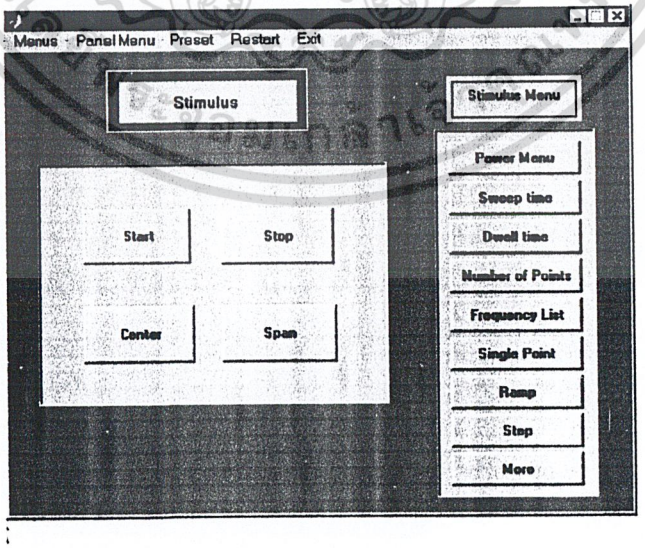


รูปที่ 5.13 เมนูของการใช้งานโปรแกรม



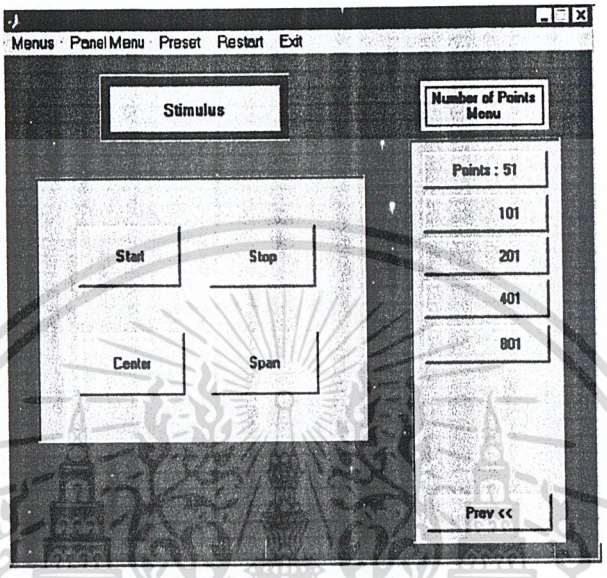
รูปที่ 5.14 โหมดของเมนูแคล

จากรูปที่ 5.14 แสดงให้เห็นถึงโหมดย่อยของการใช้งานโปรแกรม ซึ่งในโหมดนี้เป็นโหมดที่ใช้สำหรับการแคลิเบรทของเครื่องเนทเวิร์กอนาไลเซอร์ จากรูปจะเห็นว่าในโหมดนี้จะมีฟังก์ชันย่อยๆอยู่อีก 7 ฟังก์ชันที่ใช้ในการติดต่อกับเนทเวิร์กอนาไลเซอร์ ซึ่งเมื่อเราทำการเลือกใช้ในแต่ละฟังก์ชัน ก็จะมีฟังก์ชันย่อยๆของแต่ละฟังก์ชันออกมาให้เลือกอีกมากมาย ซึ่งในขณะที่ทำการเลือกโหมดของแต่ละฟังก์ชันบนคอมพิวเตอร์นั้น ทางเนทเวิร์กอนาไลเซอร์ก็ได้ทำงานตามคำสั่งของเราไปพร้อมๆกันด้วย



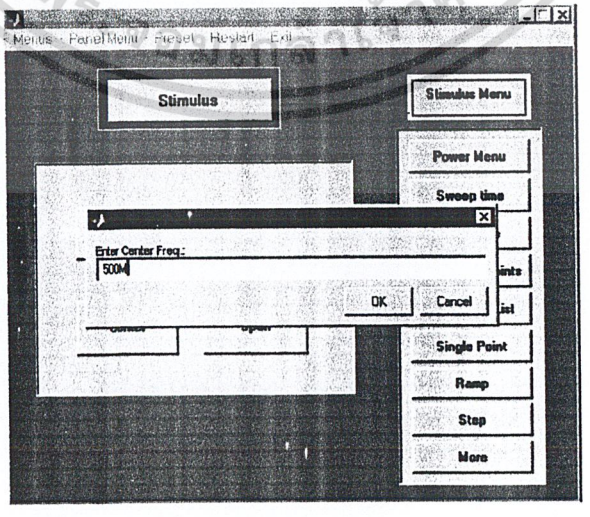
รูปที่ 5.15 การเรียกใช้เมนู Panel Menu

จากรูปที่ 5.15 แสดงการเรียกใช้เมนู Panel Menu ซึ่งรูปนี้เป็นฟังก์ชันของ Stimulus ลักษณะของโปรแกรมก็จะคล้ายหน้าต่างของโปรแกรมดังที่กล่าวมาแล้วคือจะมีฟังก์ชันเลือกใช้แล้วแต่ลักษณะของการใช้ฟังก์ชันนี้เช่นถ้าเราคลิ๊กเมาส์ที่ Number of Points โปรแกรมก็จะแสดงหน้าต่างของฟังก์ชันนี้ออกมาอยู่ที่ผู้ใช้ว่าจะเลือกใช้ในฟังก์ชั่นใดของโปรแกรม



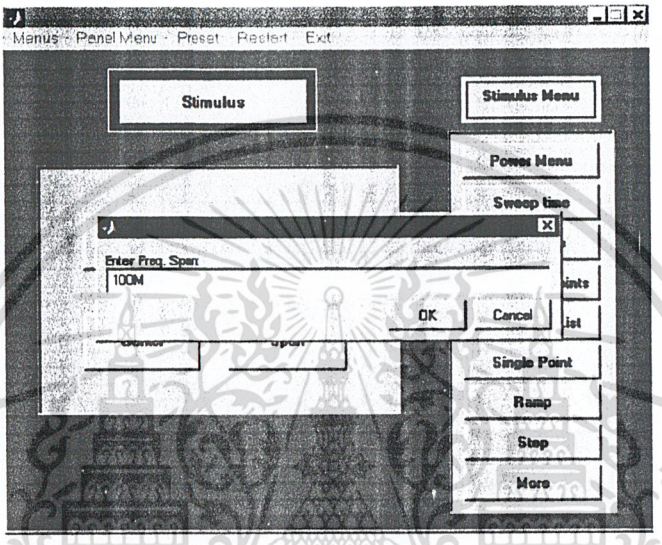
รูปที่ 5.16 ฟังก์ชัน Stimulus

จากรูปที่ 5.16 เป็นโหมดของ Stimulus ของเนทเวิร์กอนาไลเซอร์ ลักษณะการทำงานของโปรแกรม ผลที่ได้จากโปรแกรมก็คือการควบคุม Network Analyzer ด้วยโปรแกรมจะมีลักษณะการทำงานเหมือนกับการควบคุมจาก Network Analyzer เอง



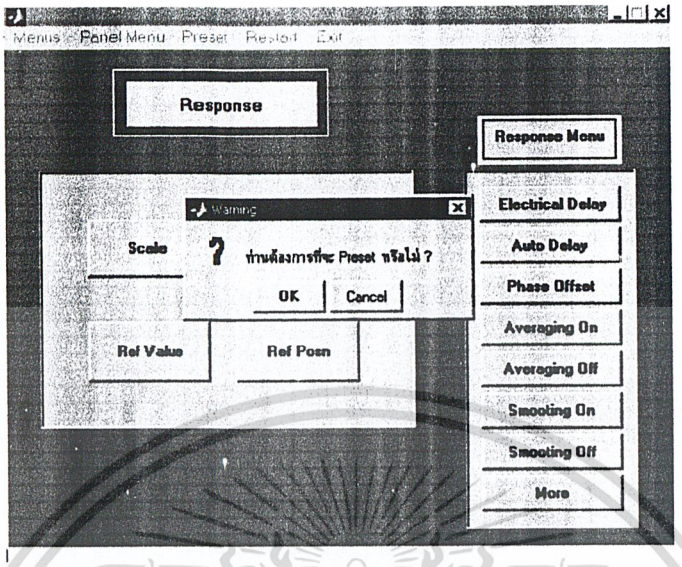
รูปที่ 5.17 ป้อนค่า Center Freq ใน Stimulus โหมด

จากรูปที่ 5.17 เป็นขั้นตอนของการป้อนค่าเซ็นเตอร์ฟรีเควนซี่ เมื่อเราทำการคลิกเมาส์เลือกที่ฟังก์ชัน “Center” โปรแกรมจะทำการสอบถามถึงค่าความถี่กลางที่เราต้องการตั้งค่าให้กับเนทเวิร์กอนาไลเซอร์ เพื่อทำการตั้งค่าความถี่ในโหมดของสติมูลัส (Stimulus) ในขั้นตอนนี้เราต้องใส่ค่าความถี่พร้อมทั้งหน่วยของความถี่เข้าไปเพื่อให้เครื่องเนทเวิร์กอนาไลเซอร์ทำการเก็บค่าที่ป้อนไว้



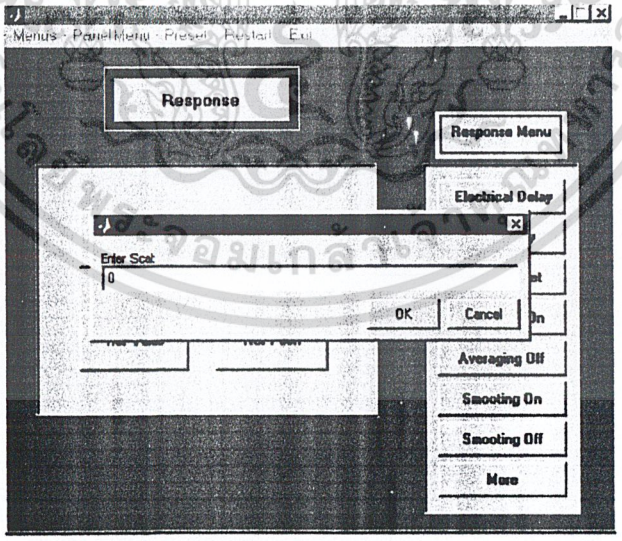
รูปที่ 5.18 ป้อนค่า Freq Span ใน Stimulus โหมด

จากรูปที่ 5.18 เมื่อเราทำการคลิกเมาส์เลือกที่ฟังก์ชัน “Span” โปรแกรมจะทำการสอบถามถึงค่าช่วงของความถี่ที่เราต้องการตั้งค่าให้กับเนทเวิร์กอนาไลเซอร์ เพื่อทำการตั้งค่าความถี่ในโหมดของสติมูลัส (Stimulus) ลักษณะของโปรแกรมในโหมดของสติมูลัส (Stimulus) นี้ ยังมีฟังก์ชันที่เราสามารถทำการเซทหรือตั้งค่าอีกคือ “Start” และ “Stop” ซึ่งการทำงานและการป้อนค่าของความถี่ก็ทำคล้ายกับการป้อนค่าของความถี่กลาง และ การป้อนค่าของช่วงความถี่ โดยหากเราต้องการที่จะตั้งค่าใดๆ หรือต้องการสั่งงานให้เครื่องเนทเวิร์กรับค่าใดๆแล้วนั้นเราสามารถทำได้โดยการเลือกฟังก์ชันของการทำงานจากรูปเป็นการป้อนค่า Freq Span ที่ 100M



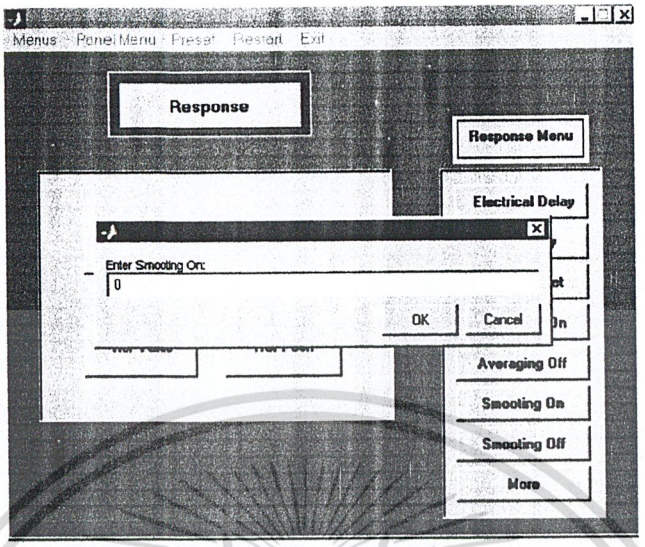
รูปที่ 5.19 ฟังก์ชัน Response

จากรูปที่ 5.19 เป็นรูปที่แสดงการใช้งานเนทเวิร์กอนาไลเซอร์ ในโหมดของเรสปอนซ์ (Response) ซึ่งจากรูปโปรแกรมได้ทำการเตือนเราว่าต้องการพรีเซทใหม่หรือไม่



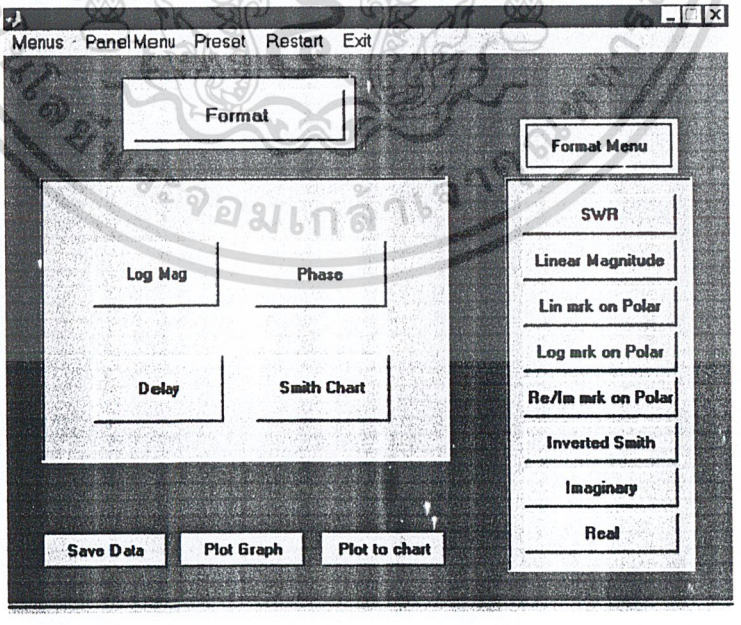
รูปที่ 5.20 การป้อนค่า Scale ในโหมด Response

จากรูปที่ 5.20 เป็นการป้อนค่าของสเกลให้กับโหมดเรสปอนซ์ (Response) โดยโปรแกรมจะทำการสอบถามถึงตัวเลขที่เราต้องการตั้งค่าสเกล



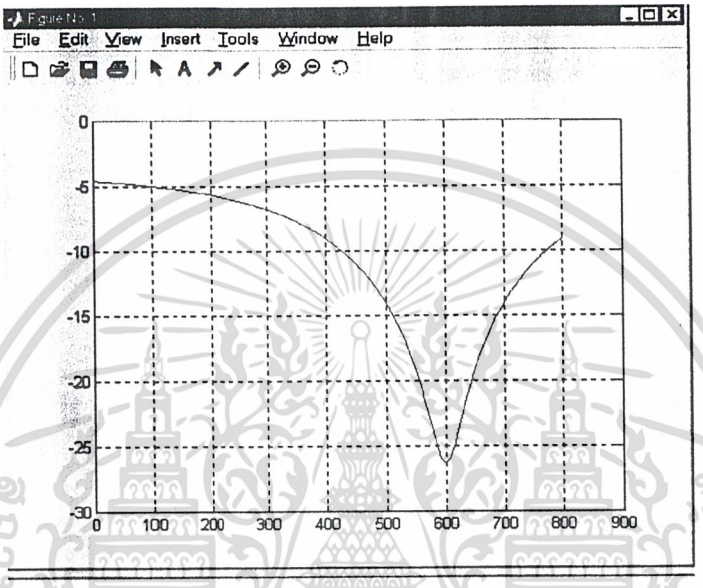
รูปที่ 5.21 การป้อนค่า Smooting On ในโหมด Response

