ภาควิชาครุศาสตร์วิศวกรรม

คณะครุศาสตร์อุตสาหกรรม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ใบรับรองปริญญานิพนธ์

ชื่อหัวข้อ การควบคุมเครื่องมือวัดผ่านทางเครือข่ายอินเตอร์เน็ตโดยพอร์ต GPIB

Instrument Controlling via Internet by GPIB Port

ชื่อนักศึกษา 1. นายจุมพล เยรบุตร รหัสประจำตัว 41031305

2. นางสาวอรนุช สัมฤทธิ์ รหัสประจำตัว 41031332

หลักสูตร ครุศาสตร์อุตสาหกรรมบัณฑิต **สาขาวิชา** อิเล็กทรอนิกส์และคอมพิวเตอร์

อาจารย์ที่ปรึกษา อาจารย์อำพล ทองระอา

คณะกรรมการสอบปริญญานิพนธ์	ลายมือชื่อ
1. อาจารย์อำพล ทองระอา	.....
2. อาจารย์กิติพงค์ มะโน	.....
3. อาจารย์ไพบูลย์ พวงวงศ์ตระกูล	.....
4. อาจารย์สุรพงษ์ สิริพงศ์ดี	.....
5. อาจารย์สุระชัย พิมพ์สาลี	.....

วัน/เดือน/ปีที่สอบ วันจันทร์ที่ 24 เมษายน พ.ศ. 2543 เวลา 16.30 น.

สถานที่สอบ ห้อง ค.310 คณะครุศาสตร์อุตสาหกรรม สจล.

ภาควิชารับรองแล้ว

ลงนาม.....

(ผศ.วิสุทธิ์ อธิพรธรรม)

หัวหน้าภาควิชาครุศาสตร์วิศวกรรม

วันที่ 10 เดือน ก.ค. พ.ศ. 2543

ปริญญานิพนธ์

การควบคุมเครื่องมือวัดผ่านเครือข่ายอินเตอร์เน็ตโดยพอร์ต GPIB

INTRUMENT CONTROLLING VIA INTERNET BY GPIB PORT

นายจุมพล เยรบุตร
นางสาวอรนุช สัมฤทธิ์

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรครุศาสตร์อุตสาหกรรมบัณฑิต
สาขาวิชาอิเล็กทรอนิกส์และคอมพิวเตอร์
ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2542

ปริญญานิพนธ์

เรื่อง การควบคุมเครื่องมือวัดผ่านเครือข่ายอินเตอร์เน็ตโดยพอร์ต GPIB

Instrument controlling via Internet by GBIP port

ผู้จัดทำ

1. นายจุมพล เยรบุตร
2. นางสาวอรนุช สัมฤทธิ์

อาจารย์ที่ปรึกษา

ลงนาม.....

(อาจารย์อำพล ทองระอา)

ลงนาม.....

(อาจารย์พงษ์เกียรติ เชษฐพิทักษ์สกุล)

หัวหน้าภาคครุศาสตร์วิศวกรรม

ลงนาม.....

(ผศ.วิสุทธิ์ อธิพรธรรม)

ปริญญานิพนธ์

เรื่อง การควบคุมเครื่องมือวัดผ่านเครือข่ายอินเตอร์เน็ตโดยพอร์ต GPIB

Instrument controlling via Internet by GBIP port

วัตถุประสงค์ของปริญญานิพนธ์

1. เพื่อศึกษาการใช้งานโปรแกรม LabVIEW ในการควบคุมเครื่องมือวัดจากเครือข่ายคอมพิวเตอร์
2. เพื่อศึกษาการออกแบบโปรแกรม LabVIEW ในการควบคุมเครื่องมือวัดจากเครือข่ายคอมพิวเตอร์
3. เพื่อศึกษาการเขียนโปรแกรม LabVIEW ในการควบคุมเครื่องมือวัดจากเครือข่ายคอมพิวเตอร์
4. เพื่อศึกษาการทำงานของพอร์ต GPIB (IEEE-488)

ประโยชน์ที่คาดว่าจะได้รับ

1. สามารถใช้โปรแกรม LabVIEW ในการควบคุมเครื่องมือวัดจากเครือข่ายคอมพิวเตอร์
2. ได้ผังการทำงานของโปรแกรม LabVIEW
3. ได้โปรแกรม LabVIEW ที่สามารถควบคุมเครื่องมือวัดจากเครือข่ายคอมพิวเตอร์
4. เข้าใจการทำงานของพอร์ต GPIB

ชื่อหัวข้อ	การควบคุมเครื่องมือวัดผ่านเครือข่ายอินเตอร์เน็ตโดยพอร์ต GPIB
ชื่อนักศึกษา	นายจุมพล เยรบุตร
	นางสาวอรนุช สัมฤทธิ์
ชื่ออาจารย์ที่ปรึกษา	อาจารย์อำพล ทองระอา
ชื่ออาจารย์ที่ปรึกษาร่วม	อาจารย์พงษ์เกียรติ เชษฐพิทักษ์สกุล
หลักสูตร	ครุศาสตร์อุตสาหกรรมบัณฑิต
สาขาวิชา	อิเล็กทรอนิกส์และคอมพิวเตอร์
ปีการศึกษา	2542

บทคัดย่อ

ปริญญานิพนธ์ฉบับนี้นำเสนอการควบคุมเครื่องมือวัดผ่านเครือข่ายอินเตอร์เน็ตโดยพอร์ต GPIB โดยได้แบ่งโปรแกรมออกเป็น 2 ส่วนใหญ่ๆ ประกอบด้วย ส่วนที่ 1 คือ Application ที่สร้างขึ้นจากโปรแกรม LabVIEW ซึ่งในส่วนนี้สามารถแบ่งเป็นส่วนย่อยได้อีก 3 ส่วนคือ โปรแกรม Local Control เป็นโปรแกรมควบคุมเครื่องมือวัดผ่านทางพอร์ต GPIB, โปรแกรม Server เป็นโปรแกรมใช้รับคำสั่งจากเครื่องลูกข่าย (Workstation) เพื่อนำไปควบคุมเครื่องมือวัดและ โปรแกรม Workstation เป็นโปรแกรมใช้ส่งคำสั่งผ่านโพรโตคอล TCP/IP ไปยังเครื่อง Server และส่วนประกอบส่วนที่ 2 ของโปรแกรม คือ ส่วนของ Web site ที่สร้างขึ้นโดยโปรแกรม Visual Basic เพื่อส่งคำสั่งไปยังเครื่อง Server เช่นเดียวกับโปรแกรม Workstation และได้ทำการทดลองควบคุมเครื่องมือวัดโดยผ่านเครือข่ายอินเตอร์เน็ตผลปรากฏว่าได้ผลถูกต้องตามที่ได้ออกแบบไว้

Thesis Title	Instrument controlling via Internet by GPIB port
Students	Mr.Jumpol Yaraboot
	Miss.Aranuch Sumrit
Advisor	Mr.Amphon Thongra-ar
Co-Advisor	Mr.Pongkait Chetpituksakul
Education Lavel	Bachelor of Science in Industrial Education
Program in	Electronics and Computer
Acadamic Year	1999

ABSTRACT

This thesis is present instrument controlling via Internet by GPIB port. Which program to device as two big parts consist application as to build by program LabVIEW and this part can device to be three minor parts is program Local Control use to control instrument by GPIB port. program Server use to receive command from Workstation for control instruments and program Work Station use to sent command pass TCP/IP protocol to server

And parts two is Web site to build by program Visual Basic for sent command to server such program Workstation. And testing to control instrument via Internet to appear result is right to design.

กิตติกรรมประกาศ

การจัดทำปริญญานิพนธ์นี้สามารถลุล่วงไปได้ด้วยดี จากความร่วมมือของสมาชิกภายในกลุ่ม นอกจากนี้ยังได้รับความกรุณาจาก อาจารย์อำพล ทองระอา อาจารย์ที่ปรึกษาโครงงาน และอาจารย์พงษ์เกียรติ เชษฐพิทักษ์สกุล อาจารย์ที่ปรึกษาร่วม ตลอดจนอาจารย์ทุกท่านในภาควิชาที่ให้ความอนุเคราะห์ในความสะดวกทั้งด้านความรู้ สถานที่ เครื่องมือ และอุปกรณ์ต่างๆ และขอขอบคุณบุพการีที่ให้การสนับสนุนในการศึกษา และเพื่อนๆ ตลอดจนผู้เกี่ยวข้องทุกคนที่ให้ความช่วยเหลือ ให้คำแนะนำและเป็นกำลังใจในการทำงาน จนทำโครงงานนี้สำเร็จลุล่วงไปได้ด้วยดี

สารบัญ

สารบัญ (ต่อ)

สารบัญ (ต่อ)

สารบัญภาพ

สารบัญภาพ(ต่อ)

สารบัญภาพ (ต่อ)

สารบัญภาพ(ต่อ)

สารบัญตาราง

บทที่ 1
บทนำ

1.1 แนวความคิดในการทำปริญญานิพนธ์

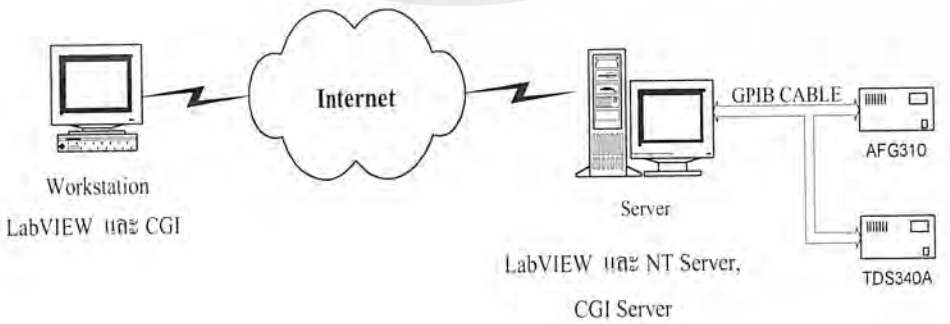
ในปัจจุบันนั้นเป็นที่ทราบกันดีว่าอุปกรณ์เครื่องมือวัดมีราคาสูง ประกอบกับเศรษฐกิจของประเทศที่ตกต่ำ ดังนั้นในการจัดซื้ออุปกรณ์เครื่องมือวัดมาใช้ในระบบการศึกษาตลอดจนหน่วยงานทั้งภาครัฐบาลและเอกชนจึงเป็นเรื่องที่ลำบาก ทำให้ขาดแคลนเครื่องมือวัดที่มีคุณภาพมาใช้งาน และประกอบกับตอนนี้เครือข่ายคอมพิวเตอร์ในปัจจุบันสามารถนำมาประยุกต์ใช้งานในรูปแบบต่างๆ ได้หลายรูปแบบและเป็นที่แพร่หลาย ดังนั้น โครงงานนี้จึงนำเอาความสามารถในการประยุกต์ใช้งานเครือข่ายอินเตอร์เน็ตมาใช้ในการติดต่อและควบคุมเครื่องมือวัดระยะไกลโดยผ่านระบบอินเตอร์เน็ต ซึ่งประกอบด้วยเครื่อง Tektronix AFG310 (Programmable Function Generator) และ Tektronix TSD340A (Programmable Oscilloscope) โดยโครงงานนี้จะเป็นการเขียนโปรแกรมจำลองลักษณะของเครื่องมือวัดทั้งสองตัว โดยใช้โปรแกรม LabVIEW ซึ่งเป็นโปรแกรมที่สามารถควบคุมเครื่องมือวัดโดยผ่านพอร์ต GPIB ได้อย่างง่าย และยังมีคุณสมบัติเป็น Web Server ได้