จากรูปที่ 5.21 เป็นการป้อนค่าของสมูทติ้งออน (Smooting On) ให้กับโหมดเรสปอนซ์ (Response) โดยโปรแกรมจะทำการสอบถามถึงตัวเลขที่เราต้องการตั้งค่าสมูทติ้งออน



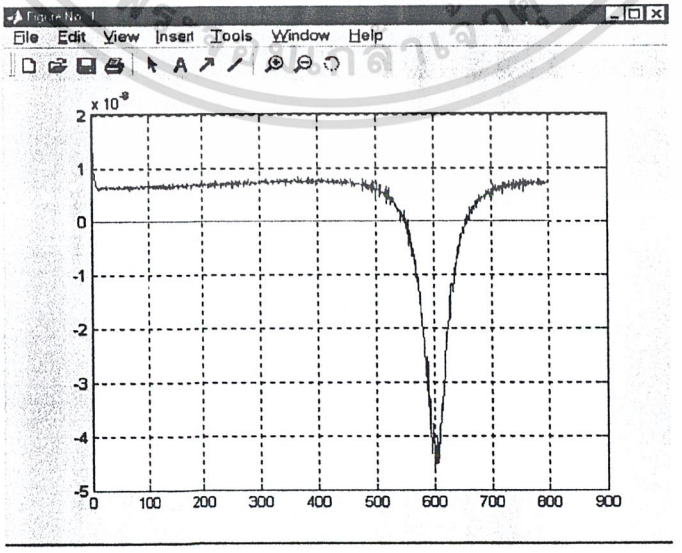
รูปที่ 5.22 ฟังก์ชัน Format Menu

จากรูปที่ 5.22 โปรแกรมได้แสดงสถานะในโหมดของ Format Menu ซึ่งเป็นโหมดของการสั่งงานที่เราสามารถจะสั่งงานให้เครื่องเนทเวิร์กอนาไลเซอร์ในรูปแบบต่างๆ โดยการแสดงผลต่างๆนั้นจะเป็นในรูปของกราฟสัญญาณ ซึ่งในโหมดนี้นั้นเราได้ทำการโปรแกรมเพื่อเพิ่มประสิทธิภาพในการใช้งานเครื่องมือวัดขึ้น โดยเราสามารถที่จะทำการบันทึกผลของการทดลองได้ในรูปของเอกสาร หรือการพิมพ์ผ่านเครื่องพิมพ์



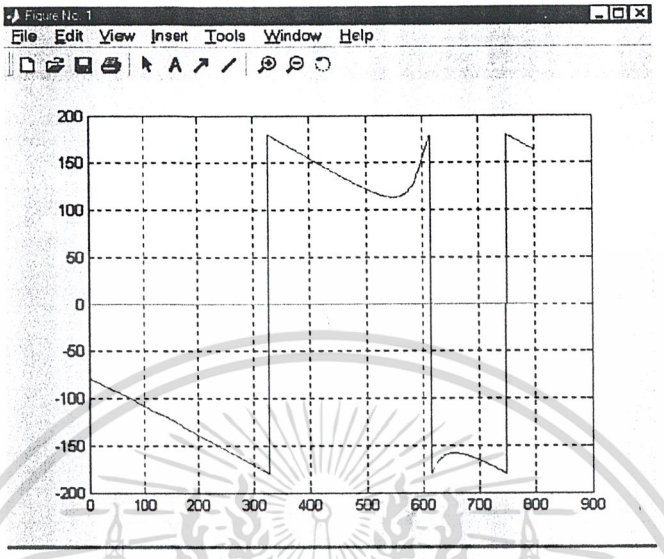
รูปที่ 5.23 กราฟสัญญาณใน Format ของ Log Mag

จากรูปที่ 5.23 เป็นรูปของกราฟสัญญาณที่ได้จากการประมวลผล โดยอยู่ใน Format ของ Log Mag



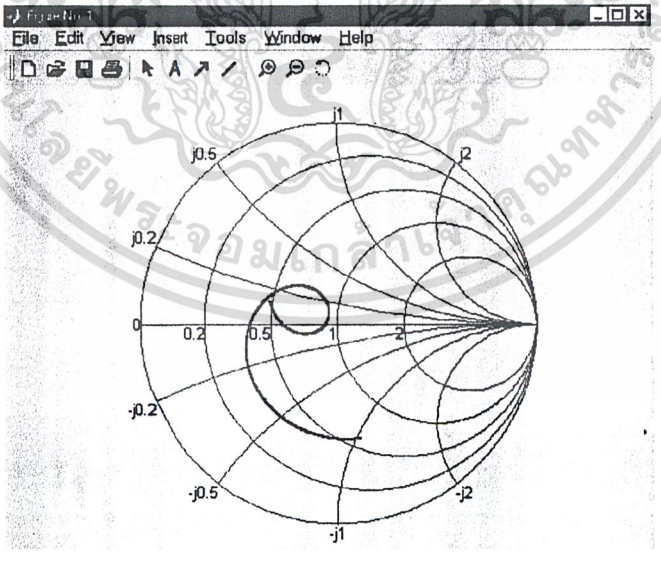
รูปที่ 5.24 กราฟสัญญาณใน Format ของ Delay

จากรูปที่ 5.24 เป็นรูปของกราฟสัญญาณที่ได้จากการประมวลผล โดยอยู่ใน Format ของ Delay



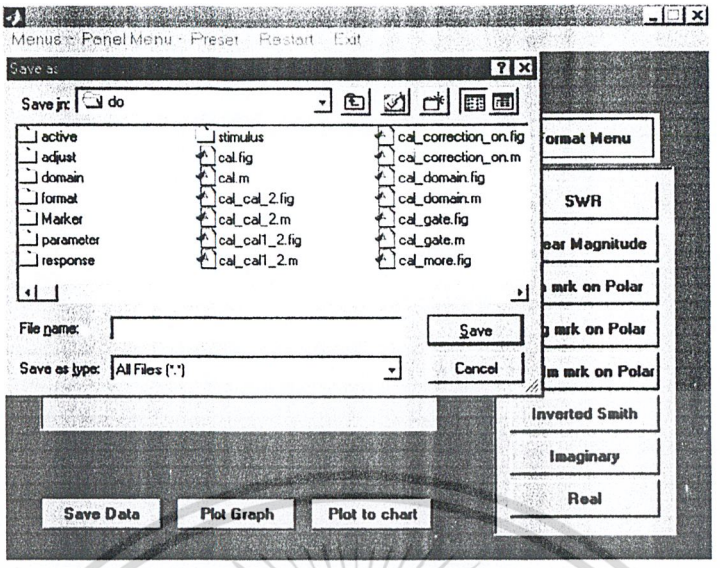
รูปที่ 5.25 กราฟสัญญาณใน Format ของ Phase

จากรูปที่ 5.25 เป็นรูปของกราฟสัญญาณที่ได้จากการประมวลผล โดยอยู่ใน Format ของ Phase



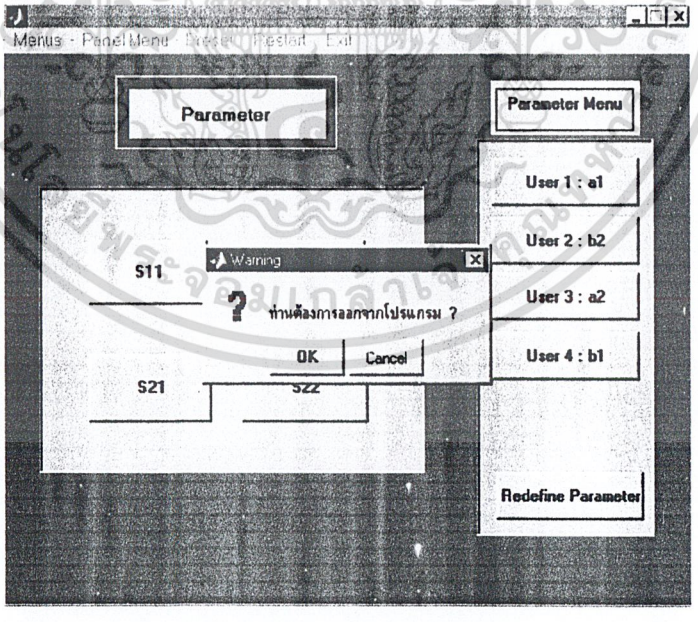
รูปที่ 5.26 กราฟสัญญาณใน Format ของ Smith Chart

จากรูปที่ 5.26 เป็นรูปของกราฟสัญญาณที่ได้จากการประมวลผล โดยอยู่ใน Format ของ Smith Chart



รูปที่ 5.27 การบันทึกผลการทดลอง

จากรูปที่ 5.27 เป็นการบันทึกผลของการทดลองในโหมดของ Format Menu ซึ่งเราสามารถบันทึกเป็นเอกสาร และสามารถเรียกเอกสารที่บันทึกไว้นั้นออกมาแสดงและพล็อตเป็นกราฟของสัญญาณได้อีกครั้ง



รูปที่ 5.28 การออกจาโปรแกรม

เมื่อเราต้องการออกจากการใช้งานโปรแกรมเราสามารถเลือกได้ที่เมนูบาร์ Exit ซึ่งเมื่อเราเลือกที่เมนู Exit แล้วนั้นโปรแกรมจะทำการเตือนว่าเราต้องการออกจากโปรแกรมหรือไม่

บทที่ 6
บทสรุปและวิจารณ์

บทสรุป

จากการที่ได้ทำการศึกษาและค้นคว้าหาข้อมูล ในการโปรแกรม โดยใช้โปรแกรม HP VEE เพื่อใช้ในการควบคุม (Control) เครื่องมือวัด (Instrument) เราพบว่าการโปรแกรมด้วย HP VEE มีความสะดวกในการติดต่อกับเครื่องมือวัด โดยการอินเตอร์เฟซผ่านการ์ด GPIB ซึ่งเราสามารถติดต่อกับเครื่องมือวัดได้เครื่องมือวัดที่เรานำมาอินเตอร์เฟซนี้คือเครื่องสเปกตรัมอนาไลซ์เซอร์ (Spectrum Analyzer) โดยในขั้นต้นเราได้ทำการทดลองด้วยการเขียนโปรแกรมเพื่อให้เราสามารถควบคุมเครื่องมือวัดชนิดนี้ได้ โดยเราสามารถป้อนค่าลงในโปรแกรมเพื่อจำลองการทดลอง และเราสามารถควบคุมเครื่องมือวัดนี้ได้ตามฟังก์ชันที่เราเขียนไป และยังสามารถให้คอมพิวเตอร์แสดงผลของข้อมูลที่ได้จากสเปคตรัมออกมาได้ ในรูปของกราฟ หรือ การพิมพ์ผ่านทางเครื่องพิมพ์

ในส่วนของการเขียนโปรแกรมเพื่อควบคุมเครื่องเนทเวิร์กอนาไลเซอร์ด้วยโปรแกรม Matlab นั้นได้มีการเขียนโปรแกรมขึ้นเพื่อทำการจลองเครื่องเนทเวิร์กอนาไลเซอร์ลงบนคอมพิวเตอร์ ซึ่งเราสามารถเรียกใช้ฟังก์ชันในเนทเวิร์กอนาไลเซอร์ผ่านคอมพิวเตอร์ได้ โดยไม่ต้องไปสั่งงานโดยตรงจากตัวเครื่องเนทเวิร์ก

ปัญหา

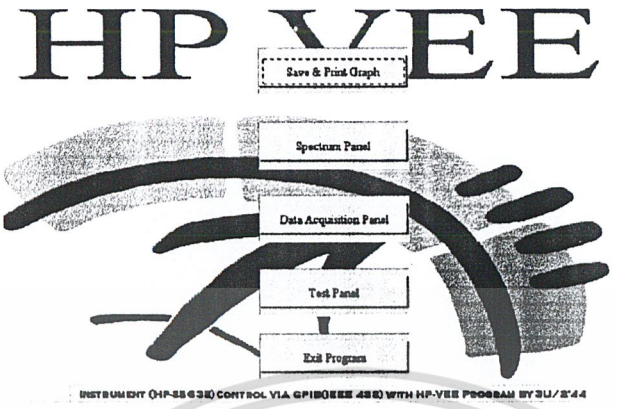
จากการทำงานปัญหาที่พบคือ ในการศึกษาค้นคว้า โปรแกรม HP VEE เพื่อให้เข้าใจในการโปรแกรม เป็นไปด้วยความลำบาก เนื่องด้วยเป็นโปรแกรมที่ถูกสร้างขึ้นในเบื้องต้นเพื่อควบคุมเครื่องมือวัด ซึ่งทำให้โปรแกรมนี้มิได้แพร่หลายนัก ข้อมูลและตำราจึงหาได้ยาก และมีอย่างจำกัด จึงทำให้ช่วงเวลาโดยส่วนใหญ่ก็คือ การศึกษาโปรแกรม สำหรับปัญหาในการเขียนโปรแกรมเพื่อควบคุมเครื่องเนทเวิร์กอนาไลเซอร์นั้น เราพบว่าในบางฟังก์ชันที่ใช้ในเครื่องเนทเวิร์ก เราไม่พบคำสั่งในการเขียนโปรแกรมเพื่อควบคุมเนทเวิร์กจากแมนนวลหรือคู่มือของเนทเวิร์ก

แนวทางการพัฒนา

ในการทดลองเพื่อควบคุมเครื่องมือวัดทั้ง 2 ชนิดไม่ว่าจะเป็นสเปคตรัมอนาไลเซอร์ หรือเนทเวิร์กอนาไลเซอร์ ผ่านคอมพิวเตอร์นั้น เราพบว่าเครื่องมือวัดทั้ง 2 ชนิดนั้นยังมีฟังก์ชันอีกมากมายในการใช้งาน ซึ่งจะเป็นแนวทางในการพัฒนาต่อไปได้

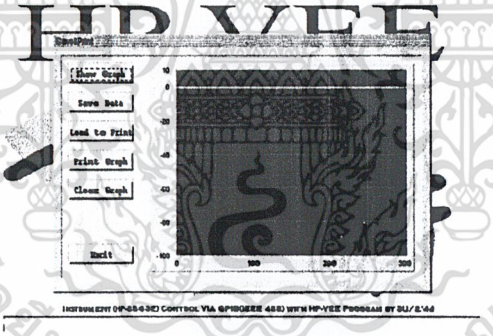
ภาคผนวก

คู่มือการใช้งานโปรแกรมควบคุม Spectrum Analyzer



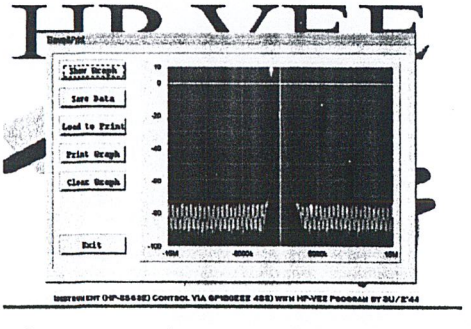
รูปที่ 1 เมนูหลักของโปรแกรม

Save & Print Graph เป็นเมนูที่สามารถบันทึกและดึงสัญญาณจากสเปกตรัมมาแสดงผลได้ เมื่อทำการเลือกมาที่เมนูแล้วโปรแกรมจะแสดงผลดังรูปที่ 2



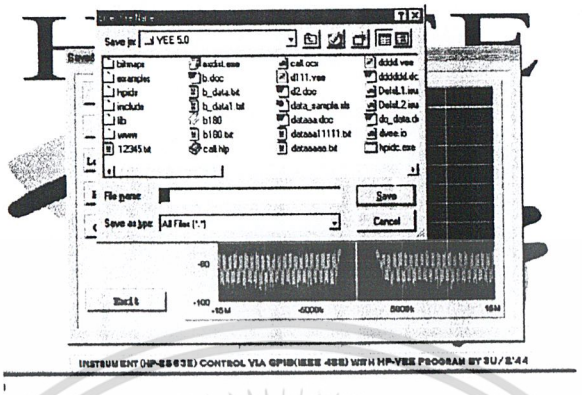
รูปที่ 2 Save&Print

Show Graph ใช้สำหรับเลือกเมื่อต้องการดึงสัญญาณจากสเปกตรัมมาแสดงผลบนคอมพิวเตอร์ และเมื่อทำการเลือกที่เมนูนี้แล้วจะเป็นดังรูปที่ 3



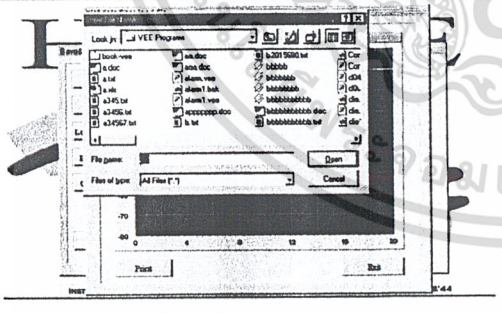
รูปที่ 3 สัญญาณที่แสดงบนคอมพิวเตอร์

Save Data ใช้สำหรับเลือกเมื่อต้องการบันทึกผลของสัญญาณจากสเปกตรัม เมื่อทำการเลือกที่เมนูนี้แล้วนั้นสามาถบันทึกเป็นแฟ้มเอกสารได้ดังรูปที่ 4

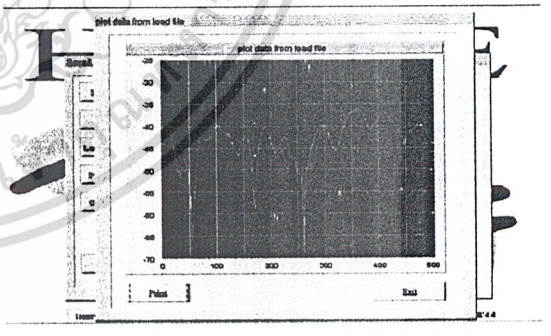


รูปที่ 4 การบันทึกผล

Load to Print ใช้สำหรับเลือกเมื่อต้องการเรียกแฟ้มเอกสารที่ได้ทำการบันทึกผลของสัญญาณจากสเปกตรัมมาพล็อตเป็นกราฟการเรียกเอกสารแสดงดังรูปที่ 5 ซึ่งเราสามารถที่จะบันทึกแฟ้มหรือเอกสารนี้ได้รูปแบบต่างๆ ขึ้นอยู่กับความต้องการของผู้ใช้งาน อาธิเช่นบันทึกในรูปของเอกสารที่มีนามสกุล .txt .doc และเมื่อทำการเรียกเอกสารมาแสดงผลเพื่อพล็อตเป็นกราฟแล้วผลของโปรแกรมจะแสดงดังรูปที่ 6



รูปที่ 5 การเรียกแฟ้มเอกสารมาพล็อต



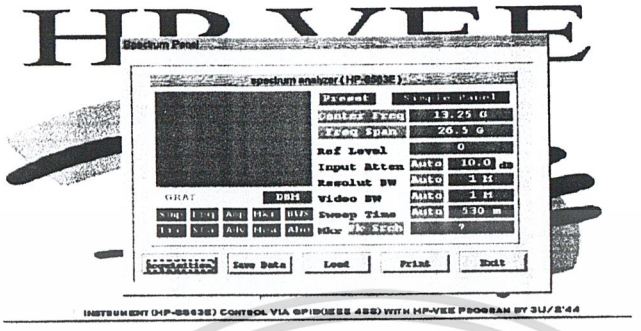
รูปที่ 6 สัญญาณที่ได้

Print Graph เมนูสำหรับเลือกเมื่อต้องการที่จะพิมพ์ภาพสัญญาณผ่านเครื่องพิมพ์

Clear Graph เมนูสำหรับเลือกเมื่อต้องการที่จะเคลียร์ภาพของสัญญาณบนคอมพิวเตอร์

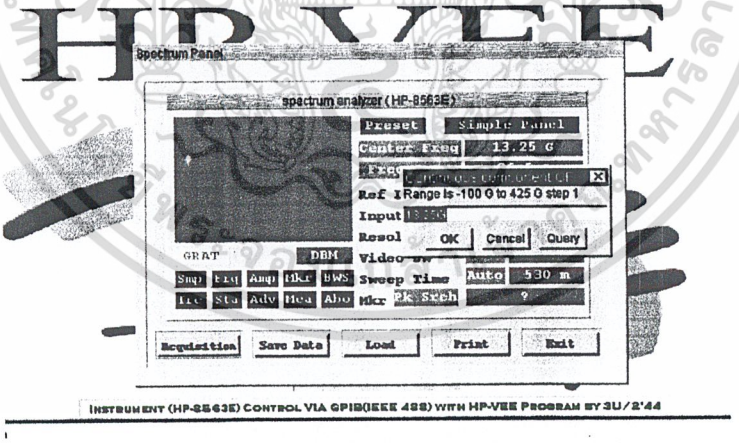
Exit เมนูสำหรับเลือกเมื่อต้องการออกสู่เมนูหลัก

จากเมนูหลักเมื่อเลือกเมนูนี้ | Spectrum Panel | โปรแกรมจะแสดงผลดังรูปที่ 7



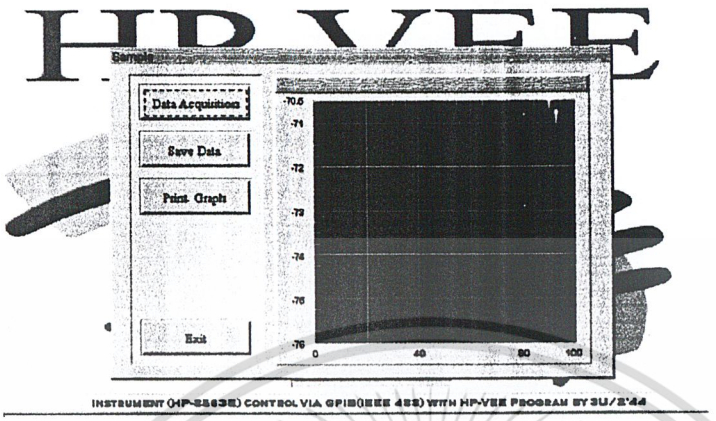
รูปที่ 7 เมนู Spectrum Panel

ซึ่งจากรูปที่ 7 ในเมนูนี้นั้นจะมีฟังก์ชันของการสั่งงานอยู่มากมายซึ่งเมื่อเราต้องการที่จะตั้งค่า Center Freq แล้วนั้นเราสามารถทำได้โดยเลือกที่ Center Freq ซึ่งเราสามารถป้อนค่าของความที่ได้โดยในการป้อนค่าของความถี่นั้นจะต้องมีหน่วยของความถี่ตามด้วยทุกครั้งเช่น เมื่อต้องการป้อนความถี่ที่ 13.25 GHz ดังแสดงตัวอย่างดังรูปที่ 8



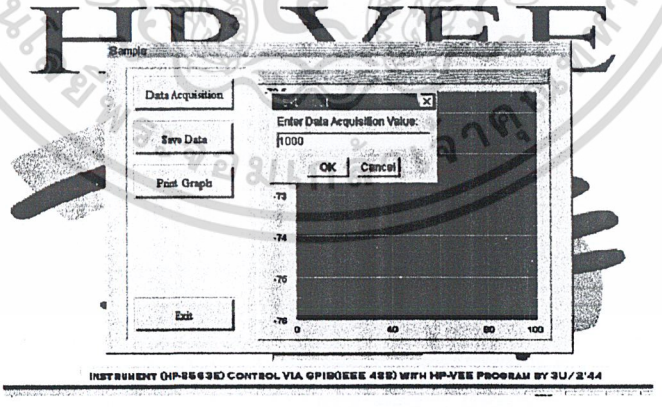
รูปที่ 8 การป้อนค่าความถี่

จากเมนูหลักเมื่อเลือกเมนูนี้ | Data Acquisition Panel | โปรแกรมจะแสดงผลดังรูปที่ 9



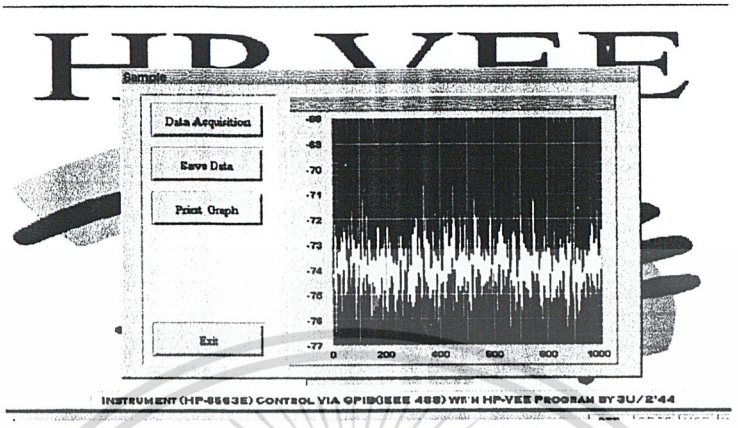
รูปที่ 9 เมนู Data Acquisition

จากรูปที่ 9 เมื่อต้องการที่จะทำการวิเคราะห์สัญญาณในเมนูนี้นั้นให้เลือกที่ | Data Acquisition | โดยเมื่อเลือกที่เมนูนี้แล้วนั้นผลของโปรแกรมจะเป็นดังรูปที่ 10 ซึ่งโปรแกรมได้ทำการเรียกให้เราทำการป้อนค่าของ Data Acquisition ซึ่งเป็นจะเป็นจำนวนใดนั้นขึ้นอยู่กับผู้ใช้ว่าต้องการให้ประมวลผลไปในจำนวนเท่าใด ซึ่งจากรูปที่ 10 นั้นเป็นตัวอย่างของการป้อนค่าจำนวน 1000 ค่า



รูปที่ 10 การป้อนค่าที่ต้องการประมวลผล

สำหรับรูปที่ 11 เป็นผลจากการประมวลผลของคอมพิวเตอร์ตามที่ได้สั่งการให้ประมวลผลออกมาจำนวน 1000 ค่าโดยภาพของสัญญาณเป็นดังรูป



รูปที่ 11 ภาพสัญญาณของ Data Acquisition

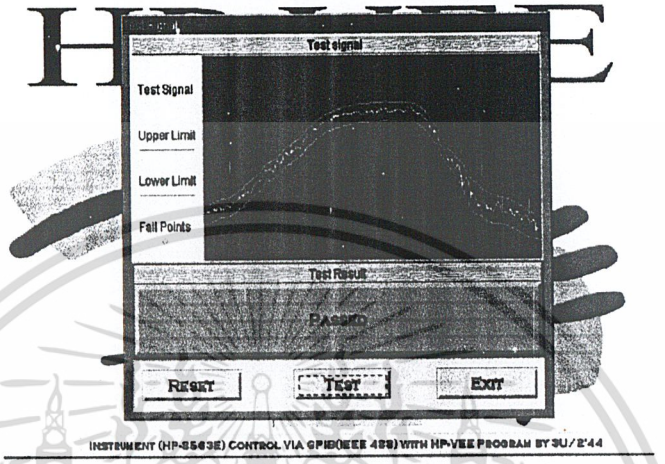
Save Data เมนูสำหรับเลือกเมื่อต้องการเก็บบันทึกผลการของ Data Acquisition

Print Graph เมนูสำหรับเลือกเมื่อต้องการที่จะพิมพ์ภาพของสัญญาณผ่านเครื่องพิมพ์

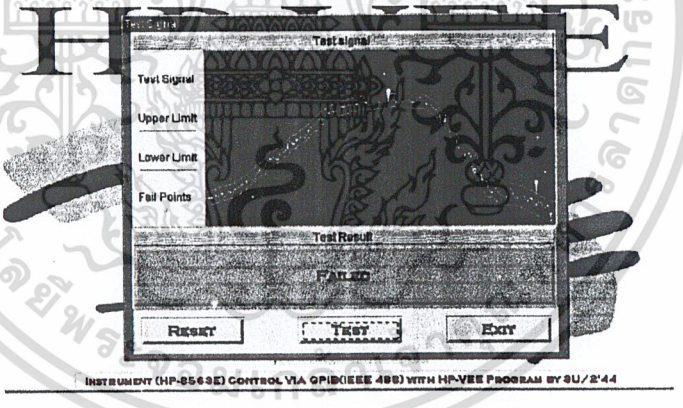
จากเมนูหลักเมื่อเลือกเมนูนี้ Test Panel โปรแกรมจะแสดงผลดังรูปที่ 12

รูปที่ 12 เมนู Test Panel

เมื่อต้องการที่จะตรวจสอบสัญญาณที่เข้ามาในเครื่องสเปกตรัมอนาไลเซอร์ ว่าอยู่ในขอบเขตที่ต้องการหรือไม่ สามารถทำได้โดยเลือกที่ **TEST** ซึ่งผลของโปรแกรมจะมีใน 2 ลักษณะคือ Passed และ Failed ซึ่งก็คือการตรวจสอบ ผ่าน และ ไม่ผ่านนั้นเองโดยได้ผลของการตรวจสอบนั้นได้แสดงดังรูปที่ 13 และรูปที่ 14



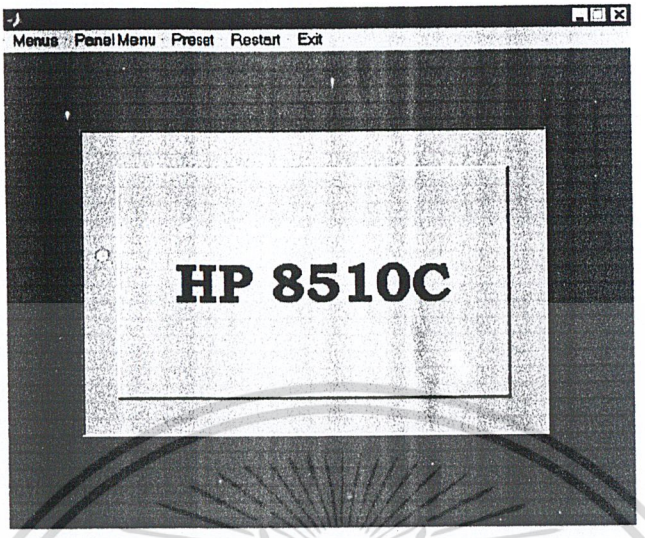
รูปที่ 13 ผลการตรวจสอบ PASSED



รูปที่ 14 ผลการตรวจสอบ FAILED

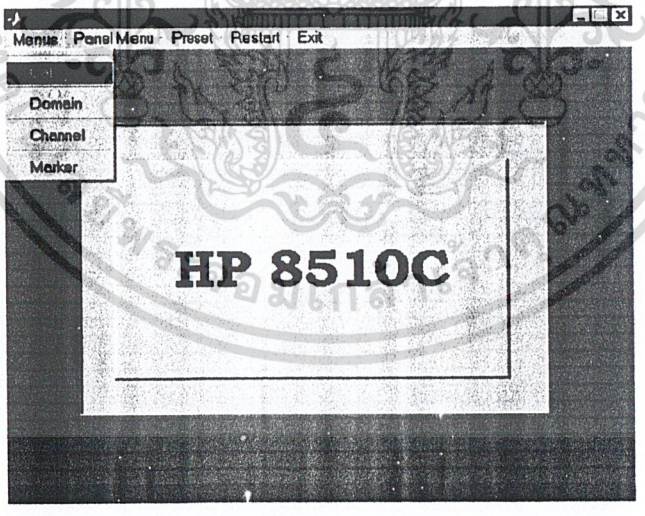
RESET เมนูสำหรับเลือกเมื่อต้องการที่จะรีเซทเพื่อทำงานใหม่