โครงสร้างของโครงงานนี้แบ่งเป็น 2 ส่วน คือ

ส่วนที่ 1 ส่วนควบคุมท้องถิ่น (Local Control) เป็นส่วนที่ควบคุมระหว่างเครื่องคอมพิวเตอร์กับเครื่องมือวัดโดยไม่ผ่านเครือข่ายอินเตอร์เน็ต สามารถควบคุมเครื่องมือวัดได้ 15 เครื่องต่อเครื่องคอมพิวเตอร์ 1 เครื่อง

ส่วนที่ 2 ส่วนควบคุมเครื่องมือวัดผ่านเครือข่ายอินเตอร์เน็ต

รูปแนวความคิดในการทำโครงงาน แสดงให้เห็น ดังรูปต่อไปนี้



รูปที่ 1.1 รูปแนวความคิดในการทำโครงงาน

1.2 เนื้อหาโดยสังเขป

ในปริญญานิพนธ์นี้ได้รวบรวมข้อมูลรายละเอียดทั้งทฤษฎีและหลักการทำงาน การออกแบบ การสร้าง และจัดทำคู่มือการใช้งาน ซึ่งมีเนื้อหาโดยสังเขปประกอบด้วย

บทที่ 1 บทนำกล่าวถึงความเป็นมาและความสำคัญของปริญญานิพนธ์ ขีดความสามารถของโครงงาน เนื้อหาโดยสังเขปของแต่ละบทและภาคผนวก

บทที่ 2 ทฤษฎีและหลักการ เป็นทฤษฎีพื้นฐานเกี่ยวกับการใช้งานโปรแกรม LabVIEW ทฤษฎีพื้นฐานเกี่ยวกับพอร์ต GPIB มาตรฐาน IEEE-488 ความรู้พื้นฐานเกี่ยวกับการทำงาน คุณสมบัติของ VISA, DataSocket

บทที่ 3 การออกแบบและการสร้าง กล่าวถึงการเขียนโปรแกรมควบคุมเครื่องมือวัดผ่านเครือข่ายอินเตอร์เน็ต หลักการเขียนโปรแกรม LabVIEW เพื่อติดต่อกับเครื่องมือวัดผ่านเครือข่ายอินเตอร์เน็ตโดยพอร์ต GPIB รวมทั้งการส่งและรับคำสั่งผ่านเครือข่ายอินเตอร์เน็ต และการสร้าง Web page

บทที่ 4 การทดลองและผลการทดลอง กล่าวถึง ลักษณะโปรแกรมและการใช้งานโปรแกรม, ลักษณะอุปกรณ์และการติดตั้ง, การทดลองและผลการทดลองโปรแกรมโดยละเอียด

บทที่ 5 สรุปอภิปรายและเสนอแนะ กล่าวถึงข้อสรุป ปัญหา แนวทางการแก้ไข และแนวทางในการพัฒนาเพื่อนำไปประยุกต์ใช้งานได้อย่างกว้างขวาง

1.3 วัตถุประสงค์ของปริญญานิพนธ์

1. เพื่อศึกษาการใช้งานโปรแกรม LabVIEW ในการควบคุมเครื่องมือวัดจากเครือข่ายคอมพิวเตอร์
2. เพื่อศึกษาการออกแบบโปรแกรม LabVIEW ในการควบคุมเครื่องมือวัดจากเครือข่ายคอมพิวเตอร์
3. เพื่อศึกษาการเขียนโปรแกรม LabVIEWในการควบคุมเครื่องมือวัดจากเครือข่ายคอมพิวเตอร์
4. เพื่อศึกษาการทำงานของพอร์ต GPIB (IEEE-488)

1.4 ขีดความสามารถของโครงงาน

1. สามารถควบคุมเครื่อง SONY Tektronix AFG310 จากเครือข่ายคอมพิวเตอร์ได้
2. สามารถควบคุมเครื่อง SONY Tektronix TSD340 จากเครือข่ายคอมพิวเตอร์ได้
3. มีใบงานบอกวิธีการใช้โปรแกรม LabVIEW ในการควบคุมเครื่องมือวัด 4 ใบงาน

1.5 ประโยชน์ที่คาดว่าจะได้รับ

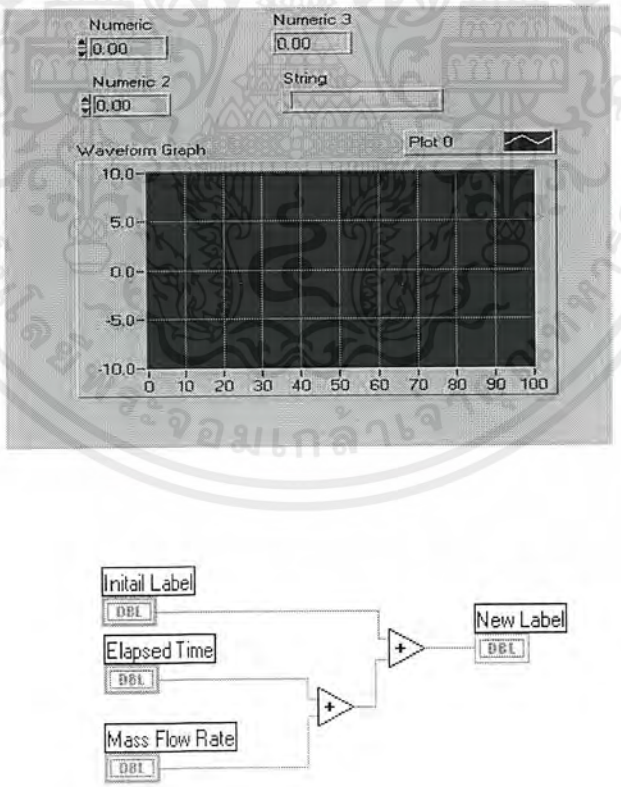
1. สามารถใช้โปรแกรม LabVIEWในการควบคุมเครื่องมือวัดจากเครือข่ายคอมพิวเตอร์
2. ได้ผังการทำงานของโปรแกรม LabVIEW
3. ได้โปรแกรม LabVIEW ที่สามารถควบคุมเครื่องมือวัดผ่านเครือข่ายคอมพิวเตอร์
4. เข้าใจการทำงานของพอร์ต GPIB

บทที่ 2
ทฤษฎีและหลักการ

2.1 การใช้งานโปรแกรม LabVIEW

2.1.1 ความหมาย

LabVIEW (Laboratory Visual Engineering Workbench) เป็นโปรแกรมที่ใช้สำหรับการระบบการวัดและการวิเคราะห์ LabVIEW เป็นโปรแกรมที่สามารถสร้าง application เหมือนภาษาโปรแกรมทั่วไป ข้อแตกต่างระหว่างการสร้างงานที่ใช้ LabVIEW สร้างและใช้ภาษาอื่นสร้างที่เห็นได้ชัด คือ ภาษาโดยทั่วไปจะเขียนเป็นชุดคำสั่งแบบใช้ตัวอักษร แต่ใน LabVIEW จะเป็นภาษาโปรแกรมมิ่งแบบกราฟฟิก(Graphical Programming language) G การเขียนจะเป็น block-diagram (คล้ายการเขียน flowChart) ดังรูปที่ 2.1



รูปที่ 2.1 ลักษณะโปรแกรม LabVIEW

2.1.2 ความสามารถของโปรแกรม

1) มี Library เกี่ยวกับ Data-acquisition ที่เรียกใช้

2) การควบคุมและการสั่งงาน GPIB และอุปกรณ์ผ่านสาย serial

3) การจัดการ Data เช่น Data analysis (วิเคราะห์ข้อมูล) Data presentation และ Data storage (จัดการเกี่ยวกับ file) LabVIEW มีความสามารถเหมือนการพัฒนาโปรแกรมทั่วไป เราสามารถตั้ง breakpoint ตรวจดูการทำงานที่ละบรรทัด (single step) และสามารถทำให้เราเห็นการทำงานของโปรแกรมที่เราสร้างขึ้น

4) เราสามารถแสดงผล Data ที่เราใช้เป็น Graphic โดยใช้ Chart, Graph

2.1.3 การทำงาน

LabVIEW โปรแกรมจะถูกเรียกว่า Visual Instrument (VIs) เพราะสิ่งที่ปรากฏบนหน้าจอเรียนแบบเครื่องวัด LabVIEW มีความสามารถเหมือนกับภาษา computer ทั่วไป เราสามารถสร้าง VIs ย่อยได้และสามารถส่งผ่าน Data เข้าไปใน VIs ย่อย (subVI เหมือนเป็น ฟังก์ชันย่อย) ได้ VIs ประกอบด้วย 3 ส่วนใหญ่ คือ

1) Front panel คือส่วนที่ติดต่อกับผู้ใช้ (User interface) เป็นส่วนที่จำลองเครื่องวัดภายในนี้สามารถรับ input จากผู้ใช้ได้และแสดงผลลัพธ์ ดังรูปข้างบน (ส่วน User interface)

2) Block diagram การทำงานของ Program แบบ Graphic ประกอบด้วย Block diagram, icons, subVI Function ที่มีไว้ให้ใช้และ Loop เราต้องทำการลากเส้นเพื่อเชื่อมแต่ละส่วนให้เป็นไปตามที่เราต้องการ ดังรูป (ส่วนที่เป็น Graphic code)

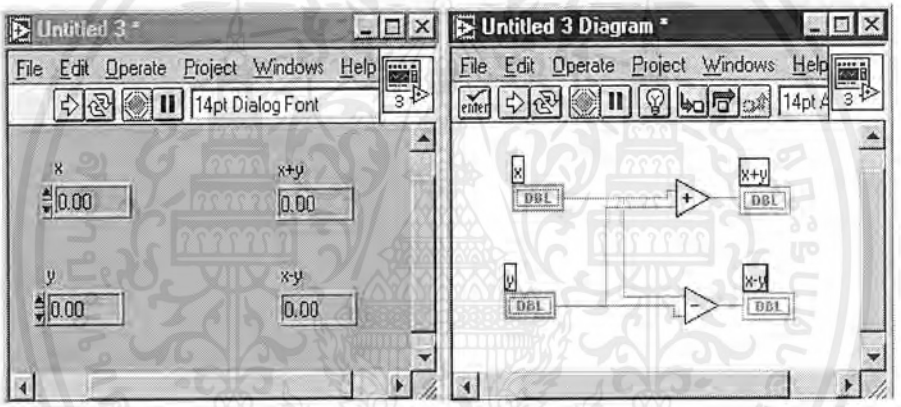
3) Icon และ Connector อนุญาติให้เราส่งข้อมูลเข้าออกให้ VIs ทำงาน icon แทน VI ส่วน connector จะแทน input และ output ของ VI

รูปที่ 2.2 Icon และ Connector ของ LabVIEW

2.1.4 ส่วนประกอบของ LabVIEW

1) Front Panel เป็นส่วนที่ติดต่อกับผู้ใช้ เราต้องเลือก control ให้เหมาะสมกับจุดมุ่งหมายของเรา ใน Front Panels มีคำจำกัดความ 2 ข้อ ความที่สำคัญคือ control คือ input device ในเครื่องวัดทั่วไป เช่น knobs และ switch ส่วน Indicators คือส่วนที่แสดงผล เราสามารถที่จะเอา Control และ Indicators มาใช้โดยเลือก menu Controls เราสามารถที่จะปรับปรุงขนาดรูปร่าง และตำแหน่งที่ได้โดยง่าย

2) Block diagram จะเก็บ Code ด้าน Graphic ของ LabVIEW VI Block diagram ของ LabVIEW จะเหมือนการเขียนโปรแกรมโดยใช้ภาษา computer เราสามารถที่จะสร้าง Block diagram โดยเชื่อมสายของแต่ละ Object ตามจุดประสงค์ของเรา ดูตัวอย่างดังรูปที่ 2.3



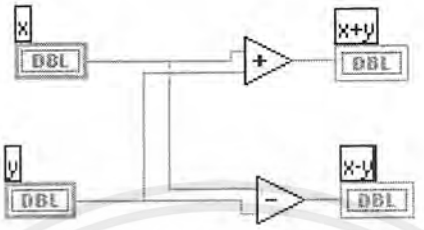
รูปที่ 2.3 ตัวอย่างของโปรแกรม LabVIEW การบวกเลข 2 ตัว

3) Terminals เมื่อเราเอา control หรือ indicator วางใน front panel LabVIEW จะสร้าง Terminal ใน Block diagram ให้อัตโนมัติ ใน Block diagram เราไม่สามารถลบ control หรือ indicator ได้ เราต้องลบใน front panel

ข้อสังเกต Terminal จะมีขนาดขอบหนา ส่วน indicator จะมีขอบบาง

4) Nodes เป็นส่วนที่ใช้ในการสั่งงานโปรแกรมให้เป็นไปตามที่เราต้องการ ประกอบด้วย statements, operators, Functions, subroutine, structure (เงื่อนไข และ Loop ต่างๆ) LabVIEW จะมี Nodes พิเศษ Formula Nodes ไว้ใช้ในกรณีที่ไม่สะดวกในการแทนสูตรยากๆ ด้วย Nodes มากๆ โดยแทนเป็นชุดคำสั่งแบบ text

5) Wires เป็นการเชื่อมระหว่าง Terminal ต้นทาง และ Terminal ปลายทาง เราต้องเชื่อมสายระหว่าง source ไปยัง destination เท่านั้น แต่ละ Wire จะมีสีและรูปร่างแตกต่างกัน ขึ้นกับชนิดของค่าที่แทน



รูปที่ 2.4 Wires Stiles

6) Dataflow Programming เนื่องจาก LabVIEW เป็นการเขียนโปรแกรมแบบ Graphic การทำงานของ LabVIEW จะเป็นแบบ Dataflow (จะทำงานก็ต่อเมื่อมี Data เข้ามา)

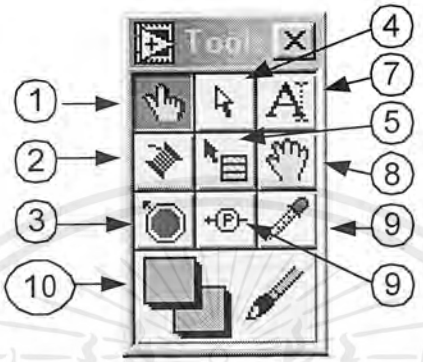
6.1) Icon และ Connector เมื่อมี subVI (VI ที่อยู่ใน VI อื่น) control และ indicator ของ subVI จะรับและส่ง Data คืนไปให้ VI ที่เรียกมา connector คล้ายกับ parameter ที่ส่งเข้าไปให้กับฟังค์ชั่น

รูปที่ 2.5 default icon ใน LabVIEW

ทุกๆ VI จะมี default icon ซึ่งจะแสดงอยู่ด้านบนขวาทั้ง panel และ diagram ทุกๆ VI จะมี connector ซึ่งเราสามารถดูได้จาก Show connector บน pop-up menu บน panel

- Pop-up Menus เราสามารถเรียก Pop-up menu ได้โดย mouse click ขวากับสิ่งที่เราสนใจ โดยทำการเปลี่ยนแปลงสิ่งนั้น เช่น เปลี่ยนสิ่งนั้นให้เป็น control หรือ Terminal เปลี่ยนชนิดข้อมูล, ให้แสดง Terminal หรือ icon แทนที่สิ่งนั้นด้วยสิ่งใหม่ (replace)

-Edit Mode และ Run Mode VI สามารถเป็นได้ทั้ง edit mode/ Run mode edit mode เราสามารถสร้างหรือแก้ไข VI ได้ ส่วน Run mode เราไม่สามารถแก้ไขได้ใน Edit mode palette จะเป็นดังรูปที่ 2.6



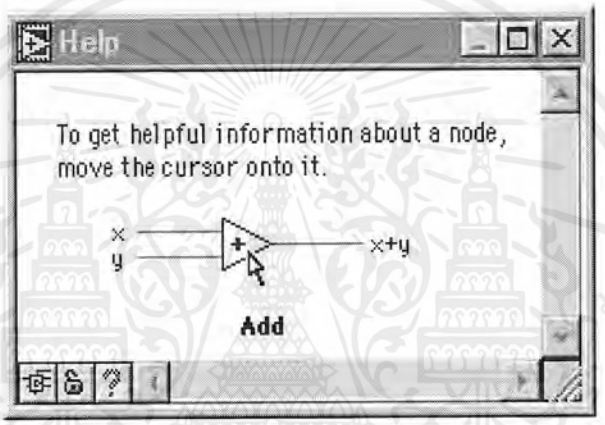
รูปที่ 2.6 Tool Palette Edit Mode

1. Operating tool ให้เราสามารถเปลี่ยนค่าใน front panel control ใน Run mode เท่านั้น และ indicator ใน edit mode
2. Wiring tool จะ wire สิ่งต่างๆ เข้าด้วยกันใน Block diagram
3. Set/Clear Breakpoints เซ็ตจุดที่ต้องการให้หยุดการทำงานเพื่อการตรวจสอบ
4. Positioning tool เครื่องมือในการเลื่อนตำแหน่ง เราสามารถเลือก,เคลื่อนย้าย และเปลี่ยนขนาด
5. Pop-up tool ใช้เปิดเมนูของ Control/Functions
6. Probe Data tool คอยตรวจจับข้อมูลในสายเชื่อมต่อกันระหว่าง Object
7. Edit text tool เครื่องมือเขียนตัวอักษร
8. Scroll tool ใช้ในการเลื่อนหน้าเอกสารทั้ง front panel และ Block diagram
9. Get Color tool ดึงสีที่ต้องการในตัง VI
10. Set color tool ใช้เปลี่ยนสีต่างๆใน front panel ของ Object และ backgrounds ใน front panel และ Block diagram จะมี tool bar ดังรูปที่ 2.7

รูปที่ 2.7 Toolbars

ปุ่มรัน เราสามารถเริ่มการ Execute ดังรูปที่ 2.7(1) โดยกดปุ่มแรก เมื่อรันอยู่จะเป็นลูกศรสีดำ ดังรูปที่ 2.7(5) ถ้าเป็นลูกศรที่หัก หมายถึง รัน ไม่ผ่าน รูปที่ 2.7(2) หมายถึง รัน แบบต่อเนื่อง รูปที่ 2.7(3)และรูปที่ 2.7(7) หยุดการรัน รูปที่ 2.7(4) หมายถึง หยุด ชั่วขณะ รูปที่ 2.7(9) มอนิเตอร์โปรแกรมขณะรัน รูปที่ 2.7 (10,11,12) เป็นการรันทีละขั้น

- LabVIEW Help windows จะช่วยเราเกี่ยวกับข้อมูลของ Function.ค่าคงที่. sup VIs control และ indicator เพื่อที่เราจะ wire สายให้เหมาะสมกับชนิดของข้อมูล



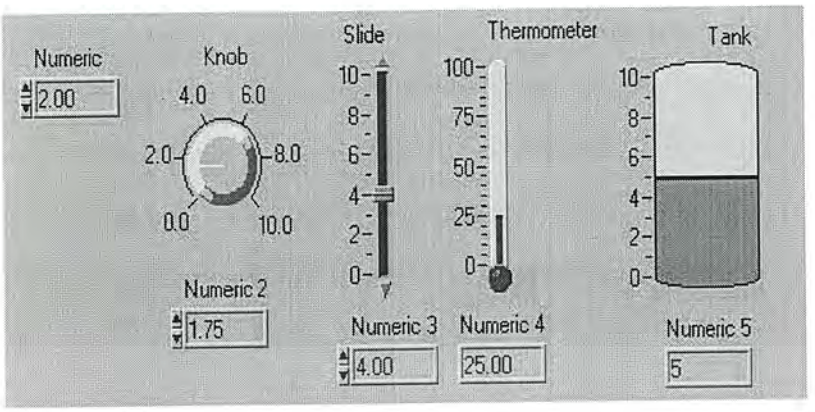
รูปที่ 2.8 LabVIEW Help windows

2.1.5 การใช้ LabVIEW

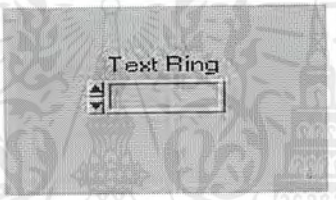
LabVIEW ประกอบด้วย control และ indicator อย่างง่ายๆ 4 ชนิด คือ numeric Boolean string และ path ซึ่งจะอธิบายต่อไป

1) Numeric Control และ Indicator

Numeric Control อนุญาติให้เราป้อน input ใน VIs ได้ ส่วน numeric indicator จะแสดงค่าตัวเลขให้เห็น รูปที่ 2.9 แสดง numeric บางชนิด ทั้งหมดอาจเป็นทั้ง control หรือ indicator เมื่อสร้าง control หรือ indicator จะมี numeric Terminal ใน Block diagram ให้ numeric control และ indicator จะมี default ให้แต่เราสามารถเปลี่ยน type ได้โดยแก้ใน format & Precision ใน pop-up (เมื่อคลิกขวาสิ่งที่ต้องการเปลี่ยน)



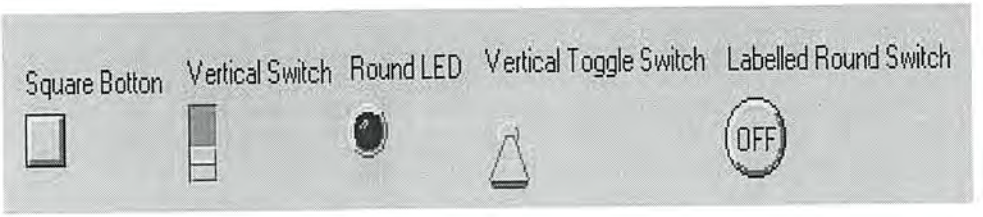
รูปที่ 2.9 Numeric Object



รูปที่ 2.10 Text Ring

Text Ring เป็น numeric ชนิดพิเศษ โดยข้อความจะมีความสำคัญกับตัวเลขเราสามารถดูความสัมพันธ์ระหว่างข้อความกับตัวเลขโดยที่ pop-up menu ของ Text Ring เลือก Add Item after หรือ Add Item Before 1

Boolean มีความหมายแค่ 2 ชนิด หรืออาจกล่าว จริงหรือ เท็จ LabVIEW มี Boolean หลายชนิด เราสามารถเปลี่ยนสถานะของ Boolean ได้โดยคลิกที่ Boolean นั้นๆ เช่น



รูปที่ 2.11 Boolean ของ LabVIEW

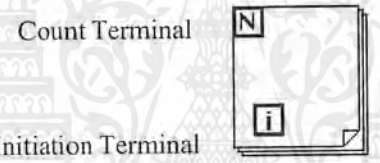
2) โครงสร้างของภาษา

โครงสร้างของภาษาแบ่งได้ 4 ชนิดคือ

1. โครงสร้างการทำซ้ำ (Repetition Structure)
2. โครงสร้างการเลือก(Selection Structure)
3. โครงสร้างลำดับ (Sequence Structure)
4. โครงสร้างสมการคณิตศาสตร์ (Formula Node)

2.1) โครงสร้างการทำซ้ำ (Repetition Structure)

การทำซ้ำจะมี 2 ชนิด คือ for และ while Loop for Loop จะวน Loop จนกว่าจะครบจำนวนรอบที่กำหนด ส่วน while จะออกจาก Loop เมื่อตรวจพบว่า condition ที่เป็นเท็จ เราสามารถหา Loop ใน LabVIEW ใน menu Structure & Constants ที่อยู่ใน Function ใน sub diagram For Loop มีลักษณะดังรูปที่ 2.12