EXIT เมนูสำหรับเลือกเมื่อต้องการออกสู่เมนูหลัก



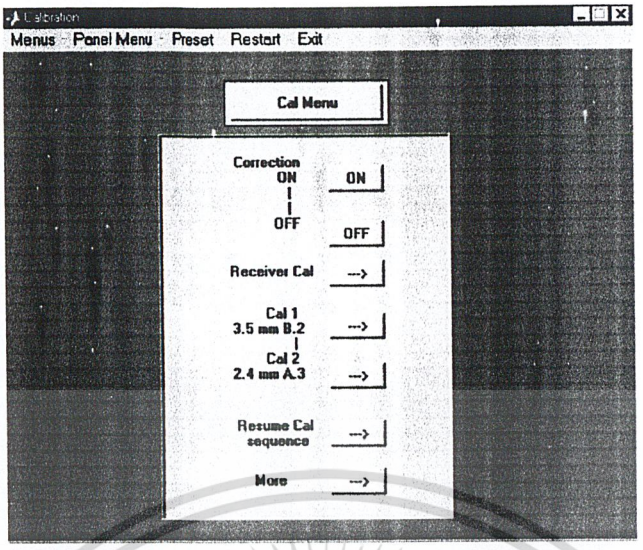
รูปที่ 16 เมนูหลักของโปรแกรม

จากรูปที่ 16 เป็นเมนูหลักของโปรแกรมซึ่งจะประกอบด้วยเมนูบาร์ที่สามารถเลือกลักษณะของการใช้งานได้



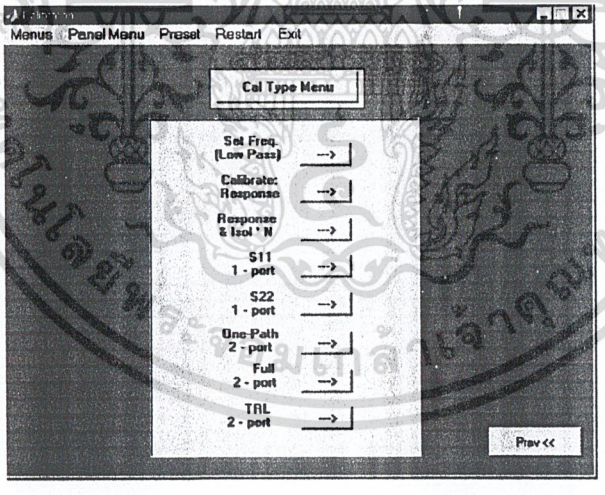
รูปที่ 17 การเรียกใช้เมนูบาร์

จากรูปที่ 17 เป็นการเลือกใช้เมนูบาร์ โดยจะมีเมนูย่อยสำหรับเลือกใช้งานตามลักษณะหรือตามความต้องการของผู้ใช้ต้องการ



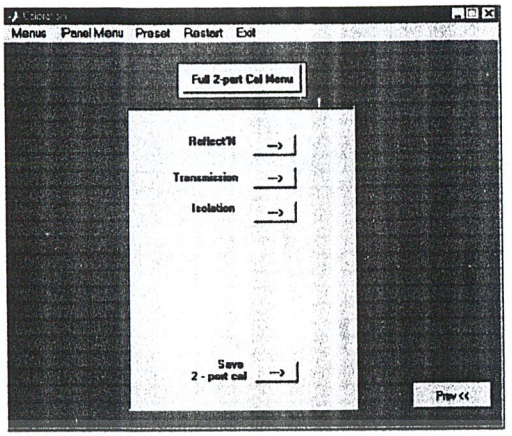
รูปที่ 18 Cal Menu

เมื่อต้องการที่จะแคลิเบรทใน Cal1 เราสามารถเลือกใช้งานในเมนู Cal 1 3.5 mm B.2 --> โดยเมื่อเลือกที่เมนูนี้แล้วนั้นโปรแกรมจะแสดงผลดังรูปที่ 19



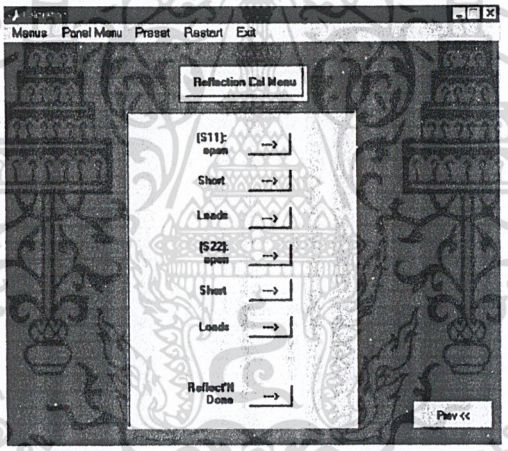
รูปที่ 19 Cal Type Menu

จากรูปที่ 19 เป็นเมนูที่ใช้สำหรับเลือกว่าต้องการที่จะแคลิเบรทพอร์ทของเนทเวิร์กอนาไลเซอร์อย่างไร โดยเมื่อเราเลือกที่จะทำการแคลิเบรททั้ง 2 พอร์ทเลือกที่ Full 2 - port --> โดยผลที่ได้จากการเลือกเมนูนี้นั้นเป็นดังรูปที่ 20



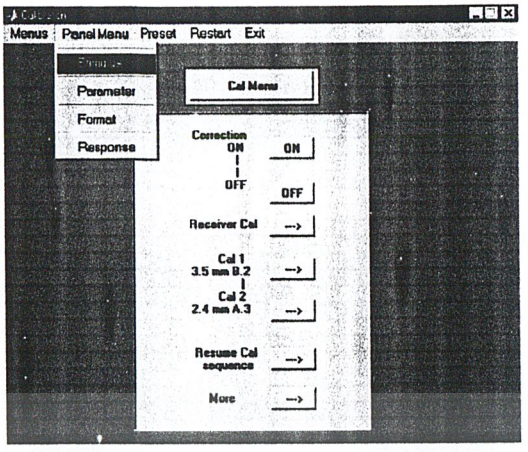
รูปที่ 20 Full 2-port Cal Menu

โดยเราต้องเลือกต้องเลือกในกี่แคลิเบรททั้ง 3 เมนูที่มีเมื่อเลือกที่ Reflect'N --> โดยผลจากการเลือกที่เมนูนี้นั้นได้แสดงดังรูปที่ 21



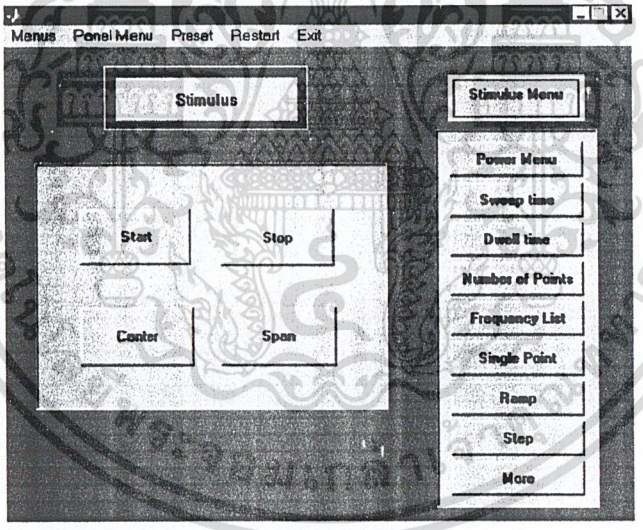
รูปที่ 21 Reflect'N Cal Menu

โดยเราต้องทำการต่ออุปกรณ์สำหรับแคลิเบรทเข้ากับเครื่องเนทเวิร์กอนาไลเซอร์ ซึ่งมีด้วยกันทั้งหมด 3 ชิ้นซึ่งก็คือ open, short, load, โดยต่ออุปกรณ์แต่ละชนิดนั้นที่พอร์ททั้ง 2 และทั้งการเลือกเมนูตามลำดับของการต่ออุปกรณ์สำหรับแคลิเบรท จากนั้นเมื่อทำการเลือกทุกฟังก์ชันของการแคลิเบรทแล้วนั้นให้เราเลือกที่ Reflect'N Done --> เมื่อเราเลือกที่เมนูนี้แล้วนั้น จากนั้นให้ทำการแคลิเบรทโดยการต่อพอร์ททั้ง 2 ของเครื่องเนทเวิร์กอนาไลเซอร์เข้าด้วยกันโดยใช้คอนเนคเตอร์ จากนั้นให้เลือกที่เมนู Transmission --> จากนั้นให้เลือกที่เมนู Isolation --> เมื่อเราทำการเลือกแคลิเบรทตามลำดับแล้วนั้นเราสามารถที่จะบันทึกสถานะของการแคลเบรทไว้ได้โดยการเลือกที่เมนู Save 2 - port cal --> จากนั้นเลือกว่าต้องการที่จะเก็บบันทึกที่ช่องใด



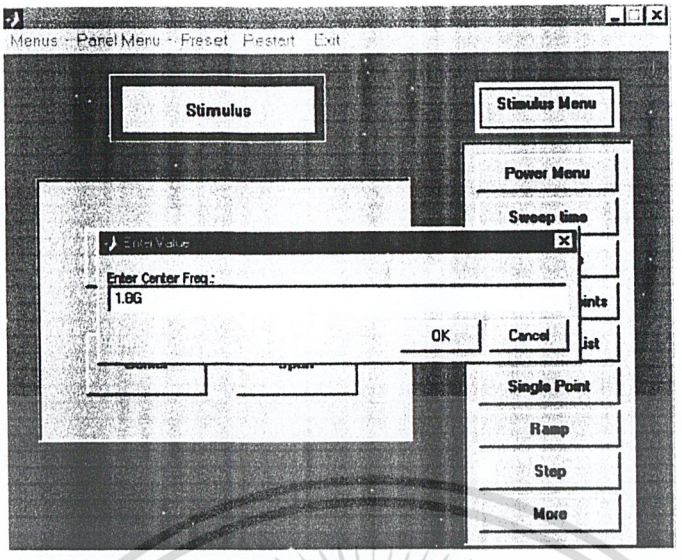
รูปที่ 22 การเรียกใช้ Stimulus Menu

เมื่อต้องการที่จะเรียกใช้งาน Stimulus Menu สามารถกระทำได้โดยการเลือกที่เมนูบาร์ดังรูปที่ 22 โดยผลของการเลือกที่เมนูนั้นจะแสดงผลดังรูปที่ 23

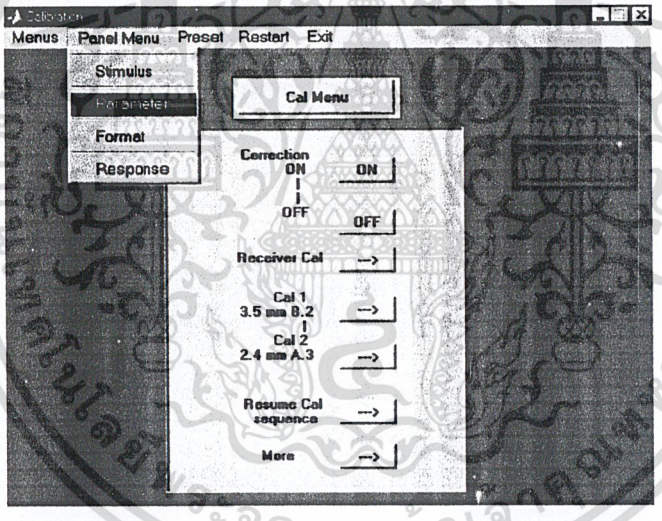


รูปที่ 23 Stimulus Menu

เป็นเมนูที่ใช้สำหรับการตั้งค่าของความถี่ที่จะใช้ในการแสดงผลสัญญาณเนทเวิร์กอนาไลเซอร์ โดยจะมีเมนูของการใช้งานต่างๆเช่น Start, Stop, Center, Span เมื่อคลิกเมาส์ไปที่คำดังกล่าวโปรแกรมจำทำการร้องขอค่าของความถี่ที่เราต้องการป้อนค่า โดยค่าของความถี่นั้นเราต้องทำการใส่หน่วยของความถี่ด้วยดังเช่นรูปที่ 24 เป็นการป้อนค่าของความถี่ Center Freq ที่ 1.8 GHz

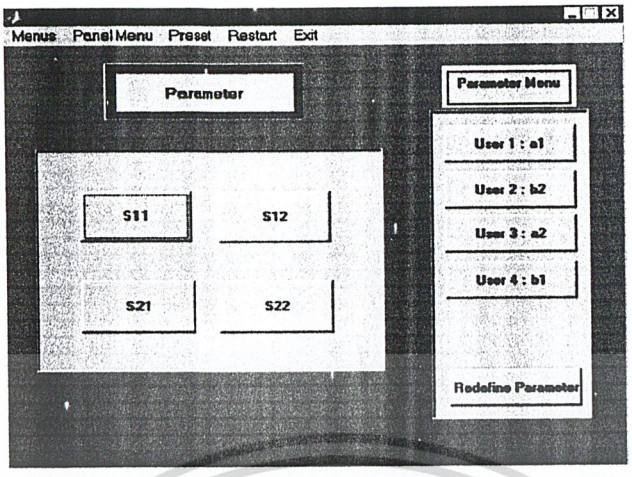


รูปที่ 27 การป้อนค่า Center Freq

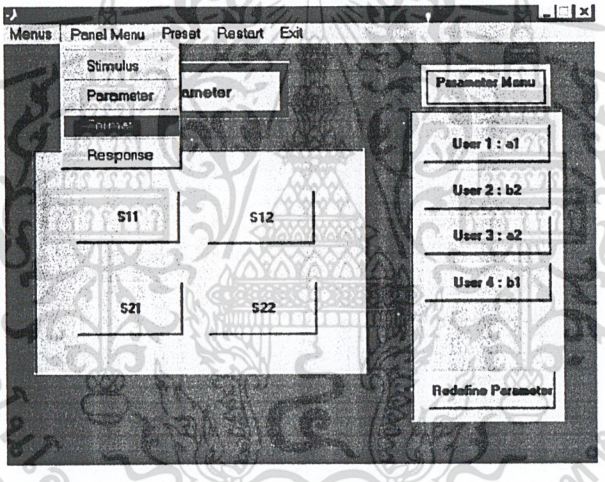


รูปที่ 28 การเรียกใช้งาน Parameter Menu

เมื่อเราทำการเลือกเมนูการใช้งานให้อยู่ใน Parameter Menu ผลของโปรแกรมจะแสดงดังรูปที่ 29 โดยในเมนูนี้นั้นเราสามารถเลือกที่จะให้เครื่องเนทเวิร์กอนาไลเซอร์ แสดงผลพารามิเตอร์ต่างๆตามที่เราต้องการซึ่งก็ได้แก่ S11, S12, S21, S22, โดยการเลือกสามารถทำได้โดยคลิกเมาส์เลือกตามตำแหน่งที่ต้องการ