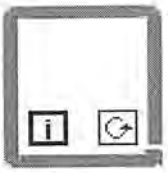
รูปที่ 2.12 For Loop

จำนวนครั้งที่วนได้ (Iteration Terminal) ระหว่างรอบแรกจะเป็น 0 และจะหยุดเมื่อ N-1 แล้วถ้าป้อนศูนย์ให้จำนวนครั้งทั้งหมดที่วน Loop (count Terminal)นั้นจะไม่วนเลย For Loop จะมีค่าเท่ากับ

for I = 0-N-1

Execute subdaigram

While Loop มีลักษณะตามรูปที่ 2.13



Initiation Terminal condconditional

รูปที่ 2.13 while Loop

ออกจาก Loop จนกว่า conditional Terminal จะมีค่าเป็น FALSE ค่า default value ของ condition Terminal มีค่าเป็นเท็จ เพราะฉะนั้นถ้าไม่ได้ wire สายของ condition Terminal จะทำงานใน Loop เพียง 1 ครั้ง while Loop มีค่าเท่ากับ

Do Loop

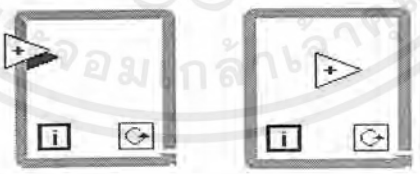
Execute subdiagram (which sets condition)

While condition is TRUE

ข้อสังเกต

- เราสามารถขยายขนาดของ Loop ได้โดยการคลิกที่ Loop แล้วเลือกไปที่มุมขยายขนาดตามต้องการ

- เมื่อเรามี Loop เราต้องการนำ Objects อื่นเข้ามาใส่ใน Loop เพื่อใช้งาน ให้คลิกที่ Object ที่ต้องการ แล้วลาก (drag) มาไว้ใน Loop ถ้าเราเลือก structure เข้าไปที่ Object สังเกต Object จะมีเงาดำหนาแสดงว่า Object ตัวนั้นไม่อยู่ใน Loop ดังรูป 2.14

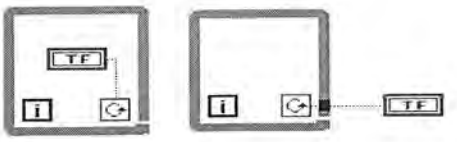


Not in Loop in Loop

รูปที่ 2.14 ความแตกต่างระหว่าง Object ที่อยู่ใน Loop และไม่อยู่ใน Loop

- Input ที่เข้ามาใน Loop จะส่ง Data เข้ามาก่อน Loop จะ execute ส่วน Loop output 0จะส่ง Data ออกได้ก็ต่อเมื่อออกจาก Loop แล้ว เราต้องใส่ Terminal ใน Loop ถ้าเราต้องการการตรวจ

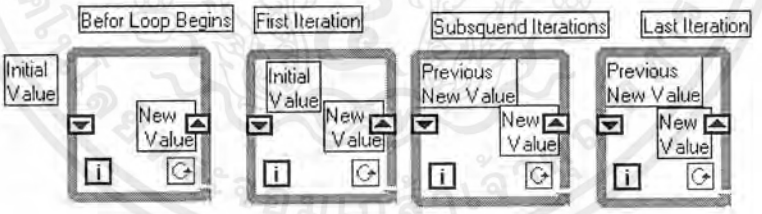
สอบในแต่ละครั้ง ดังรูป 15a ถ้าเรานำเอา Terminal มาไว้นอก Loop รูป 15b Loop นั้น อาจวน Loop infinite หรือ วน Loop เพียง 1 ครั้ง (ขึ้นอยู่กับค่าเริ่มต้นของ Boolean)



รูปที่ 2.15 การเอา Terminal มาไว้ใน/นอก structure

Shift Registers

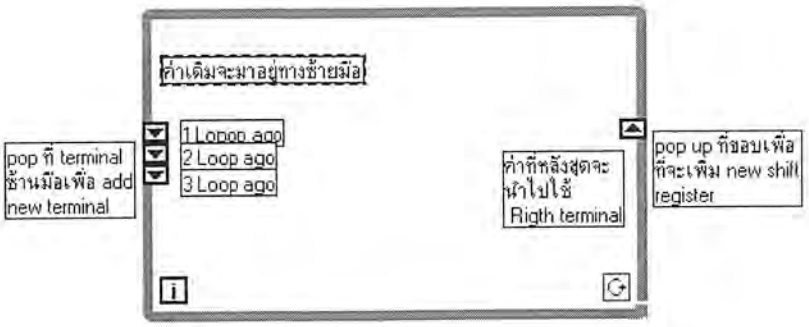
Shift Registers (มีไว้สำหรับ For หรือ while Loop) เป็น local variable ซึ่งย้ายค่าจากตำแหน่งหนึ่งไปยังอีกตำแหน่งหนึ่งในแต่ละครั้งของการวน Loop มันมีความสำคัญต่อ LabVIEW ซึ่งเป็น Program Graphic structure มาก เราสามารถสร้าง Shift Register ได้โดย เรียก Pop-up menu บนขอบบนของ Loop แล้วเลือก Add Shift Register จาก pop-up menu จะปรากฏ Terminal 1 คู่ Shift Register จะสามารถเก็บค่าได้หลาย type เช่น numeric, Boolean, Array และอื่นๆ จะเป็นค่าใดนั้นขึ้นกับว่า Object ในจะเป็นตัวเลขที่ wire เข้ากับ Terminal Terminal ขวามือของ Shift Register จะเก็บข้อมูลที่ได้ในแต่ละครั้งของการวน Loop (ถ้ามีการ wire เข้ากับ Terminalขวามือ) และ Data จะปรากฏทาง Terminal ซ้ายมือในอีก Loop ถัดไป ดังรูปที่ 2.16



รูปที่ 2.16 Shift Register

เราสามารถทำให้ Shift Register จำค่าเดิมจาก previous iteration ได้หลายค่า โดย pop-up ที่ Terminal ซ้ายมือของ Shift Register เลือก Add Element ดังรูปที่ 2.17

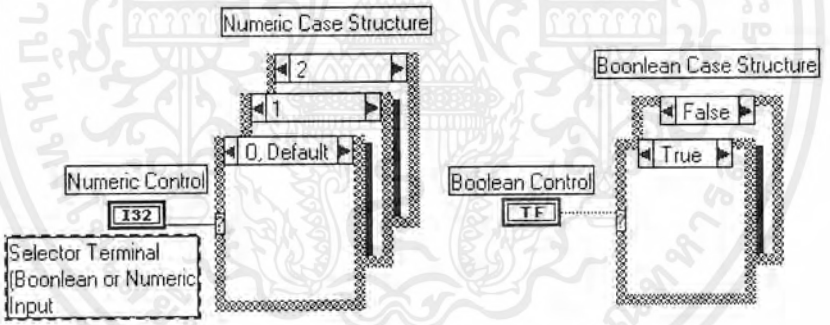
ค่าของ Shift Register ถ้าไม่มีการ initializes จะใช้ค่าเดิมจากคราวที่แล้วทำให้ไม่ตรงกับความต้องการของเราได้



รูปที่ 2.17 Shift Register หลาย Element

2.2) โครงสร้างการเลือก (Selection Structure)

Case Structure ใน LabVIEW หาได้จาก Structure & Constants ใน Function menu ใน case นี้สามารถเป็นได้ทั้ง if-then-else และ case ธรรมดา ขึ้นอยู่กับ input ที่ wire ให้ selector Terminal ว่าเป็น Boolean หรือ numeric ถ้าเป็น Boolean case จะทำหน้าที่เหมือน if-then-else โดยมีเงื่อนไขได้มากกว่า มีเงื่อนไขตั้งแต่ 2^{11} เงื่อนไขค่าเริ่มต้นเพียง 0 และ 1 ดังรูปที่ 2.18

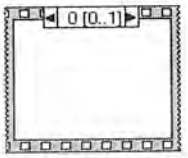


รูปที่ 2.18 Case Structure

ถ้าเรา Wire floating point numeric เข้ากับ selector LabVIEW จะทำการประมาณค่าให้เป็นเลข Integer ที่มีค่าใกล้เคียงที่สุด ถ้าป้อนค่าติดลบ LabVIEW จะทำให้เป็น 0 มากกว่า case ที่มี LabVIEW จะปัดให้เป็นค่า case ที่สูงสุดให้ ถ้าเราต้องการเพิ่มเงื่อนไข caseทำได้โดยทำการ pop-up ที่ขอบล่างของ case แล้วเลือก Add Case After หรือ Add Case Before ถ้าเราต้องการ copy case ทำได้โดยการเลือก Duplicate Case ถ้าเราต้องการลบ case เลือก Remove case ที่ pop-up menu

2.3) โครงสร้างลำดับ (Sequence Structure)

เราสามารถทำให้ LabVIEW ทำงานเป็นลำดับขั้นตอนเหมือนการเขียนโปรแกรมภาษาต่างๆ โดยใช้ Sequence structure ซึ่งเรียกใช้ได้จาก structures & constants ใน Function menu ดังรูป 2.19

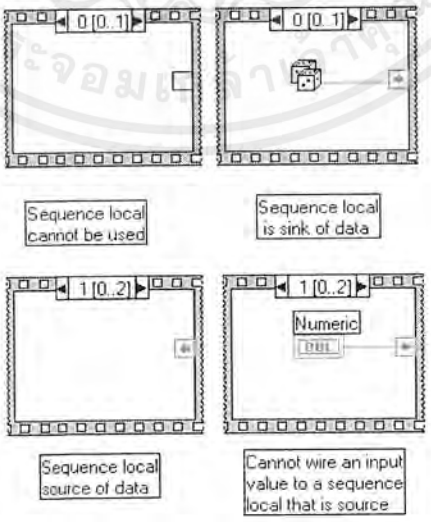


รูปที่ 2.19 Sequence structure

Sequence structure ทำงานทีละเฟรม โดยทำตั้งแต่ frame ตามลำดับ เราสามารถเพิ่ม sequence โดยเลือก pop-up ที่ของ structure เลือก Add frame After หรือ Add frame before

Output ของ sequence structure สามารถมีได้แค่เพียง 1 ซึ่งต่างจาก case ซึ่งต้องมี output ต่อ 1 source/case แต่ output ของ sequence จะออกมาเมื่อ sequence เสร็จแล้วเท่านั้น

Sequence Local เป็นตัวแปรที่ส่งให้เฉพาะ sequence structure เราสามารถสร้าง Sequence local ได้โดย pop-up ที่ขอบของ sequence structure เลือก Add sequence local จะปรากฏดังรูปที่ 2.20 เมื่อเรา wire ข้อมูลเข้าไปที่ sequence local จะปรากฏลูกศรขึ้นดังรูปที่ 2.20 และถ้าไปที่ Frame อื่นที่อยู่ใต้ Frame นี้จะสามารถนำข้อมูลที่ส่งไปใช้ได้ แต่ Frame อื่นที่ไม่ได้อยู่ใต้ frame นี้สามารถนำไปใช้ได้ดังรูปที่ 2.20



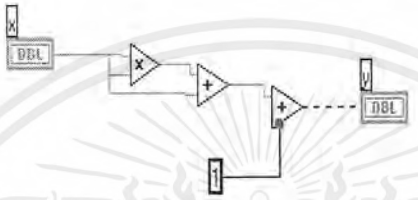
รูปที่ 2.20 Sequence Local Terminal

Timing

บางครั้งถ้าเราต้องการจัดการเกี่ยวกับเวลา เช่น ทำการหน่วงเวลา LabVIEW จะอยู่ที่ Dialog & Date/ Time ใน Function wait (ms) จะรอตามระยะเวลาที่กำหนด ส่วน Tick count (ms) ประมาณ 55 ms