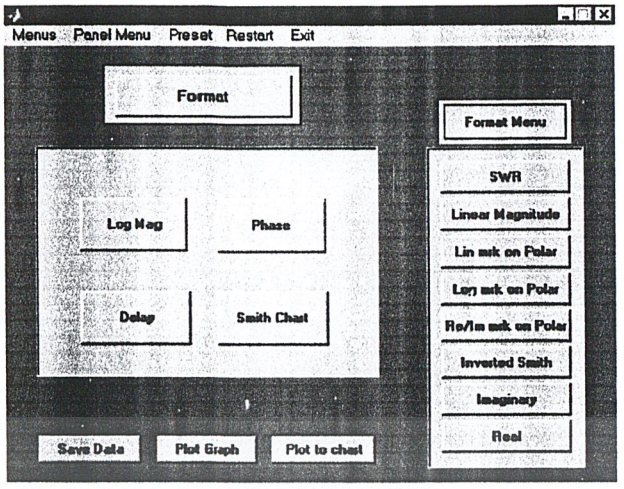


รูปที่ 29 Parameter Menu



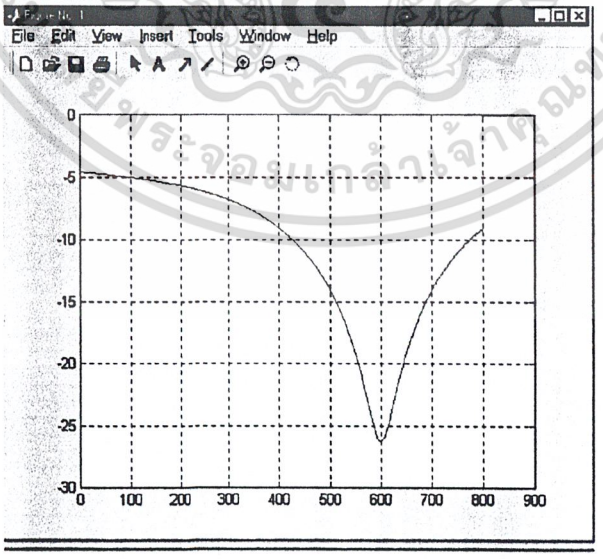
รูปที่ 30 การเรียกใช้งาน Format Menu

เราสามารถเรียกใช้งาน Format Menu ได้โดยการเลือกที่เมนูบาร์ โดยผลของการเลือก Format Menu ผลของโปรแกรมแสดงดังรูปที่ 31



รูปที่ 31 Format Menu

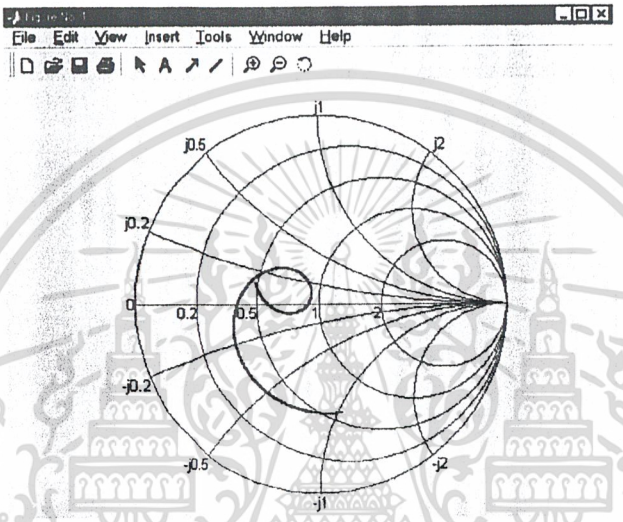
เมื่อทำการเลือกใช้งานมาที่ Format Menu จะเป็นเมนูที่ใช้สำหรับการเลือกว่าจะให้เครื่องเนทเวิร์กอนาไลเซอร์แสดงผลในรูปแบบใดโดยจะมีเมนูให้เลือกรูปแบบของการแสดงผลก็คือ Log Mag, Phase, Delay, Smith Chart, โดยเราสามารถเลือกใช้เมนูต่างๆได้โดยการคลิกเมาส์เลือกไปที่ชื่อเมนูรูปแบบตามที่เราต้องการให้เครื่องเนิทเวิร์กแสดงผล โดยใน Format Menu เราสามารถที่สั่งให้คอมพิวเตอร์เก็บหรือบันทึกค่าได้โดยการเลือกที่ Save Data และหากต้องการให้ทำการพล็อตก็สามารถทำได้โดยเลือกที่ Plot Graph ถ้าต้องการให้ทำการพล็อตในรูปของ Smith Chart เลือกที่ Plot to chart



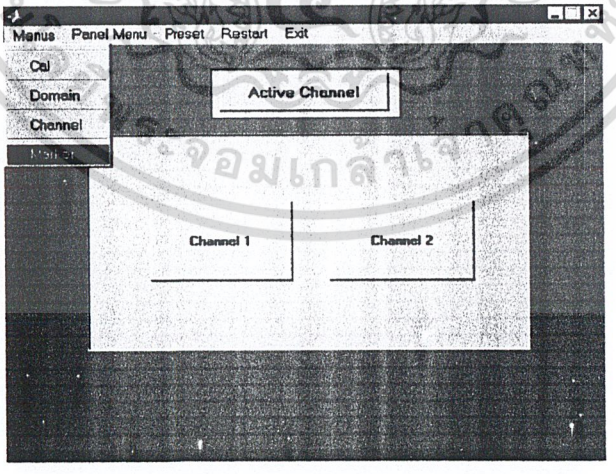
รูปที่ 32 กราฟสัญญาณใน Format ของ Log Mag

จากรูปที่ 32 เราได้เลือกให้ทำการพล็อตใน Format ของ Log Mag ซึ่งกระทำได้โดยการเลือก Log Mag ตามด้วย Plot Graph

เราสามารถที่จะให้โปรแกรมแสดงผลของการพล็อตสัญญาณใน Format ต่างๆได้เช่นกัน โดยหากต้องการให้แสดงผลใน Format ของ Smith Chart ทำได้โดย การเลือกที่เมนู Smith Chart ตามด้วยการเลือกที่ Plot to chart ซึ่งผลของการแสดงผลเป็นดังรูปที่ 33

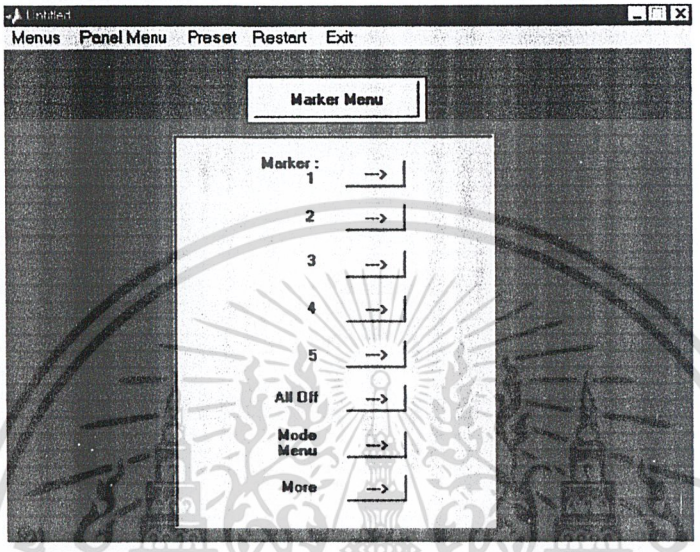


รูปที่ 33 กราฟสัญญาณใน Format ของ Smith Chart



รูปที่ 34 การเรียกใช้งาน Marker

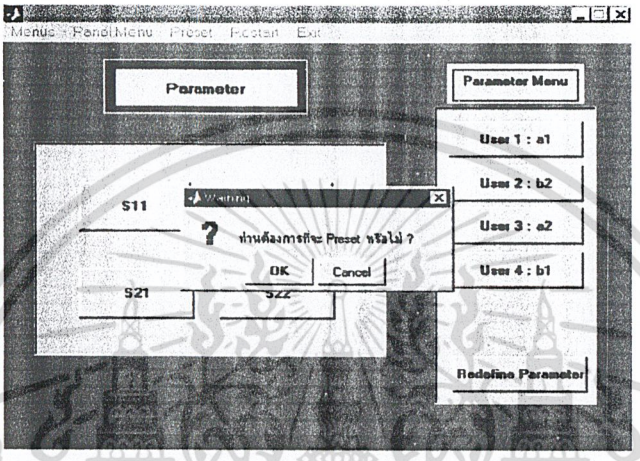
เราสามารถเรียกใช้งาน Marker ได้ด้วยการเรียกใช้ที่เมนูบาร์ สำหรับการเรียกใช้ Marker ก็เพื่อทำการตรวจวัดข้อมูลของสัญญาณในช่องของความถี่ที่เราต้องการตามสเกลที่เราสามารถเลือกได้โดยผลของโปรแกรมหลังจากเลือกเมนู Marker แล้วนั้นจะเป็นดังรูปที่ 35



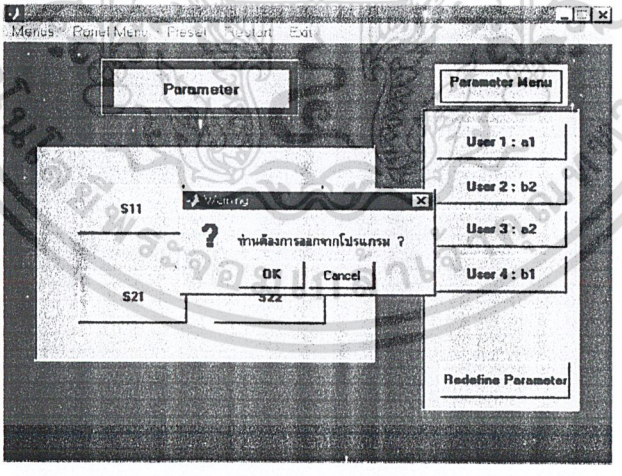
รูปที่ 35 Marker Menu

จากรูปที่ 35 ในเมนูของ Marker Menu นั้นเราสามารถเลือกได้ว่าจะทำการเรียกใช้ตัว Marker จำนวนกี่ตัวโดยเราสามารถเรียกใช้งานได้สูงสุดถึง 5 ตัว และหากเราต้องการที่จะทำการควบคุมการเคลื่อนไหวของ Marker ตัวใดเราก็สามารถทำได้โดยการเลือกเมาส์คลิกไปที่เลขของ Marker ตัวนั้นได้

เมื่อเราต้องการที่ Preset หรือออกจากโปรแกรมนั้น โปรแกรมจะทำการเตือนเราทุกครั้งว่าต้องการออกจากโปรแกรม หรือต้องการที่จะ Preset จริงหรือไม่ ดังแสดงดังรูปที่ 36 ซึ่งหากเราไม่ต้องการก็สามารถเลือกได้ที่ Cancel และถ้าต้องการออกจริงเลือกที่ OK เหตุผลที่ต้องทำการเตือนเพราะว่าหากกระทำการวัดสัญญาณใดๆไว้จะถูกรีเซทและต้องทำการวัดใหม่หมดเมื่อเลือก Preset หรือ Exit



รูปที่ 36 การเตือนเมื่อทำการ Preset



รูปที่ 37 การเตือนเมื่อทำการออกจากโปรแกรม

โปรแกรม HP VEE ที่ใช้ในการควบคุมเครื่องมือวัด

Source file: "C:\\My Documents\\VEE Programs\\test_Pro.vee"

File last revised: Wed Oct 10 18:33:36 2001

Date documented: Wed Oct 10 18:34:16 2001

VEE revision: 5.01

Compatibility mode: VEE 5

Convert Infinity on Binary Read: no

I/O Configuration

My configuration

HP-IB0

GPIO_Transfer(@(NOT LIVE))

Timeout (sec): 5.000000

Byte ordering: MSB

HP 3852A(@(NOT LIVE))

Timeout (sec): 5.000000

Byte ordering: MSB

hp34510a(@(NOT LIVE))

Panel Driver: hp34510.cid

Timeout (sec): 5.000000

Byte ordering: MSB

hp4195a(@(NOT LIVE))

Panel Driver: hp4195a.cid

Timeout (sec): 5.000000

Byte ordering: MSB

hp4291a(@(NOT LIVE))

Panel Driver: c:\smk\hp4291a\hp4291a.id

Timeout (sec): 5.000000

Byte ordering: MSB

HP_3852A(@(NOT LIVE))

Timeout (sec): 5.000000

Byte ordering: MSB

k2000(@(NOT LIVE))

Panel Driver: k2000.cid

Timeout (sec): 5.000000

Byte ordering: MSB

HP-IB7

dmm(hp34401a@(NOT LIVE))

Panel Driver: hp34401a.cid

Timeout (sec): 5.000000

Byte ordering: MSB

dvm(hp3478a@(NOT LIVE))

Panel Driver: hp3478a.cid

Timeout (sec): 5.000000

Byte ordering: MSB

fgen(hp3325b@(NOT LIVE))

Panel Driver: hp3325b.cid

Timeout (sec): 5.000000

Byte ordering: MSB

funcgen(hp33120a@(NOT LIVE))

Panel Driver: hp33120a.cid

Timeout (sec): 5.000000

Byte ordering: MSB

newDevice(@716)

Panel Driver: hp8510c.cid

Timeout (sec): 5.000000

Byte ordering: MSB

newDevice2(@717)

Timeout (sec): 5.000000

Byte ordering: MSB

oscope(hp54600@(NOT LIVE))

Panel Driver: hp54600.cid

Timeout (sec): 5.000000

Byte ordering: MSB

scope(hp54504a@(NOT LIVE))

Panel Driver: hp54504a.cid

Timeout (sec): 5.000000

Byte ordering: MSB

spectrum analyzer(hp8563e@718)

Plug&play Driver: HP856XE

VISA address: GPIB0::18::INSTR

Panel Driver: hp856xe.cid

Timeout (sec): 5.000000

Byte ordering: MSB

synthesized sweeper(hp83620a@(NOT LIVE))

Panel Driver: hp83620a.cid

Timeout (sec): 5.000000

Byte ordering: MSB

Serial9

serial(@(NOT LIVE))

Timeout (sec): 0.000000

Byte ordering: MSB

VXI0

vxi(@(NOT LIVE))

Timeout (sec): 5.000000

M: Main

Device Type : Main

Context is secured : off

Trig mode : Degrees

Popup Panel Title Text : Main

Show Popup Panel Title : on

Show Popup Panel Border : on

Popup Moveable : on

Popup Panel Title Text Color : Object Title Text

Popup Panel Title Background Color : Object Title

Popup Panel Title Text Font : Object Title Text

Delete Globals at Prerun : off

M.240: Main/Until Break

Device Type : Until Break

Output pin 1 : Continuous

M.241: Main/Cyclic Button

Device Type : Selection Control

Output pin 1 : Enum

Output pin 2 : Ordinal

Wait For Event : on

Auto execute : off

Initialize At Prerun : on

Initialize at Activate : on

Constant size fixed : off

Password masking : off

Indices Enabled : on

Enum Value : "Spectrum Panel"

Selection possibilities:

0: "Spectrum Panel"

M.242: Main/Cyclic Button

Device Type : Selection Control

Output pin 1 : Enum

Output pin 2 : Ordinal

Wait For Event : on

Auto execute : off

Initialize At Prerun : on

Initialize at Activate : on

Constant size fixed : off

Password masking : off

Indices Enabled : on

Enum Value : "Sample Panel"

Selection possibilities:

0: "Sample Panel"

M.243: Main/Cyclic Button

Device Type : Selection Control

Output pin 1 : Enum

Output pin 2 : Ordinal

Wait For Event : on

Auto execute : off

Initialize At Prerun : on

Initialize at Activate : on

Constant size fixed : off

Password masking : off

Indices Enabled : on

Enum Value : "Test Panel"

Selection possibilities:

0: "Test Panel"

M.244: Main/Call sample

Device Type : Call

Function name : sample

M.246: Main/Call spectrum

Device Type : Call

Function name : spectrum

M.247: Main/Until Break

Device Type : Until Break

Output pin 1 : Continuous

M.248: Main/Until Break

Device Type : Until Break

Output pin 1 : Continuous

M.249: Main/Cyclic Button

Device Type : Selection Control

Output pin 1 : Enum

Output pin 2 : Ordinal

Wait For Event : on

Auto execute : off

Initialize At Prerun : on

Initialize at Activate : on

Constant size fixed : off

Password masking : off

Indices Enabled : on

Enum Value : "Exit"

Selection possibilities:

0: "Exit"