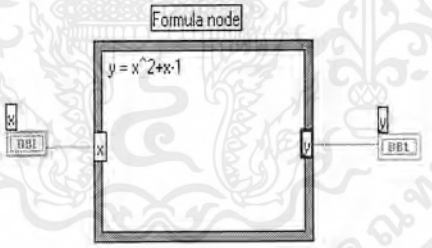
2.4 โครงสร้างสมการคณิตศาสตร์ (Formula Node)

เพื่อแทนสูตรการคำนวณที่ซับซ้อน ๆ เช่น $y=x^2+x+1$ ถ้าทำแบบตรงๆ จะเป็นดังรูปที่ 2.21



รูปที่ 2.21 การสสร้าง LabVIEW ที่แทนสมการ $y = x^2 + x + 1$ วิธีธรรมดา

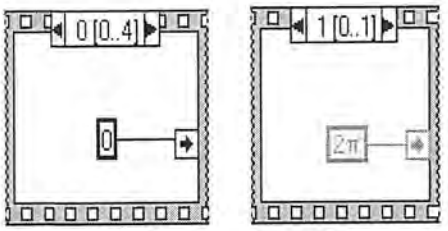
แต่ถ้าใช้ Formula node จะง่ายกว่าดังรูปที่ 2.22



รูปที่ 2.22 LabVIEW ที่แทนสมการ $y = x^2 + x + 1$ วิธี Formula node

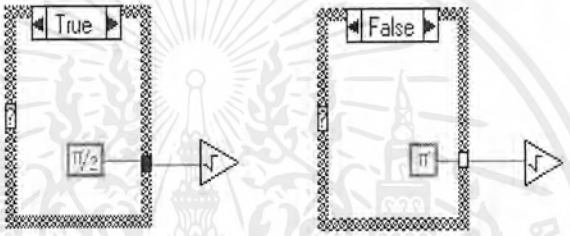
ปัญหาการ Wire สาย

- ใน Sequence Local มีการให้ค่าเกินมากกว่า 1 ค่า ถ้าเราพยายามที่จะให้ค่าอื่นแก่ตัวแปรเดียวกัน (แต่คนละ frame) จะเกิด error



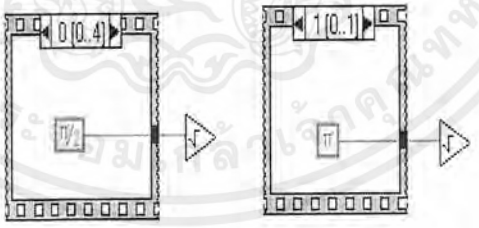
รูปที่ 2.23 ปัญหาการ Wire สาย เนื่องจาก multiple source error

- ไม่ได้ Wire สาย ได้ค่าที่ออกมาจาก tunnel ในทุกๆ case ของ case structure



รูปที่ 2.24 ปัญหาการ Wire สาย เนื่องจาก output ที่ออกจาก tunnel ใน case ไม่ออกมาทุกกรณี

- การให้ค่าแก่ Tunnel บน sequence structure เดียวกัน



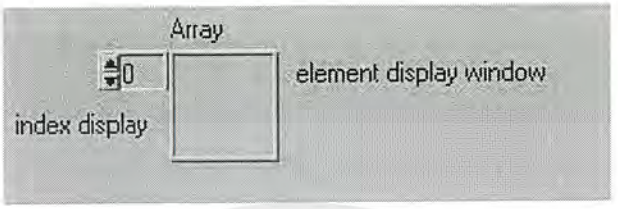
รูปที่ 2.25 การให้ค่าแก่ Tunnel1 บน sequence structure เดียวกัน

3) Array and Clusters

Array คือ ข้อมูลชนิดเดียวกันที่เก็บเป็นชุด Array สามารถมีได้หลายขนาด หลายมิติ Array ใน LabVIEW สามารถเป็น type อะไรก็ได้ยกเว้น Array, Chart หรือ Graph เราสามารถอ้างตำแหน่งของ Array ได้ โดยใช้ index index จะมีค่าอยู่ระหว่าง 0 ถึง N=1 ซึ่ง N เป็นจำนวนสมาชิกใน Array

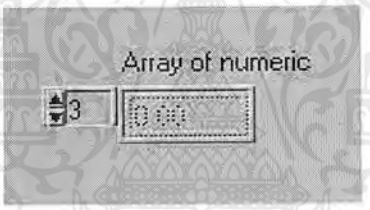
3.1) การสร้าง Array control และ Indicators

เลือก Array ใน Array & Graph ใน Control menu จะปรากฏดังรูปที่ 2.26



รูปที่ 2.26 Array

การสร้าง Array ให้คลิกขวาเรียก pop-up menu เลือกชนิดของ Array เช่นจากข้างต้นเรา เลือกชนิดข้อมูลเป็นตัวเลขจะเป็นดังรูปที่ 2.27



รูปที่ 2.27 Array of numeric

ในทำนองเดียวกันกับ Array ใน diagram เริ่มแรก Array จะไม่มี type และหลังจากเราให้ Array type เป็น numeric จะปรากฏดังรูปที่ 2.28

รูปที่ 2.28 Array Diagram

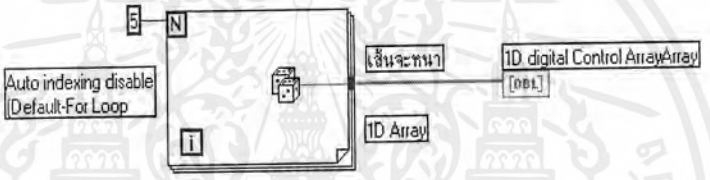
two-dimensional Arrays

Array 2 มิติ ประกอบด้วย column และ row เราสามารถเพิ่ม dimensions โดย pop-up ที่ Array แล้ว เลือก Add Dimension

รูปที่2.29 Array 2 มิติ

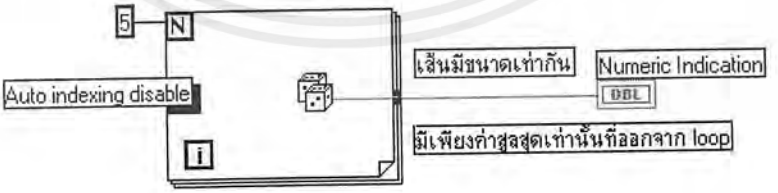
3.2) การสร้าง Array โดยใช้ Auto-Indexing

For loop และ while loop สามารถทำ Auto-Indexing Array ที่ขอบของมัน หลังจากวน loop เสร็จ Array จะถูกส่งออกจาก for loop ไปยัง indicator เส้นที่ออกมาจาก loop จะหนา



รูปที่ 2.30 การสร้าง Array โดยใช้ auto- indexing

เราต้องการเอาค่าออกมาจาก Loop โดยไม่ต้องสร้าง Array เราทำได้โดยวิธี disable auto-indexing โดยเรียก menu pop-up ที่ tunnel (บริเวณสี่เหลี่ยมผืนผ้าสีดำ) เลือก Disable Indexing หลังจาก disable จะมีเพียงค่าสุดท้ายที่ออกมาจาก loop ดังรูปที่ 2.31

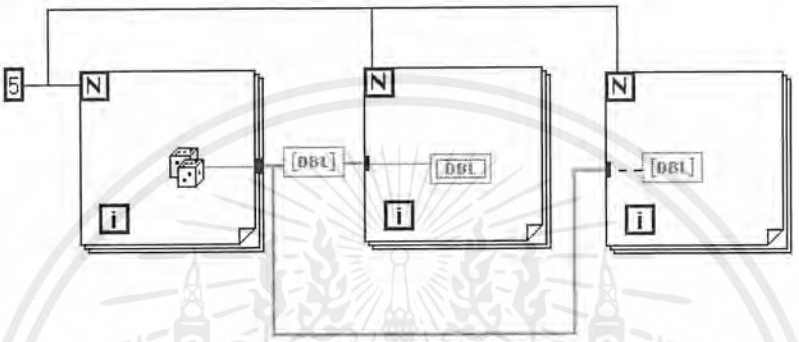


รูปที่ 2.31 การ Disable auto indexing

เมื่อเรานำค่า Array ใส่ใน Loop จะแบ่งเป็น 2 กรณี

1. Array ใส่ในตัวธรรมดา ค่าที่อยู่ใน Loop จะเป็นค่าของ Array indexes ที่ I (จำนวนรอบที่วน) ถ้า I มากกว่าค่า Array size ค่าที่อยู่ใน Loop จะเป็นค่าตัวสุดท้าย

2. Array ใส่ใน Array เราต้อง disables indexing เหมือนเราใส่ Array ทั้งหมดใน Loop ที่เดียว ดังรูปที่ 2.32



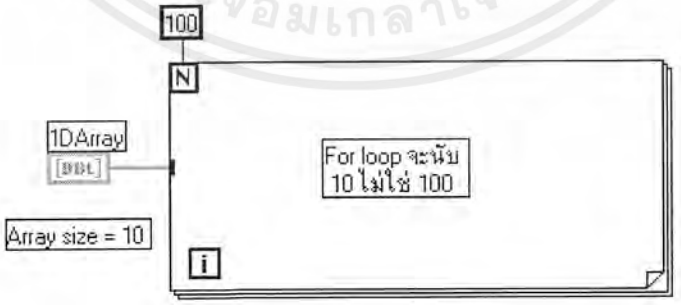
รูปที่ 2.32 การ Disables indexing

3.3) การสร้าง Array 2 มิติ

เราสามารถใช้ For Loop 2 อันในการสร้างArray 2 มิติได้โดย Loop ที่อยู่ด้านในจะสร้างแถว (row) Loop ที่อยู่นอกจะสร้างหลัก (column)

3.4) การสร้าง Auto Indexing เพื่อที่จะ set การวน Loop

ถ้าการวน Loop มากกว่า Array size จะยึดเอา Array size โดยจะเป็นดังรูปที่ 2.33



รูปที่ 2.33 การสร้าง Auto-Indexing เพื่อที่จะ Set การวน Loop

3.5) Function ที่ใช้ในการสร้าง Array

จะอยู่ใน Diagram Function Array & Cluster

- Initialize Array จะสร้างและ Initialize Array dimension ให้เป็นตามค่าที่เราต้องการ
- Array size จะคืนค่าจำนวน element ที่อยู่ใน input Array
- Build Array จะต่อหรือรวม Array 2 Array หรือเพิ่ม element เข้าไปให้กลายเป็น 1 Array Build จะมี input ได้ 2 ชนิด คือ Array และ element
- Array Subset จะ return Array ที่เริ่มที่ index และมีความยาว Length elements
- Index Array จะให้ค่าที่ตำแหน่งที่ออกมาให้

3.6) Polymorphism

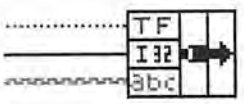
ใน Function การคำนวณของ LabVIEW เช่น ADD, Multiply, Device จะมีกฎโดย input สามารถมีได้หลาย Data type ตัวอย่าง polymorphism ของการบวก

ตารางที่ 2.1 ตัวอย่าง Polymorphism ของการบวก

Combination	Result
Scalar+scalar	Scalar
Scalar+Array	Array โดยนำเอา scalar บวกทุก element ใน Array
Array+Array	Array

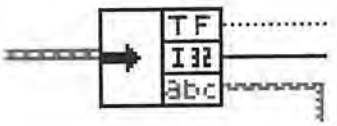
3.7) Clusters

Clusters คล้าย Array คือ เป็นการ group Data แต่ Clusters จะต่างจาก Array ตรงที่สามารถ group Data ที่มี type ต่างกัน เหมือนโครงสร้าง record ใน Pascal หรือ structure ใน C โดยการทำ Cluster ทำได้โดย bundle (มัด) wire แต่ละ wire จะแทน element ที่ต่างกันของ Cluster wire จะเหลือเส้นเดียว ทำให้สามารถสะดวก เข้าใจง่าย และสามารถนำไปใช้ได้กลับ subVIs Cluster สามารถต่อหลายๆ Cluster ได้



รูปที่ 2.34 Bundle (มัด)

เมื่อต้องการจะ Access ค่าที่อยู่ใน Cluster โดยทำ Unbundled



รูปที่ 2.35 Bundling Data และ Unbundling Data

3.8) การสร้าง Cluster Controls และ indicators

โดยเลือกจาก Cluster shell (Array & Graph ที่ Control menu) หลังจากนั้นเลือก type ที่จะทำ Cluster โดยคลิกขวาเลือก type ชนิดต่างๆที่ต้องการจะทำ Cluster Object ชนิดต่างๆที่อยู่ใน Cluster ต้องการ control ทั้งหมด หรือ indicator ทั้งหมด

Cluster จะมี logical order ที่ไม่มีความสัมพันธ์ในตำแหน่งของมันใน shell Object แรกที่อยู่ใน Cluster จะเป็น element 0 ต่อมาเป็น element 1 ถ้าเราลบ order จะปรับโดยอัตโนมัติเราสามารถที่จะเปลี่ยน order โดย pop-up ที่ขอบของ Cluster แล้วเลือก Cluster order

การ Bundling Data โดยเลือก Bundle Function (ใน Array & Cluster menu) ปรับขนาดของ bundle ให้เหมาะสมกับ input แล้ว wire สายเข้ากับ bundle order ของ Cluster จะเป็นไปตาม input จากบนลงล่าง

การทำ Unbundling จะทำการแยก Cluster ออกมาเป็น component แต่ละอัน order ของ element เหมือนกับการทำ bundle

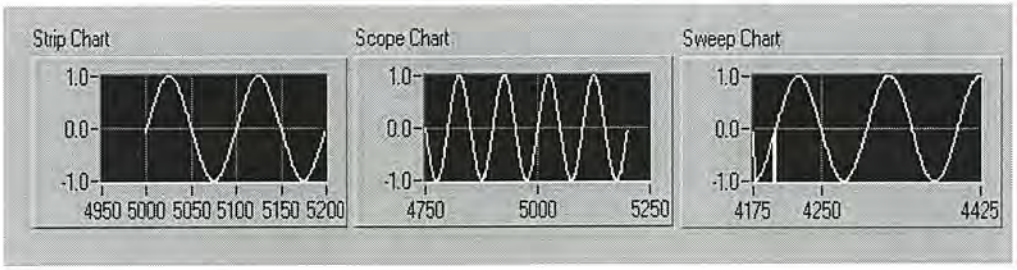
4) Charts and Graphs

Waveform Charts

Waveform Charts อยู่ใน Array & Graph ใน control menu waveform Chart เป็น numeric indicator ชนิดพิเศษที่สามารถแสดงได้ 1 หรือ มากกว่า 1 plot (หรือ แสดงผลข้อมูลได้หลายชุดพร้อมกัน)

4.1) การแสดงผลบน Chart

มีได้ 3 วิธี คือ Strip Chart, scope Chart และ sweep Chart เราสามารถกำหนดรูปแบบของ Chart ได้โดยการทำ pop-up บน waveform Chart เลือก Data operation แล้วเลือก update mode ที่ต้องการ



รูปที่ 2.36 การแสดงผลบน Chart

Strip Chart เมื่อถึงขอบขวาสุดจะเลื่อนให้เอง

Scope Chart เมื่อถึงขอบขวาสุดจะ clear Chart เดิมก่อน และเริ่ม plot จากด้านซ้ายมือ และ plot ต่อจากเดิม

Sweep Chart จะไม่ clear Chart เดิม แต่จะเริ่ม plot จากซ้ายมือ และ plot ต่อจากเดิม

Single plot Chart เราสามารถเอาผลลัพธ์จาก scalar ไปออก Chart ได้โดยตรง ส่วนถ้ามี scalar หลานชุด (หลาย plot) เราสามารถ Bundle Data เหล่านั้นแล้วแสดงออก Chart ได้

Waveform Chart สามารถ

- แสดงหรือซ่อนตัวเลขดิจิตอล
- scrollbar Chart สามารถดู Data เก่าได้โดยจะมี scrollbar เราสามารถเรียกโดยเรียก pop-up บน Chart
- สามารถลบค่าที่อยู่บน Chart ได้ โดยเลือก Clear Chart จาก Data Operation บน pop-up menu
- Chart History Length โดยปกติ Chart สามารถเก็บรายละเอียดได้ 1024 จุด แต่เราสามารถตั้งค่าได้โดยเลือก Chart History Length จาก pop-up menu

4.2) Chart และ Graph Component

Chart และ Graph จะปรับ scale ทั้งแกนตั้งและแกนนอนให้อัตโนมัติ แต่เราสามารถตั้งให้ปรับแกน X หรือ Y อัตโนมัติได้ โดย pop-up ที่ Chart เลือก Auto ScaleXจาก Data Operation นอกจากนี้ทั้ง Xscale และ Yscale สามารถเลือกความละเอียดของจุดทศนิยม (เลือกจาก Format & Precision ใน X หรือ Y scale) ตั้งการ Mapping Mode (ว่าเป็น log หรือ Linear)

- Legend

ถ้าเรามีหลาย Plot เราสามารถเซ็ตค่าแต่ละ plot ให้มีรูปร่างลักษณะอย่างไร สีอะไร เราสามารถดูรายละเอียดและตั้งค่าการแสดงผลแต่ละ plot ได้โดยเรียก legend จาก Show ใน pop-up menu ของ Graph และ สามารถกำหนดแต่ละ plot ได้

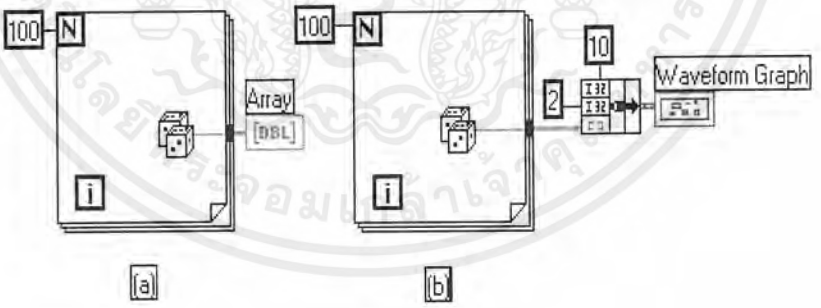
รูปที่ 2.37 legend ของ Chart และ Graph

4.3) Graph

Graph ต่างจาก Chart คือ input ที่เข้าต้องเป็น Array LabVIEW จะมี Graph 2 ชนิด คือ wave Graph และ XYGraph ซึ่งทั้ง 2 ชนิดนี้การแสดงผลออกมาเหมือนกัน แต่มีหน้าที่ต่างกัน

เราสามารถเรียกใช้ Graph indicator จาก Array & Graph ใน control menu waveform Graph จะ plot เฉพาะสมการที่มีตัวแปรเดียว ส่วน XY Graph สามารถ plot สมการที่มีตัวแปรหลายตัวได้ เช่น สมการวงกลม

- Single Plot Waveform Graph



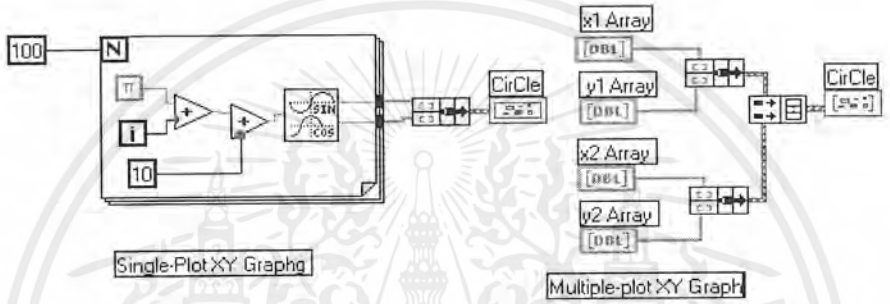
รูปที่ 2.38 waveform Graphs

สำหรับ Single Plot Graph Array Y สามารถส่งไปให้ waveform Graph ได้เลย วิธีนี้จะกำหนดให้ค่าเริ่มต้นของแกน X และค่า delta X เป็น 1 และ 1 ตามลำดับ ดังรูปที่ 38(a) แต่ถ้าต้องการกำหนดค่าเริ่มต้นเองสามารถทำได้ดังรูป 38 (b)

สำหรับ Multiple plot Waveform Graphs เมื่อมีหลาย Array เข้ามาต้องรวมเป็น Array เดียว แต่เพิ่มมิติ โดยใช้ Build Array ก่อนแสดงผลบน waveform Chart

- XY Graph

เหมาะสำหรับการ plot Function ทางคณิตศาสตร์ที่ยาก โดยเราต้องระบุจุดโดยใช้ (x,y) สำหรับ single plot XY Graph Data ก่อนเข้า XY Graph ต้อง Bundle ระหว่าง X Array และ Y Array ถ้าเป็น multiple plot ต้อง build Array ก่อนเข้า ดังรูปที่ 2.39



รูปที่ 2.39 XY Graph แบบ Single plot และ Multiple Plot

2.2 IEEE-488 (GPIB)

2.2.1 โครงสร้างของ IEEE-488

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

1) **Talker**

ทำหน้าที่ส่งข้อมูล โดยในระบบสามารถมี Talker ได้หลายตัว แต่จะมีเพียงตัวเดียว เท่านั้นที่กำลังทำงานอยู่

2) **Listener**

ทำหน้าที่เป็นตัวรับข้อมูล โดยในระบบเดียวกันสามารถมี Listener ได้หลายตัวเช่นเดียวกัน แต่ Listener สามารถทำงานได้ครั้งละหลายๆ ตัวได้

3) **Controller**

ทำหน้าที่ควบคุมอุปกรณ์ต่างๆ ในระบบ โดยจะกำหนดให้ Talker ทำการส่งข้อมูลหรือกำหนดให้ Listener ทำการรับข้อมูล

อุปกรณ์ที่มี GPIB นั้นสามารถแบ่งตามหน้าที่ได้ดังนี้

1. ทำหน้าที่เป็น Talker เท่านั้น เช่น เครื่องมือวัด เป็นต้น
2. ทำหน้าที่เป็น Listener เท่านั้น เครื่องพิมพ์ (Printer) เครื่องบันทึก (Recorder) เป็นต้น
3. ทำหน้าที่เป็นทั้ง Talker และ Listener เช่นคอมพิวเตอร์ เครื่องมือวัดที่สามารถควบคุมได้จากภายนอก เป็นต้น
4. ทำหน้าที่เป็น Talker, Listener และ Controller ในตัวเดียวกัน เช่นคอมพิวเตอร์ที่ทำหน้าที่ควบคุมระบบ

2.2.2 ขีดจำกัดของ IEEE-488

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

2. สายเคเบิลที่ต่อระหว่างอุปกรณ์ จะต้องยาวไม่เกิน 4 เมตร และความยาวรวมของสายเคเบิลในระบบจะต้องไม่เกิน 20 เมตร

3. ความเร็วในการส่งข้อมูลจะต้องไม่เกิน 1Mb/ Sec (1 ล้านไบต์ต่อวินาที) ต้องมีการจ่ายไฟให้กับอุปกรณ์มากกว่าครึ่งหนึ่งของระบบ

2.2.3 รายละเอียดเกี่ยวกับ IEEE-488

ลักษณะทางกายภาพ IEEE-488 นั้นคือ เป็นสายสัญญาณแบบ 24 เส้นขนานกัน และมีขั้วต่ออยู่ทางปลายทั้งสองของสาย เพื่อต่อกับอุปกรณ์หรือต่อกันเพื่อให้สายสัญญาณมีความยาวเพิ่มขึ้น ในจำนวนสายสัญญาณ 24 เส้น มีเพียง 16 เส้นเท่านั้น ที่ทำหน้าที่นำสัญญาณ ส่วนที่เหลืออีก 8 เส้น ทำหน้าที่กราว (ground) และ ชีลด์ (shield) ดังรูปที่ 2.40