M.250: Main/Until Break

Device Type : Until Break

Output pin 1 : Continuous

M.251: Main/Stop

Device Type : Stop

Exit code : 0

F5: sample

Device Type : UserFunction

Context is secured : off

Trig mode : Degrees

Popup Panel Title Text : Sample

Show Popup Panel Title : on

Show Popup Panel Border : on

Popup Moveable : off

Popup Panel Title Text Color : Object Title Text

Popup Panel Title Background Color : Object Title

Popup Panel Title Text Font : Object Title Text

F5.0: sample/CommonDialog: CommonDialog1

Device Type : ActiveX Control

Object Type : MSComDlg.CommonDialog

Variable Name : CommonDialog1

F5.1: sample/dddd

Device Type : UserObject

Input pin 1 : A (Any, Any)

Input pin 2 : B (Any, Any)

Output pin 1 : X

Context is secured : off

Trig mode : Degrees

Popup Panel Title Text : UserObject

Show Popup Panel Title : on

Show Popup Panel Border : on

Popup Moveable : on

Popup Panel Title Text Color : Object Title Text

Popup Panel Title Background Color : Object Title

Popup Panel Title Text Font : Object Title Text

F5.1.1: sample/dddd/Formula

Device Type : Formula

Output pin 1 : FileName

Formula :

1. CommonDialog1.DialogTitle = "Save File";
2. CommonDialog1.Filter = "File Type (*.*)|*.*|Documents File (*.doc) | *.doc |Excel File (*.xls) |*.xls |Text File (*.txt) |*.txt|";
3. CommonDialog1.ShowSave();
4. Filename = CommonDialog1.FileName;

F5.1.2: sample/dddd/To File

Device Type : To File

Input pin 1 : A (Any, Any)

Input pin 2 : File Name (Text, Scalar)

Transactions : WRITE TEXT a REAL SCI:6 EOL

From : C:\Program Files\Hewlett-Packard\VEE 5.0\b180.txt

Clear at PreRun & Open : yes

F5.2: sample/Build Record

Device Type : Build Record

Input pin 1 : A (Any, Any)

Input pin 2 : B (Any, Any)

Output pin 1 : Record

Record output shape : Scalar

F5.3: sample/Build Record

Device Type : Build Record

Input pin 1 : A (Any, Any)

Input pin 2 : B (Any, Any)

Output pin 1 : Record

Record output shape : Scalar

F5.4: sample/Collector

Device Type : Collector

Input pin 1 : Data (Any, Any)

Input pin 2 : XEQ (Any, Any)

Output pin 1 : Array

Clear At Prerun : on

Clear at Activate : on

Output shape 1d : on

F5.5: sample/Collector

Device Type : Collector

Input pin 1 : Data (Any, Any)

Input pin 2 : XEQ (Any, Any)

Output pin 1 : Array

Clear At Prerun : on

Clear at Activate : on

Output shape 1d : on

F5.6: sample/spectrum analyzer (hp8563e @ 718)

Device Type : Direct I/O

Output pin 1 : mka

Transactions :

1. WRITE TEXT "sngls;ts;mka?;" EOL

2. READ TEXT mka REAL

F5.7: sample/For Range

Device Type : For Range

Input pin 1 : Thru (Real, Scalar)

Output pin 1 : Data

From Value : 1

Thru Value : 200

Step Value : 1

F5.8: sample/Data Capturing

Device Type : Logging AlphaNumeric

Input pin 1 : Data (Any, Any)

Input pin 2 : Clear (Any, Any)

Clear At Prerun : on

Clear at Activate : on

Buffer Size : 10000

F5.9: sample/spectrum analyzer (hp8563e @ 718)

Device Type : Direct I/O

Transactions : WRITE TEXT "sngls;ts;mkpk hi;" EOL

F5.10: sample/elapsed time

Device Type : Timer

Input pin 1 : Time1 (Any, Any)

Input pin 2 : Time2 (Any, Any)

Output pin 1 : T2 - T1

Clear At Prerun : on

Clear at Activate : on

F5.11: sample/Message Box

Device Type : Message Box

Output pin 1 : OK

Output pin 2 : Cancel

Dialog Auto Timeout : off

Dialog Timeout : 60

Dialog Popup Title : Message Box

Dialog Show Popup Title : on

Dialog Popup Position : (223,112)

Message : !!!!! Press "OK" to capturing !!!!!

Symbol : Question

Buttons : "OK Cancel"

Default button : "OK"

F5.12: sample/Logging AlphaNumeric

Device Type : Logging AlphaNumeric

Input pin 1 : Data (Any, Any)

Input pin 2 : Clear (Any, Any)

Clear At Prerun : on

Clear at Activate : on

Buffer Size : 10000

F5.13: sample/Timer

Device Type : Timer

Input pin 1 : Time1 (Any, Any)

Input pin 2 : Time2 (Any, Any)

Output pin 1 : T2 - T1

Clear At Prerun : on

Clear at Activate : on

F5.14: sample/Next

Device Type : Next

F5.15: sample/Until Break

Device Type : Until Break

Output pin 1 : Continuous

F5.16: sample/Next

Device Type : Next

F5.17: sample/Set Variable

Device Type : Set Variable

Input pin 1 : Data (Any, Any)

Variable Name : globalA

F5.18: sample/Get Variable

Device Type : Get Variable

Output pin 1 : Data

Variable Name : globalA

F5.19: sample/Logging AlphaNumeric

Device Type : Logging AlphaNumeric

Input pin 1 : Data (Any, Any)

Clear At Prerun : on

Clear at Activate : on

Buffer Size : 10000

F5.20: sample/Until Break

Device Type : Until Break

Output pin 1 : Continuous

F5.21: sample/Cyclic Button

Device Type : Selection Control

Output pin 1 : Enum

Output pin 2 : Ordinal

Wait For Event : on

Auto execute : off

Initialize At Prerun : on

Initialize at Activate : on

Constant size fixed : off

Password masking : off

Indices Enabled : on

Enum Value : "Save"

Selection possibilities:

0: "Save"

F5.22: sample/Cyclic Button

Device Type : Selection Control

Output pin 1 : Enum

Output pin 2 : Ordinal

Wait For Event : on

Auto execute : off

Initialize At Prerun : on

Initialize at Activate : on

Constant size fixed : off

Password masking : off

Indices Enabled : on

Enum Value : "Sample"

Selection possibilities:

0: "Sample"

F5.23: sample/Graph

Device Type : X vs Y Plot

Input pin 1 : XData (Any, Any)

Input pin 2 : Power Level (dBM) (Any, Any)

Input pin 3 : Clear (Any, Any)

Input pin 4 : Print (Any, Any)

Clear At Prerun : on

Clear at Activate : on

Variable Name :

Marker 1 Domain Value : 0

Marker 1 Range Value : 0

Marker 2 Domain Value : 0

Marker 2 Range Value : 0

Delta Marker Domain Value: 0

Delta Marker Range Value : 0

F5.24: sample/Declare Variable

Device Type : Declare Variable

Variable Scope : Global

Variable Name : global1

Record Value : <Arrays not printed>

F5.25: sample/Logging AlphaNumeric

Device Type : Logging AlphaNumeric

Input pin 1 : Data (Any, Any)

Input pin 2 : Clear (Any, Any)

Clear At Prerun : on

Clear at Activate : on

Buffer Size : 10000

F5.28: sample/Cyclic Button

Device Type : Selection Control

Output pin 1 : Enum

Output pin 2 : Ordinal

Wait For Event : on

Auto execute : off

Initialize At Prerun : on

Initialize at Activate : on

Constant size fixed : off

Password masking : off

Indices Enabled : on

Enum Value : "Print "

Selection possibilities:

0: "Print "

F5.30: sample/Integer Input

Device Type : Integer Input

Output pin 1 : Value

Output pin 2 : Cancel

Dialog Auto Timeout : off

Dialog Timeout : 60

Dialog Popup Title : Text Input

Dialog Show Popup Title : on

Dialog Popup Position : (200,134)

Prompt/Label : Enter Sampling Value:

Default Value : 0

Value Constraint :

Error Message : You must enter Sampling Value???

Password masking : off

OK Label : OK

Cancel Label : Cancel

Displayed Characters : 20

Error Msg Popup Position : (200,200)

F5.31: sample/Next

Device Type : Next

F5.32: sample/On Cycle

Device Type : On Cycle

Output pin 1 : Alarm

Cycle Time : 0

F5.34: sample/Exit UserObject

Device Type : Exit UserObject

F5.35: sample/exit to main

Device Type : OK

Output pin 1 : Go

Assign to Enter Button : on

Assign to Escape Button : on

F6: spectrum

Device Type : UserFunction

Context is secured : off

Trig mode : Degrees

Popup Panel Title Text : Spectrum Panel

Show Popup Panel Title : on

Show Popup Panel Border : on

Popup Moveable : off

Popup Panel Title Text Color : Object Title Text

Popup Panel Title Background Color : Object Title

Popup Panel Title Text Font : Object Title Text

F6.0: spectrum/spectrum analyzer (HP-8563E)

Device Type : State Driver

F6.1: spectrum/Until Break

Device Type : Until Break

Output pin 1 : Continuous

F6.2: spectrum/Exit UserObject

Device Type : Exit UserObject

F6.3: spectrum/exit to main

Device Type : OK

Output pin 1 : Go

Assign to Enter Button : on

Assign to Escape Button : on

โปรแกรม MATLAB ที่ใช้ในการควบคุมเครื่องมือวัด

โปรแกรมย่อย Cal

```
function varargout = cal(varargin)
if nargin == 0  % LAUNCH GUI
        fig = openfig(mfilename,'reuse');
        % Generate a structure of handles to pass to callbacks, and store it.
        handles = guihandles(fig);
        guidata(fig, handles);
        if nargout > 0
                varargout{1} = fig;
        end
elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK
        try
                [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
        catch
                disp(lasterr);
        end
end
function varargout = pushbutton4_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fclose(g)
   delete(g)
   clear g
cal
close(gcbf)
function varargout = pushbutton8_Callback(h, eventdata, handles, varargin)
clear all
close all
```

```
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fprintf(g,'corron;');
fclose(g)
delete(g)
clear g
cal_correction_on
close(gcbf)
function varargout = pushbutton11_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fprintf(g,'cal1;');
   fclose(g)
   delete(g)
   clear g
cal_cal1_2
close(gcbf)
function varargout = pushbutton12_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fprintf(g,'cal2;');
   fclose(g)
   delete(g)
   clear g
cal_cal_2
```

```
close(gcbf)
function varargout = pushbutton14_Callback(h, eventdata, handles, varargin)
cal_more
close(gcbf)
function varargout = pushbutton10_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fprintf(g,'calrcvr;');
   fclose(g)
   delete(g)
   clear g
receiver_cal
close(gcbf)
function varargout = pushbutton9_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fprintf(g,'corroff;');
fclose(g)
delete(g)
clear g
function varargout = pushbutton13_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
```

```
    fprintf(g,'resc;');
    fclose(g)
    delete(g)
    clear g
function varargout = pushbutton4_CreateFcn(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menustim;');
    fclose(g)
    delete(g)
    clear g
    stim_cal
    close(gcbf)
function varargout = pushbutton24_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menustim;');
    fclose(g)
    delete(g)
    clear g
    stim_cal
    close(gcbf)
function varargout = pushbutton25_Callback(h, eventdata, handles, varargin)
net2
close(gcbf)
function varargout = pushbutton16_Callback(h, eventdata, handles, varargin)
cal('pushbutton8_Callback',gcbo,[],guidata(gcbo))
```

โปรแกรมย่อย cal_cal_2

```
if nargin == 0  % LAUNCH GUI
        fig = openfig(mfilename,'reuse');
        handles = guihandles(fig);
        guidata(fig, handles);
        if nargout > 0
                varargout{1} = fig;
        end
elseif ischar(varargin{1})
        try
                [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
        catch
                disp(lasterr);
        end
end
function varargout = pushbutton4_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menustim;');
   fclose(g)
   delete(g)
   clear g
stim_cal
close(gcbf)
function varargout = pushbutton8_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
```

```
    fprintf(g,'cal2;');
    fprintf(g,'setf;');
    fclose(g)
    delete(g)
    clear g
function varargout = pushbutton11_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'cal2;');
    fprintf(g,'calis111;');
    fclose(g)
    delete(g)
    clear g
cal2_s11
close(gcbf)
function varargout = pushbutton12_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'cal2;');
    fprintf(g,'calis221;');
    fclose(g)
    delete(g)
    clear g
cal2_s22
close(gcbf)
function varargout = pushbutton6_Callback(h, eventdata, handles, varargin)
clear all
```

```
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fclose(g)
   delete(g)
   clear g
cal
close(gcbf)
function varargout = pushbutton5_Callback(h, eventdata, handles, varargin)
close(gcbf)
function varargout = pushbutton16_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fclose(g)
   delete(g)
   clear g
cal
close(gcbf)
function varargout = pushbutton14_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fprintf(g,'cal2;');
   fprintf(g,'califul2;');
   fclose(g)
   delete(g)
```

```
    clear g
full_cal
close(gcbf)
function varargout = pushbutton10_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'cal2;');
    fprintf(g,'calirai;');
    fclose(g)
    delete(g)
    clear g
 close(gcbf)
function varargout = pushbutton9_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'cal2;');
    fprintf(g,'caliresp;');
    fclose(g)
    delete(g)
    clear g
 close(gcbf)
function varargout = pushbutton15_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
```

```
    fprintf(g,'menucal;');
    fprintf(g,'cal2;');
    fprintf(g,'calitrl2;');
    fclose(g)
    delete(g)
    clear g
    trl_cal2
 close(gcbf)
function varargout = pushbutton13_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'cal2;');
    fprintf(g,'calione2;');
    fclose(g)
    delete(g)
    clear g
one_path_cal2
    close(gcbf)
```

โปรแกรมย่อย cal_correction_on

```
function varargout = cal_correction_on(varargin)
if nargin == 0  % LAUNCH GUI
        fig = openfig(mfilename,'reuse');
                handles = guihandles(fig);
        guidata(fig, handles);
        if nargout > 0
                varargout{1} = fig;
        end
elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK
        try
                [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
        catch
                disp(lasterr);
        end
end
function varargout = pushbutton4_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menustim;');
   fclose(g)
   delete(g)
   clear g
stim_cal
close(gcbf)
function varargout = pushbutton8_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
```

```
    fprintf(g,'menucal;');
    fprintf(g,'corron;');
    fprintf(g,'cals1;');
    fclose(g)
    delete(g)
    clear g
function varargout = pushbutton6_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fclose(g)
    delete(g)
    clear g
cal
close(gcbf)
function varargout = pushbutton5_Callback(h, eventdata, handles, varargin)
close(gcbf)
function varargout = pushbutton16_Callback(h, eventdata, handles, varargin)
cal
close(gcbf)
function varargout = pushbutton9_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'corron;');
    fprintf(g,'cals2;');
    fclose(g)
    delete(g)
    clear g
```

```
function varargout = pushbutton10_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fprintf(g,'corron;');
   fprintf(g,'cals3;');
   fclose(g)
   delete(g)
   clear g
function varargout = pushbutton11_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fprintf(g,'corron;');
   fprintf(g,'cals4;');
   fclose(g)
   delete(g)
   clear g
function varargout = pushbutton12_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fprintf(g,'corron;');
   fprintf(g,'cals5;');
   fclose(g)
   delete(g)
```

```
    clear g
function varargout = pushbutton13_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'corron;');
    fprintf(g,'cals6;');
    fclose(g)
    delete(g)
    clear g
function varargout = pushbutton14_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'corron;');
    fprintf(g,'cals8;');
    fclose(g)
    delete(g)
    clear g
function varargout = pushbutton15_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'corron;');
    fprintf(g,'cals8;');
    fclose(g)
```

```
    delete(g)
    clear g
function varargout = pushbutton17_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fclose(g)
    delete(g)
    clear g
 cal
close(gcbf)
```

โปรแกรมย่อย cal_domain

```
function varargout = cal_domain(varargin)
if nargin == 0  % LAUNCH GUI
        fig = openfig(mfilename,'reuse');
        handles = guihandles(fig);
        guidata(fig, handles);
        if nargout > 0
                varargout{1} = fig;
        end
elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK
        try
                [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
        catch
                disp(lasterr);
        end
end
function varargout = pushbutton5_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menudoma;');
   fprintf(g,'freq;');
   fclose(g)
   delete(g)
   clear g
function varargout = pushbutton4_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menudoma;');
```

```
    fprintf(g,'timl;');
    fclose(g)
    delete(g)
    clear g
time_low
close(gcbf)
function varargout = pushbutton3_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menudoma;');
    fprintf(g,'timb;');
    fclose(g)
    delete(g)
    clear g
function varargout = pushbutton2_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menudoma;');
    fprintf(g,'auxv;');
    fclose(g)
    delete(g)
    clear g
function varargout = pushbutton6_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menudoma;');
```

```
    fprintf(g,'pulp;');
    fclose(g)
    delete(g)
    clear g
function varargout = pushbutton7_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menudoma;');
    fprintf(g,'powd;');
    fclose(g)
    delete(g)
    clear g
p_domain
close(gcbf)
function varargout = pushbutton1_Callback(h, eventdata, handles, varargin)
cal_gate
close(gcbf)
```

โปรแกรมย่อย cal_s11

```
function varargout = cal_s11(varargin)
if nargin == 0  % LAUNCH GUI
        fig = openfig(mfilename,'reuse');
        handles = guihandles(fig);
        guidata(fig, handles);
        if nargout > 0
                varargout{1} = fig;
        end
elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK
        try
                [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
        catch
                disp(lasterr);
        end
end
function varargout = pushbutton4_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menustim;');
   fclose(g)
   delete(g)
   clear g
 stim_cal
 close(gcbf)
function varargout = pushbutton8_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
```

```
    fprintf(g,'menucal;');
    fprintf(g,'cal1;');
    fprintf(g,'calis111;');
    fprintf(g,'class11a;');
    fclose(g)
    delete(g)
    clear g
function varargout = pushbutton10_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'cal1;');
    fprintf(g,'calis111;');
    fprintf(g,'class11c;');
    fclose(g)
    delete(g)
    clear g
cal_s11_load
close(gcbf)
function varargout = pushbutton6_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fclose(g)
    delete(g)
    clear g
cal
close(gcbf)
function varargout = pushbutton5_Callback(h, eventdata, handles, varargin)
```

```
close(gcbf)
function varargout = pushbutton17_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fprintf(g,'cal1;');
   fclose(g)
   delete(g)
   clear g
cal_cal1_2
close(gcbf)
function varargout = pushbutton15_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fprintf(g,'cal1;');
   fprintf(g,'calis111;');
   fprintf(g,'sav1;');
   fclose(g)
   delete(g)
   clear g
save12_cal
close(gcbf)
function varargout = pushbutton9_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
```

```
    fprintf(g,'menucal;');
    fprintf(g,'cal1;');
    fprintf(g,'calis111;');
    fprintf(g,'class11b;');
    fclose(g)
    delete(g)
    clear g
function varargout = pushbutton19_Callback(h, eventdata, handles, varargin)
cal_s11('pushbutton8_Callback',gcbo,[],guidata(gcbo))
```

โปแกรมย่อย cal_s22_load

```
function varargout = cal_s22_load(varargin)
if nargin == 0  % LAUNCH GUI
        fig = openfig(mfilename,'reuse');
                handles = guihandles(fig);
        guidata(fig, handles);
        if nargout > 0
                varargout{1} = fig;
        end
elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK
        try
                [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
        catch
                disp(lasterr);
        end
end
function varargout = pushbutton8_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fprintf(g,'cal1;');
   fprintf(g,'calis221;');
   fprintf(g,'class22c;');
   fprintf(g,'stana;');
   fclose(g)
   delete(g)
   clear g
function varargout = pushbutton6_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
```

```
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
  fclose(g)
   delete(g)
   clear g
cal
close(gcbf)
function varargout = pushbutton5_Callback(h, eventdata, handles, varargin)
close(gcbf)
function varargout = pushbutton18_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fprintf(g,'cal1;');
   fprintf(g,'calis221;');
   fclose(g)
   delete(g)
   clear g
cal_s22
close(gcbf)
function varargout = pushbutton9_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fprintf(g,'cal1;');
   fprintf(g,'calis221;');
   fprintf(g,'class22c;');
   fprintf(g,'stanb;');
```

```
    fclose(g)
    delete(g)
    clear g
sliding12_load
close(gcbf)
function varargout = pushbutton10_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'cal1;');
    fprintf(g,'calis221;');
    fprintf(g,'class22c;');
    fprintf(g,'stanc;');
    fclose(g)
    delete(g)
    clear g
function varargout = pushbutton17_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'cal1;');
    fprintf(g,'calis221;');
    fprintf(g,'class22c;');
    fprintf(g,'stand;');
    fclose(g)
    delete(g)
    clear g
    offset12_load
close(gcbf)
```

```
function varargout = pushbutton15_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'cal1;');
    fprintf(g,'calis221;');
    fprintf(g,'class22c;');
    fprintf(g,'done;');
    fclose(g)
    delete(g)
    clear g
```

โปรแกรมย่อย full_load

```
function varargout = full_load(varargin)
if nargin == 0  % LAUNCH GUI
        fig = openfig(mfilename,'reuse');
        handles = guihandles(fig);
        guidata(fig, handles);
        if nargout > 0
                varargout{1} = fig;
        end
elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK
        try
                [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
        catch
                disp(lasterr);
        end
end
function varargout = pushbutton4_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menustim;');
   fclose(g)
   delete(g)
   clear g
stim_cal
close(gcbf)
function varargout = pushbutton8_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
```

```
fprintf(g,'menucal;');
fprintf(g,'cal1;');
fprintf(g,'califul2;');
fprintf(g,'refl;');
fprintf(g,'class11c;');
fprintf(g,'stana;');
fclose(g)
delete(g)
clear g
function varargout = pushbutton6_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
fprintf(g,'menucal;');
fclose(g)
delete(g)
clear g
cal
close(gcbf)
function varargout = pushbutton5_Callback(h, eventdata, handles, varargin)
close(gcbf)
function varargout = pushbutton18_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
fprintf(g,'menucal;');
fprintf(g,'cal1;');
fprintf(g,'califul2;');
fprintf(g,'refl;');
fclose(g)
delete(g)
```

```
    clear g
full_reflec_cal
close(gcbf)
function varargout = pushbutton9_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'cal1;');
    fprintf(g,'califul2;');
    fprintf(g,'refl;');
    fprintf(g,'class11c;');
    fprintf(g,'stanb;');
    fclose(g)
    delete(g)
    clear g
sliding_load_full
close(gcbf)
function varargout = pushbutton10_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'cal1;');
    fprintf(g,'califul2;');
    fprintf(g,'refl;');
    fprintf(g,'class11c;');
    fprintf(g,'stanc;');
    fclose(g)
    delete(g)
    clear g
```

```
function varargout = pushbutton17_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fprintf(g,'cal1;');
   fprintf(g,'califul2;');
   fprintf(g,'refl;');
   fprintf(g,'class11c;');
   fprintf(g,'stand;');
   fclose(g)
   delete(g)
   clear g
offset_load_full
close(gcbf)
function varargout = pushbutton15_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'cal1;');
    fprintf(g,'califul2;');
    fprintf(g,'refl;');
    fprintf(g,'class11c;');
    fprintf(g,'done;');
    fclose(g)
    delete(g)
    clear g
```

โปรแกรมย่อย full_reflec_cal

```
function varargout = full_reflec_cal(varargin)
if nargin == 0  % LAUNCH GUI
        fig = openfig(mfilename,'reuse');
                handles = guihandles(fig);
        guidata(fig, handles);
        if nargout > 0
                varargout{1} = fig;
        end
elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK
        try
                [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
        catch
                disp(lasterr);
        end
end
function varargout = pushbutton4_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menustim;');
   fclose(g)
   delete(g)
   clear g
   stim_cal
close(gcbf)
function varargout = pushbutton8_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
```

```
fprintf(g,'menucal;');
fprintf(g,'cal1;');
fprintf(g,'califul2;');
fprintf(g,'refl;');
fprintf(g,'class11a;');
fclose(g)
delete(g)
clear g
function varargout = pushbutton11_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
fprintf(g,'menucal;');
fprintf(g,'cal1;');
fprintf(g,'califul2;');
fprintf(g,'refl;');
fprintf(g,'class22a;');
fclose(g)
delete(g)
clear g
function varargout = pushbutton12_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
fprintf(g,'menucal;');
fprintf(g,'cal1;');
fprintf(g,'califul2;');
fprintf(g,'refl;');
fprintf(g,'class22b;');
fclose(g)
delete(g)
```

```
    clear g
function varargout = pushbutton14_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'cal1;');
    fprintf(g,'califul2;');
    fprintf(g,'refl;');
    fprintf(g,'refd;');
    fclose(g)
    delete(g)
    clear g
function varargout = pushbutton10_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'cal1;');
    fprintf(g,'califul2;');
    fprintf(g,'refl;');
    fprintf(g,'class11c;');
    fclose(g)
    delete(g)
    clear g
full_load
close(gcbf)
function varargout = pushbutton5_Callback(h, eventdata, handles, varargin)
close(gcbf)
function varargout = pushbutton9_Callback(h, eventdata, handles, varargin)
clear all
```

```
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fprintf(g,'cal1;');
   fprintf(g,'califul2;');
   fprintf(g,'refl;');
   fprintf(g,'class11b;');
   fclose(g)
   delete(g)
   clear g
function varargout = pushbutton15_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fprintf(g,'cal1;');
   fprintf(g,'califul2;');
   fclose(g)
   delete(g)
   clear g
full_cal
close(gcbf)
function varargout = pushbutton13_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fprintf(g,'cal1;');
   fprintf(g,'califul2;');
```

```
    fprintf(g,'refl;');
    fprintf(g,'class22c;');
    fclose(g)
    delete(g)
    clear g
full1_load22
close(gcbf)
```

โปรแกรมย่อย full1_load

```
function varargout = full1_load(varargin)
if nargin == 0  % LAUNCH GUI
        fig = openfig(mfilename,'reuse');
        handles = guihandles(fig);
        guidata(fig, handles);
        if nargout > 0
                varargout{1} = fig;
        end
elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK
        try
                [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
        catch
                disp(lasterr);
        end
end
function varargout = pushbutton4_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menustim;');
   fclose(g)
   delete(g)
   clear g
   stim_cal
   close(gcbf)
function varargout = pushbutton8_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
```

```
fprintf(g,'menucal;');
fprintf(g,'cal1;');
fprintf(g,'califul2;');
fprintf(g,'refl;');
fprintf(g,'class22c;');
fprintf(g,'stana;');
fclose(g)
delete(g)
clear g
function varargout = pushbutton6_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
fprintf(g,'menucal;');
fclose(g)
delete(g)
clear g
cal
close(gcbf)
function varargout = pushbutton5_Callback(h, eventdata, handles, varargin)
close(gcbf)
function varargout = pushbutton18_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
fprintf(g,'menucal;');
fprintf(g,'cal1;');
fprintf(g,'califul2;');
fprintf(g,'refl;');
fclose(g)
delete(g)
```

```
    clear g
full_reflec_cal
close(gcbf)
function varargout = pushbutton9_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'cal1;');
    fprintf(g,'califul2;');
    fprintf(g,'refl;');
    fprintf(g,'class22c;');
    fprintf(g,'stanb;');
    fclose(g)
    delete(g)
    clear g
close(gcbf)
function varargout = pushbutton10_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'cal1;');
    fprintf(g,'califul2;');
    fprintf(g,'refl;');
    fprintf(g,'class22c;');
    fprintf(g,'stanc;');
    fclose(g)
    delete(g)
    clear g
function varargout = pushbutton17_Callback(h, eventdata, handles, varargin)
```

```
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fprintf(g,'cal1;');
   fprintf(g,'califul2;');
   fprintf(g,'refl;');
   fprintf(g,'class22c;');
   fprintf(g,'stand;');
   fclose(g)
   delete(g)
   clear g
close(gcbf)
function varargout = pushbutton15_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'cal1;');
    fprintf(g,'califul2;');
    fprintf(g,'refl;');
    fprintf(g,'class22c;');
    fprintf(g,'done;');
    fclose(g)
    delete(g)
    clear g
```

โปรแกรมย่อย fullcal22_load

```
function varargout = fullcal22_load(varargin)
if nargin == 0  % LAUNCH GUI
        fig = openfig(mfilename,'reuse');
        handles = guihandles(fig);
        guidata(fig, handles);
        if nargout > 0
                varargout{1} = fig;
        end
elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK
        try
                [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
        catch
                disp(lasterr);
        end
end
function varargout = pushbutton4_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menustim;');
   fclose(g)
   delete(g)
   clear g
stim_cal
close(gcbf)
function varargout = pushbutton8_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
```

```
    fprintf(g,'menucal;');
    fprintf(g,'cal2;');
    fprintf(g,'califul2;');
    fprintf(g,'refl;');
    fprintf(g,'class22c;');
    fprintf(g,'stana;');
    fclose(g)
    delete(g)
    clear g
function varargout = pushbutton6_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fclose(g)
    delete(g)
    clear g
cal
close(gcbf)
function varargout = pushbutton5_Callback(h, eventdata, handles, varargin)
close(gcbf)
function varargout = pushbutton18_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'cal2;');
    fprintf(g,'califul2;');
    fprintf(g,'refl;');
    fclose(g)
    delete(g)
```

```
    clear g
full_reflec_cal2
close(gcbf)
function varargout = pushbutton9_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'cal2;');
    fprintf(g,'califul2;');
    fprintf(g,'refl;');
    fprintf(g,'class22c;');
    fprintf(g,'stanb;');
    fclose(g)
    delete(g)
    clear g
sliding_load_fullcal22
close(gcbf)
function varargout = pushbutton10_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'cal2;');
    fprintf(g,'califul2;');
    fprintf(g,'refl;');
    fprintf(g,'class22c;');
    fprintf(g,'stanc;');
    fclose(g)
    delete(g)
    clear g
```

```
function varargout = pushbutton17_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'cal2;');
    fprintf(g,'califul2;');
    fprintf(g,'refl;');
    fprintf(g,'class22c;');
    fprintf(g,'stand;');
    fclose(g)
    delete(g)
    clear g
offset_load_fullcal22
close(gcbf)
function varargout = pushbutton15_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'cal2;');
    fprintf(g,'califul2;');
    fprintf(g,'refl;');
    fprintf(g,'class22c;');
    fprintf(g,'done;');
    fclose(g)
    delete(g)
    clear g
```

โปรแกรมย่อย isolation_cal

```
function varargout = isolation_cal(varargin)
if nargin == 0  % LAUNCH GUI
        fig = openfig(mfilename,'reuse');
        handles = guihandles(fig);
        guidata(fig, handles);
        if nargout > 0
                varargout{1} = fig;
        end
elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK
        try
                [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
        catch
                disp(lasterr);
        end
end
function varargout = pushbutton4_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menustim;');
   fclose(g)
   delete(g)
   clear g
   stim_cal
   close(gcbf)
function varargout = pushbutton8_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
```

```
    fprintf(g,'menucal;');
    fprintf(g,'cal1;');
    fprintf(g,'califul2;');
    fprintf(g,'isol;');
    fprintf(g,'omii;');
    fclose(g)
    delete(g)
    clear g
function varargout = pushbutton10_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'cal1;');
    fprintf(g,'califul2;');
    fprintf(g,'isol;');
    fprintf(g,'revi;');
    fclose(g)
    delete(g)
    clear g
function varargout = pushbutton6_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fclose(g)
    delete(g)
    clear g
cal
close(gcbf)
function varargout = pushbutton5_Callback(h, eventdata, handles, varargin)
```

```
close(gcbf)
function varargout = pushbutton17_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fprintf(g,'cal1;');
   fprintf(g,'califul2;');
   fclose(g)
   delete(g)
   clear g
full_cal
close(gcbf)
function varargout = pushbutton15_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fprintf(g,'cal1;');
   fprintf(g,'califul2;');
   fprintf(g,'isol;');
   fprintf(g,'isod;');
   fclose(g)
   delete(g)
   clear g
function varargout = pushbutton9_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
```

```
    fprintf(g,'menucal;');
    fprintf(g,'cal1;');
    fprintf(g,'califul2;');
    fprintf(g,'isol;');
    fprintf(g,'fwdi;');
    fclose(g)
    delete(g)
    clear g
function varargout = pushbutton23_Callback(h, eventdata, handles, varargin)
cal('pushbutton26_Callback',gcbo,[],guidata(gcbo))
```