1) จำนวนสายที่ใช้นำสัญญาณ 16 เส้น ได้แบ่งได้เป็น 3 ประเภท ตามรูปที่ 2.40 คือ

1.1) บัสข้อมูล (Data Bus) จำนวน 8 สาย

DI01-DI08

1.2) สายสัญญาณควบคุม (Control Line) จำนวน 5 สายคือ

IFC (Interface Clear)

ATN (Attention)

SRQ (Service Enable)

REN (Remote Enable)

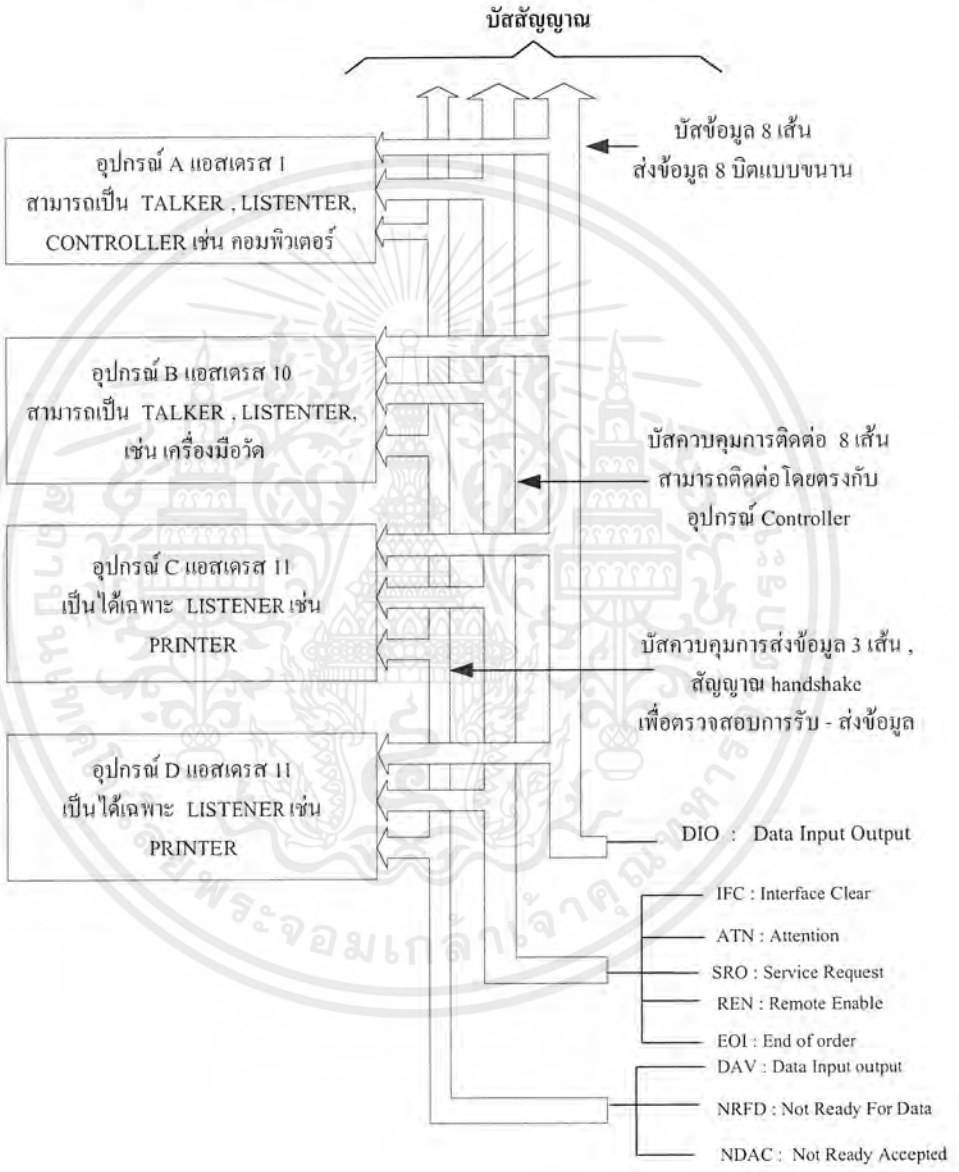
EOI (End or Identify)

1.3) สายแฮนด์เชค (Hand shake) 3 สายคือ

DAV (Data Valid)

NRFD (Not Ready For Data)

NDAC (Not Data Accepted)



รูปที่ 2.40 แสดงการแบ่งเส้นสายนำสัญญาณ

2.2.4 ความหมายของสัญญาณต่างๆ ภายใน IEEE-488

ดังที่ได้กล่าวมาแล้วว่าสายสัญญาณต่างๆใน GPIB ได้แบ่งเป็น 3 กลุ่ม และอธิบายความหมายของสัญญาณต่างๆ ดังนี้

1) กลุ่มสัญญาณข้อมูล

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

2) กลุ่มสัญญาณควบคุมการเชื่อมต่อ (Interface)

2.1) IFC (Interface Clear)

เป็นสัญญาณรีเซต หรือ เคลียร์ระบบ กำหนดได้โดยตัวควบคุม (Controller) เท่านั้น เมื่ออุปกรณ์ในบัสได้รับสัญญาณเคลียร์นี้กลับคืนสู่สภาวะเริ่มต้นใหม่ ซึ่งเป็นสภาวะแรกเริ่มก่อนการกำหนดฟังก์ชันเหมือนแรกเปิดสวิตช์

2.2) ATN (Attention)

เป็นสัญญาณที่ถูกส่งโดยอุปกรณ์ที่เป็นตัวควบคุมเช่นเดียวกัน ใช้ในการสั่งให้อุปกรณ์ทุกตัวในระบบเตรียมพร้อมเพื่อรอรับคำสั่งต่อไป

2.3) SRQ (Service Request)

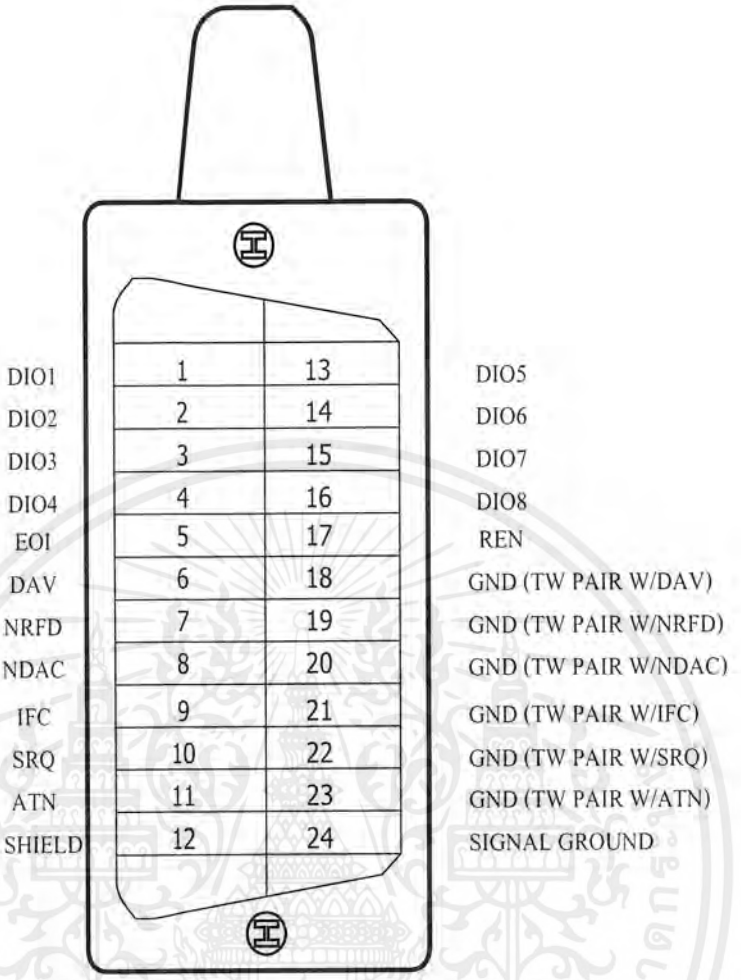
เป็นสัญญาณที่ถูกส่งจากอุปกรณ์ต่างๆ เพื่อเป็นการบอกแก่ระบบว่าขณะนี้อุปกรณ์ดังกล่าวต้องการติดต่อจากตัวควบคุม

2.4) REN (Remote Enable)

เป็นสัญญาณที่ถูกส่งมาจากตัวควบคุมเพียงตัวเดียวเท่านั้น เพื่อใช้สั่งให้อุปกรณ์ต่างๆ เปลี่ยนจากโหมดที่ใช้งานปกติ มาเป็นการควบคุมโดยตัวควบคุมแทน

2.5) EOI (End or Identify)

เป็นสัญญาณที่ถูกส่งได้โดยอุปกรณ์ที่เป็นตัวควบคุม (Controller) หรืออุปกรณ์ที่เป็นตัวส่ง (Talker) ก็ได้ ใช้สำหรับแสดงข่าวสารที่ส่งเป็นชุดนั้นได้เสร็จสิ้นลง



รูปที่ 2.41 ขั้วต่อของ GPIB และการจัดขาของสัญญาณต่างๆ

3) กลุ่มสัญญาณควบคุมการรับ-การส่งข้อมูล

3.1) DAV (Data Valid)

เมื่อสัญญาณนี้ถูกดึงเป็นลอจิก "LOW" โดยอุปกรณ์ที่เป็นตัวส่ง (Talker) เป็นการแจ้งแก่ระบบบัสว่า ขณะนี้ตัวส่งได้ทำการส่งข้อมูลลงไปที่สายสัญญาณข้อมูลเรียบร้อยแล้ว

3.2) NRFD (Not Ready For Data)

เมื่อสัญญาณนี้มีลอจิกเป็น "LOW" จะเป็นการแสดงว่าในขณะนี้ระบบบัสยังไม่พร้อมที่จะรับข้อมูล เนื่องจากอุปกรณ์ในระบบยังพร้อมไม่หมดทุกตัว ซึ่งสัญญาณเส้นนี้จะไม่เป็น "Hi" จนกว่าอุปกรณ์ทุกตัวให้ลอจิกเป็น "Hi" ครบถ้วน สัญญาณนี้มีประโยชน์ในกรณีที่อุปกรณ์ในระบบมีความแตกต่างกัน

3.3) NDAC (Not Data Accepted)

เป็นสัญญาณที่ถูกควบคุมโดยอุปกรณ์ที่เป็นตัวรับ (Listener) โดยสัญญาณนี้จะมีลอจิกเป็น "LOW" ในขณะที่อุปกรณ์ที่เป็นตัวรับกำลังเก็บข้อมูล จากสายข้อมูล (Data Bus) และจะเป็น "Hi" เมื่ออุปกรณ์นั้นได้ทำการอ่านข้อมูลเสร็จเรียบร้อยแล้ว

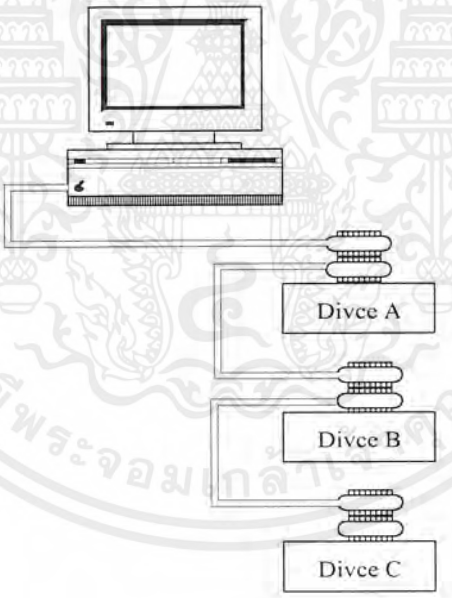
โดยสัญญาณลอจิกที่ใช้ใน DATA BUS (D1-D8) ของ IEEE-488 มีลักษณะเป็นคอมพลีเมนต์ทั้งหมด คือ "1" เท่ากับ "LOW" และ "0" เท่ากับ "Hi" ซึ่งตรงข้ามกับวงจรที่เราคุ้นเคย