โปรแกรมย่อย isolation_trl2

```
function varargout = isolation_trl2(varargin)
if nargin == 0  % LAUNCH GUI
        fig = openfig(mfilename,'reuse');
        handles = guihandles(fig);
        guidata(fig, handles);
        if nargout > 0
                varargout{1} = fig;
        end
elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK
        try
                [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
        catch
                disp(lasterr);
        end
end
function varargout = pushbutton4_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menustim;');
   fclose(g)
   delete(g)
   clear g
   stim_cal
   close(gcbf)
function varargout = pushbutton8_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
```

```
    fprintf(g,'menucal;');
    fprintf(g,'cal2;');
    fprintf(g,'calitrl2;');
    fprintf(g,'trll;');
    fprintf(g,'omii;');
    fclose(g)
    delete(g)
    clear g
function varargout = pushbutton10_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'cal2;');
    fprintf(g,'calitrl2;');
    fprintf(g,'trll;');
    fprintf(g,'revi;');
    fclose(g)
    delete(g)
    clear g
function varargout = pushbutton6_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fclose(g)
    delete(g)
    clear g
cal
close(gcbf)
function varargout = pushbutton5_Callback(h, eventdata, handles, varargin)
```

```
close(gcbf)
function varargout = pushbutton17_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fprintf(g,'cal2;');
   fprintf(g,'calitrl2;');
   fclose(g)
   delete(g)
   clear g
trl_cal2
close(gcbf)
function varargout = pushbutton15_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fprintf(g,'cal2;');
   fprintf(g,'calitrl2;');
   fprintf(g,'trll;');
   fprintf(g,'isod;');
   fclose(g)
   delete(g)
   clear g
function varargout = pushbutton9_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
```

```
fprintf(g,'menucal;');
fprintf(g,'cal2;');
fprintf(g,'calitrl2;');
fprintf(g,'trll;');
fprintf(g,'fwdi;');
fclose(g)
delete(g)
clear g
```

โปรแกรมย่อย marker_mode

```
function varargout = marker_mode(varargin)
if nargin == 0  % LAUNCH GUI
        fig = openfig(mfilename,'reuse');
        handles = guihandles(fig);
        guidata(fig, handles);
        if nargout > 0
                varargout{1} = fig;
        end
elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK
        try
                [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
        catch
                disp(lasterr);
        end
end
function varargout = pushbutton5_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menumark;');
   fprintf(g,'delr1;');
   marker
   fprintf(g,'menumark;');
   fclose(g)
   delete(g)
   clear g
close(gcbf)
function varargout = pushbutton4_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
```

```
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menumark;');
   fprintf(g,'delr2;');
   marker
   fprintf(g,'menumark;');
   fclose(g)
   delete(g)
   clear g
close(gcbf)
function varargout = pushbutton3_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menumark;');
   fprintf(g,'delr3;');
   marker
   fprintf(g,'menumark;');
   fclose(g)
   delete(g)
   clear g
close(gcbf)
function varargout = pushbutton2_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menumark;');
   fprintf(g,'delr4;');
   marker
   fprintf(g,'menumark;');
   fclose(g)
```

```
    delete(g)
    clear g
close(gcbf)
function varargout = pushbutton6_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menumark;');
    fprintf(g,'delr5;');
    marker
    fprintf(g,'menumark;');
    fclose(g)
    delete(g)
    clear g
close(gcbf)
function varargout = pushbutton7_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menumark;');
    fprintf(g,'delro;');
    marker
    fprintf(g,'menumark;');
    fclose(g)
    delete(g)
    clear g
close(gcbf)
function varargout = pushbutton11_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
```

```
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menumark;');
   fclose(g)
   delete(g)
   clear g
marker
close(gcbf)
function varargout = pushbutton10_Callback(h, eventdata, handles, varargin)
cal('pushbutton26_Callback',gcbo,[],guidata(gcbo))
```

โปรแกรมย่อย offset_load

```
function varargout = offset_load(varargin)
if nargin == 0  % LAUNCH GUI
        fig = openfig(mfilename,'reuse');
        handles = guihandles(fig);
        guidata(fig, handles);
        if nargout > 0
                varargout{1} = fig;
        end
elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK
        try
                [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
        catch
                disp(lasterr);
        end
end
function varargout = pushbutton4_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fclose(g)
   delete(g)
   clear g
cal
close(gcbf)
function varargout = pushbutton8_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
```

```
    fprintf(g,'menucal;');
    fprintf(g,'cal1;');
    fprintf(g,'calis111;');
    fprintf(g,'class11c;');
    fprintf(g,'stand;');
    fprintf(g,'loan;');
    fclose(g)
    delete(g)
    clear g
function varargout = pushbutton11_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'cal1;');
    fclose(g)
    delete(g)
    clear g
cal_cal1_2
close(gcbf)
function varargout = pushbutton12_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'cal2;');
    fclose(g)
    delete(g)
    clear g
cal_cal_2
close(gcbf)
```

```
function varargout = pushbutton14_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fprintf(g,'cal1;');
   fprintf(g,'calis111;');
   fprintf(g,'class11c;');
   fprintf(g,'stand;');
   fprintf(g,'ofld;');
   fclose(g)
   delete(g)
   clear g
function varargout = pushbutton10_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fprintf(g,'calrcvr;');
   fclose(g)
   delete(g)
   clear g
receiver_cal
close(gcbf)
function varargout = pushbutton5_Callback(h, eventdata, handles, varargin)
close(gcbf)
function varargout = pushbutton9_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
```

```
fopen(g)
   fprintf(g,'menucal;');
   fprintf(g,'corroff;');
fclose(g)
delete(g)
clear g
close(gcbf)
function varargout = pushbutton13_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fprintf(g,'resc;');
   fclose(g)
   delete(g)
   clear g
   close(gcbf)
function varargout = pushbutton16_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fprintf(g,'cal1;');
   fprintf(g,'calis111;');
   fprintf(g,'class11c;');
   fprintf(g,'stand;');
   fprintf(g,'loao;');
   fclose(g)
   delete(g)
   clear g
function varargout = pushbutton15_Callback(h, eventdata, handles, varargin)
```

```
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fprintf(g,'cal1;');
   fprintf(g,'calis111;');
   fprintf(g,'class11c;');
   fclose(g)
   delete(g)
   clear g
   cal_s11_load
   close(gcbf)
function varargout = pushbutton4_CreateFcn(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menustim;');
   fclose(g)
   delete(g)
   clear g
   stim_cal
   close(gcbf)
```

โปรแกรมย่อย offset_load_full

```
function varargout = offset_load_full(varargin)
if nargin == 0  % LAUNCH GUI
        fig = openfig(mfilename,'reuse');
        handles = guihandles(fig);
        guidata(fig, handles);
        if nargout > 0
                varargout{1} = fig;
        end
elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK
        try
                [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
        catch
                disp(lasterr);
        end
end
function varargout = pushbutton4_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fclose(g)
   delete(g)
   clear g
cal
close(gcbf)
function varargout = pushbutton8_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
```

```
    fprintf(g,'menucal;');
    fprintf(g,'cal1;');
    fprintf(g,'califul2;');
    fprintf(g,'refl;');
    fprintf(g,'class11c;');
    fprintf(g,'stand;');
    fprintf(g,'loan;');
    fclose(g)
    delete(g)
    clear g
function varargout = pushbutton11_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'cal1;');
    fclose(g)
    delete(g)
    clear g
cal_cal1_2
close(gcbf)
function varargout = pushbutton12_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'cal2;');
    fclose(g)
    delete(g)
    clear g
cal_cal_2
```

```
close(gcbf)
function varargout = pushbutton14_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fprintf(g,'cal1;');
   fprintf(g,'califul2;');
   fprintf(g,'refl;');
   fprintf(g,'class11c;');
   fprintf(g,'stand;');
   fprintf(g,'ofld;');
   fclose(g)
   delete(g)
   clear g
function varargout = pushbutton10_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fprintf(g,'calrcvr;');
   fclose(g)
   delete(g)
   clear g
receiver_cal
close(gcbf)
function varargout = pushbutton5_Callback(h, eventdata, handles, varargin)
close(gcbf)
function varargout = pushbutton9_Callback(h, eventdata, handles, varargin)
clear all
close all
```

```
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'corroff;');
fclose(g)
delete(g)
clear g
close(gcbf)
function varargout = pushbutton13_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'resc;');
    fclose(g)
    delete(g)
    clear g
    close(gcbf)
function varargout = pushbutton16_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'cal1;');
    fprintf(g,'califul2;');
    fprintf(g,'refl;');
    fprintf(g,'class11c;');
    fprintf(g,'stand;');
    fprintf(g,'loao;');
    fclose(g)
```

```
    delete(g)
    clear g
function varargout = pushbutton15_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'cal1;');
    fprintf(g,'califul2;');
    fprintf(g,'refl;');
    fprintf(g,'class11c;');
    fclose(g)
    delete(g)
    clear g
    full_load
    close(gcbf)
function varargout = pushbutton4_CreateFcn(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menustim;');
    fclose(g)
    delete(g)
    clear g
    stim_cal
    close(gcbf)
```

โปรแกรมย่อย offset_load_fullcal22

```
function varargout = offset_load_fullcal22(varargin)
if nargin == 0  % LAUNCH GUI
        fig = openfig(mfilename,'reuse');
        handles = guihandles(fig);
        guidata(fig, handles);
        if nargout > 0
                varargout{1} = fig;
        end
elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK
        try
                [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
        catch
                disp(lasterr);
        end
end
function varargout = pushbutton4_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fclose(g)
   delete(g)
   clear g
cal
close(gcbf)
function varargout = pushbutton8_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
```

```
    fprintf(g,'menucal;');
    fprintf(g,'cal2;');
    fprintf(g,'califul2;');
    fprintf(g,'refl;');
    fprintf(g,'class22c;');
    fprintf(g,'stand;');
    fprintf(g,'loan;');
    fclose(g)
    delete(g)
    clear g
function varargout = pushbutton11_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'cal1;');
    fclose(g)
    delete(g)
    clear g
cal_cal1_2
close(gcbf)
function varargout = pushbutton12_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'cal2;');
    fclose(g)
    delete(g)
    clear g
cal_cal_2
```

```
close(gcbf)
function varargout = pushbutton14_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fprintf(g,'cal2;');
   fprintf(g,'califul2;');
   fprintf(g,'refl;');
   fprintf(g,'class22c;');
   fprintf(g,'stand;');
   fprintf(g,'ofld;');
   fclose(g)
   delete(g)
   clear g
function varargout = pushbutton10_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fprintf(g,'calrcvr;');
   fclose(g)
   delete(g)
   clear g
receiver_cal
close(gcbf)
function varargout = pushbutton5_Callback(h, eventdata, handles, varargin)
close(gcbf)
function varargout = pushbutton9_Callback(h, eventdata, handles, varargin)
clear all
close all
```

```
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fprintf(g,'corroff;');
fclose(g)
delete(g)
clear g
close(gcbf)
function varargout = pushbutton13_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fprintf(g,'resc;');
   fclose(g)
   delete(g)
   clear g
   close(gcbf)
function varargout = pushbutton16_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fprintf(g,'cal2;');
   fprintf(g,'califul2;');
   fprintf(g,'refl;');
   fprintf(g,'class22c;');
   fprintf(g,'stand;');
   fprintf(g,'loao;');
   fclose(g)
```

```
    delete(g)
    clear g
function varargout = pushbutton15_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'cal2;');
    fprintf(g,'califul2;');
    fprintf(g,'refl;');
    fprintf(g,'class22c;');
    fclose(g)
    delete(g)
    clear g
    fullcal22_load
    close(gcbf)
function varargout = pushbutton4_CreateFcn(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menustim;');
   fclose(g)
    delete(g)
    clear g
    stim_cal
    close(gcbf)
```

โปรแกรมย่อย offset_load21

```
function varargout = offset_load21(varargin)
if nargin == 0  % LAUNCH GUI
        fig = openfig(mfilename,'reuse');
        handles = guihandles(fig);
        guidata(fig, handles);
        if nargout > 0
                varargout{1} = fig;
        end
elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK
        try
                [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
        catch
                disp(lasterr);
        end
end
function varargout = pushbutton4_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fclose(g)
   delete(g)
   clear g
cal
close(gcbf)
function varargout = pushbutton8_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
```

```
    fprintf(g,'menucal;');
    fprintf(g,'cal2;');
    fprintf(g,'calis111;');
    fprintf(g,'class11c;');
    fprintf(g,'stand;');
    fprintf(g,'loan;');
    fclose(g)
    delete(g)
    clear g
function varargout = pushbutton11_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'cal1;');
    fclose(g)
    delete(g)
    clear g
cal_cal1_2
close(gcbf)
function varargout = pushbutton12_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'cal2;');
    fclose(g)
    delete(g)
    clear g
cal_cal_2
close(gcbf)
```

```
function varargout = pushbutton14_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fprintf(g,'cal2;');
   fprintf(g,'calis111;');
   fprintf(g,'class11c;');
   fprintf(g,'stand;');
   fprintf(g,'ofld;');
   fclose(g)
   delete(g)
   clear g
function varargout = pushbutton10_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fprintf(g,'calrcvr;');
   fclose(g)
   delete(g)
   clear g
receiver_cal
close(gcbf)
function varargout = pushbutton5_Callback(h, eventdata, handles, varargin)
close(gcbf)
function varargout = pushbutton9_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
```

```
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'corroff;');
fclose(g)
delete(g)
clear g
close(gcbf)
function varargout = pushbutton13_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'resc;');
    fclose(g)
    delete(g)
    clear g
    close(gcbf)
function varargout = pushbutton16_Callback(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
    fprintf(g,'menucal;');
    fprintf(g,'cal2;');
    fprintf(g,'calis111;');
    fprintf(g,'class11c;');
    fprintf(g,'stand;');
    fprintf(g,'loao;');
    fclose(g)
    delete(g)
    clear g
function varargout = pushbutton15_Callback(h, eventdata, handles, varargin)
```

```
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menucal;');
   fprintf(g,'cal2;');
   fprintf(g,'calis111;');
   fprintf(g,'class11c;');
   fclose(g)
   delete(g)
   clear g
   cal2_s11_load
   close(gcbf)
function varargout = pushbutton4_CreateFcn(h, eventdata, handles, varargin)
clear all
close all
g=visa('agilent','GPIB0::16::INSTR');
set(g,'InputBufferSize',60960);
fopen(g)
   fprintf(g,'menustim;');
    fclose(g)
   delete(g)
   clear g
stim_cal
close(gcbf)
```

โปรแกรมย่อย test

```
function varargout = test(varargin)
if nargin == 0  % LAUNCH GUI
        fig = openfig(mfilename,'reuse');
        handles = guihandles(fig);
        guidata(fig, handles);
        if nargout > 0
                varargout{1} = fig;
        end
elseif ischar(varargin{1}) % INVOKE NAMED SUBFUNCTION OR CALLBACK
        try
                [varargout{1:nargout}] = feval(varargin{:}); % FEVAL switchyard
        catch
                disp(lasterr);
        end
end
function varargout = popupmenu3_Callback(h, eventdata, handles, varargin)
function varargout = ppp_Callback(h, eventdata, handles, varargin)
net2
close(gcbf)
```