2.2.5 การเชื่อมต่ออุปกรณ์ต่างๆในระบบ IEEE-488 BUS

สำหรับการเชื่อมต่อระหว่างอุปกรณ์ต่างๆ ในระบบ IEEE-488 Bus นั้นมีอยู่ 2 วิธี คือ

1) **การเชื่อมต่อแบบเรียงต่อเนื่องกัน** (Daisy Chain Configuration)

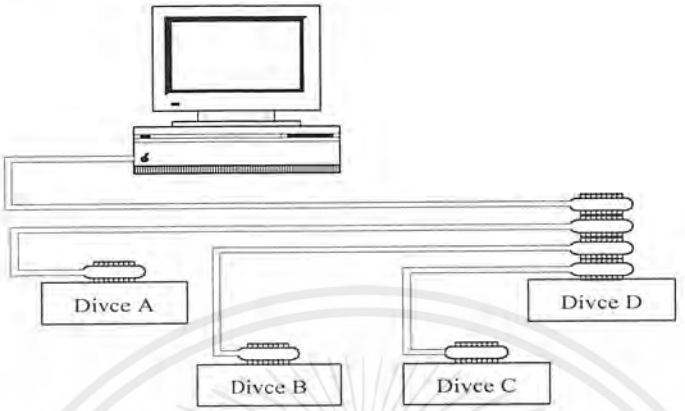
การเชื่อมต่อแบบเรียงต่อเนื่องกัน แสดงดังรูปที่ 2.42



รูปที่ 2.42 การเชื่อมต่อแบบเรียงต่อเนื่องกัน (Daisy Chain Configuration)

2) **การเชื่อมต่อแบบกระจาย** (Star Configuration)

การเชื่อมต่อแบบกระจาย แสดงดังรูปที่ 2.43



รูปที่ 2.43 การเชื่อมต่อแบบกระจาย (Star Configuration)

2.2.6 คำสั่งใช้งานของ GPIB

การสั่งการต่างๆเพื่อกำหนดหน้าที่การทำงานและกำหนดฟังก์ชัน เช่น กำหนดช่วงการวัด โหมดการวัด หรืออื่นๆ แก่เครื่องวัดที่ตกอยู่เหล่านั้น ตัวควบคุมจะเป็นตัวกำหนดการส่งรหัสคำสั่งไปที่อุปกรณ์โดยผ่าน DI1-DI6 รหัสคำสั่งนี้จะถูกส่งไปในช่วงที่สายสัญญาณ ATN เป็น LOW

คำสั่งสำหรับการกำหนดหน้าที่การทำงานต่างๆ ตามมาตราฐานของ GPIB มีอยู่ 128 คำสั่ง โดยแบ่งเป็น 5 กลุ่มคำสั่ง

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

1) **กลุ่มคำสั่งเจาะจงจุดหมาย** (addressed command group)

เป็นคำสั่งที่ส่งไปยังอุปกรณ์ที่เป็นตัวส่งหรือตัวรับที่กำหนดไว้ล่วงหน้าแล้ว คำสั่งนี้ประกอบด้วย

GTL (got to local) สั่งให้อุปกรณ์กลับสู่สภาพการควบคุมปกติด้วยมือ

SDC (selected device clear) สั่งให้อุปกรณ์เคลียร์ตัวเองสู่สภาพเริ่มต้นใหม่

PPC (paralleled poll configure) เป็นคำสั่งสำหรับทำการจัดสายสัญญาณของการจัดสรรสายสัญญาณของการทำกระบวนการตรวจสภาพอุปกรณ์โดยวิธีขนาน โดยใช้กับกลุ่มคำสั่งรอง

GET (group execute trigger) ใช้สั่งเริ่มต้นการทำงานของอุปกรณ์ทีละหลายตัว

TCT (take control) เป็นการกำหนดให้อุปกรณ์ตัวส่งมาหน้าที่เป็นตัวควบคุม

2) กลุ่มคำสั่งครอบคลุม (universal command group)

เป็นคำสั่งที่ส่งไปยังอุปกรณ์ทุกตัวที่ต่ออยู่ในบัส ประกอบด้วย

LLO (local lockout) เป็นการสั่งให้อุปกรณ์ล็อคอยู่ที่สภาวะควบคุมโดยปุ่มปรับที่หน้าปัดตามปกติ

DCL (device clear) สั่งให้อุปกรณ์ทุกตัวกลับไปสู่สภาวะเริ่มต้น

PPU (parallel poll unconfigure) ใช้ยกเลิกกระบวนการตรวจสอบสภาพแบบขนาดทั้งหมด

SPE (serial poll enable) เปลี่ยนโหมดการตรวจสอบสภาพเป็นแบบอนุกรมในโหมดนี้ จะเป็นการส่งสถานะของเครื่องแทนการส่งข้อมูล

SPD (serial poll disable) ยกเลิกการตรวจสอบแบบอนุกรม

3) กลุ่มคำสั่งกำหนดอุปกรณ์ตัวรับ (listener address group)

เป็นคำสั่งสำหรับการกำหนดให้อุปกรณ์เป็นตัวรับ ตามรหัสหมายเลขจาก 0-30 และมีคำสั่ง UNT (untalker) สำหรับยกเลิก

4) กลุ่มคำสั่งกำหนดอุปกรณ์ตัวส่ง (talker address group)

สำหรับกำหนดให้อุปกรณ์เป็นตัวส่ง ตามรหัสหมายเลขจาก 0-30 และมีคำสั่ง UNL (unlisted) สำหรับยกเลิกเช่นกัน คำสั่งกลุ่มที่ 1 –4 นั้น จัดเป็นกลุ่มคำสั่งที่มีความหมายตายตัวยังมีคำสั่งอีกกลุ่มที่ขึ้นอยู่กับการกำหนดภายหลังนั่นคือ กลุ่มคำสำรอง

5) กลุ่มคำสำรอง (secondary command group)

เป็นคำสั่งที่กำหนดรายละเอียดย่อยของอุปกรณ์แต่ละตัวที่ต่ออยู่ในระบบ ให้มีการทำงานอย่างไร ตามจุดประสงค์ใช้งานตามเครื่องมือนั้นเช่นเดียวกับการปรับปุ่มต่างๆ ด้วยมือนั่นเอง คำสั่งรองนี้จะตามหลังคำสั่งหลัก คือจะใช้หลังจากอุปกรณ์ต่างๆถูกกำหนดวางตัวในระบบเรียบร้อยแล้ว

คำสั่งต่างๆที่กล่าวไป ซึ่งใช้ในการกำหนดสภาวะการทำงานของอุปกรณ์ แต่ละสภาวะที่กำหนดไปนั้นเป็นอย่างไร และมีจุดประสงค์เพื่ออะไร ดังต่อไปนี้

Device clear / Interface Clear

Device clear ใช้ในการทำให้อุปกรณ์ที่ต่ออยู่ในบัสกลับไปสู่สภาวะเริ่มต้น ยังไม่มีการกำหนดฟังก์ชันใดๆ สภาวะเริ่มต้นนี้จะแตกต่างกันไป แล้วแต่อุปกรณ์นั้นออกแบบไว้อย่างไร Device clear มีอยู่ 2 ลักษณะคือเคลียร์หมดทุกตัวที่มีอยู่ (DCL) กับเคลียร์เจาะจงอุปกรณ์ตัวใดตัวหนึ่ง (SDC)

แต่ว่าในการเคลียร์อุปกรณ์ให้อยู่ในสภาวะเริ่มต้นนั้นไม่ได้หมายความว่า Interface Function ของ GPIB จะถูกเคลียร์อุปกรณ์ให้ไปอยู่ในสภาวะเริ่มต้นด้วยแต่อย่างใด Interface Function คือสภาพการ Interface ที่ได้กำหนดไว้ในระบบประกอบด้วยฟังก์ชันต่างๆ ดังแสดงในตารางที่ 2.2

ตารางที่ 2.2 แสดง Interface Function

ฟังก์ชั่น	สัญลักษณ์	การกลับสู่จุดเริ่มต้นโดย IFC
Source handshake	SH	ได้
Acceptor handshake	AH	ได้
Talker or enlarge talker	Tor LE	ได้
Listener or enlarge listener	Lor LT	ได้
Service request	S R	ไม่ได้
Remote/local	R L	ไม่ได้
Parallel poll	P P	ไม่ได้
Device clear	D C	ได้
Device trigger	D T	ได้
Controller	C	ได้

2.2.7 Remote / Local

Remote เป็นการกำหนดให้อุปกรณ์ที่อยู่ในระบบเช่น เครื่องมือวัดให้อยู่ในการควบคุมของอุปกรณ์ตัวอื่นแทน ซึ่งปุ่มปรับต่างๆ บนหน้าปัดเครื่องจะไม่มีผลการทำงาน ส่วน Local เป็นการควบคุมการทำงานของเครื่องมือวัดด้วยปุ่มปรับบนหน้าปัดตามปกติ

การใช้ Remote มีประโยชน์ในแง่ที่ขณะที่ตัวการควบคุมเช่น คอมพิวเตอร์กำลังติดต่ออุปกรณ์ตัวนั้นอยู่ หากไม่มีการตัดการควบคุมโดยปุ่มปรับหน้าปัดออก ถ้ามีใครมาปรับแต่งก็จะทำให้การทำงานผิดพลาดไปได้ การทำงานของ GPIB ใน remote and local มี 4 ลักษณะดังนี้

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

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

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

4. LWLS มีสภาพเช่นเดียวกับ local แต่จะแตกต่างกันตรงที่สภาพ local โดย LWLS นี้ เมื่อได้รับคำสั่งกำหนดอุปกรณ์ตัวรับจะเปลี่ยนไปอยู่ในสภาพแบบล็อกหรือ RWLS ทันทีในการที่จะมาที่สภาพ LWLS นั้นได้มี 2 กรณีคือ เมื่ออยู่ในสภาพ local ธรรมดา (LOCS) แล้วได้รับคำสั่ง LLO หรืออยู่ใน RWLS แล้วได้รับคำสั่ง GTL

2.2.8 การขอบริการและการตรวจสอบ (Service Request and Polling)

เมื่อตัวควบคุมได้รับ SRQ เป็น LOW จะให้อุปกรณ์ส่งข้อมูลแสดงสถานะการทำงาน ซึ่งมี 2 วิธี คือ

1) การตรวจสอบแบบอนุกรม ซึ่งมีขั้นตอนดังนี้

1.1 ATN ถูกดึงเป็น LOW หลังจากได้รับ LOW จากสายสัญญาณ SRQ

1.2 คำสั่ง UNL ถูกสั่งไปยังอุปกรณ์

1.3 ตัวควบคุมจะแจ้งรหัสตัวรับของตน และกำหนดรหัสตัวส่งอุปกรณ์ที่จะตรวจสอบไปที่บัส

1.4 ตามด้วยคำสั่ง SPE และสาย ATN กลายเป็น High ซึ่งอุปกรณ์ที่ถูกเรียกจะส่งข้ามแสดงสถานะออกมา 1 ไบต์ โดยบิตที่ 7 จะเป็นตัวชี้ว่าอุปกรณ์เส้นเป็นตัวขอบริการ ถ้าใช่จะเป็น LOW ส่วนบิตอื่นๆ ก็ใช้บอกข้อมูลอื่นๆซึ่งไม่ได้กำหนดเฉพาะ สาย ATN ถูกดึงเป็น LOW อีกครั้งเพื่อส่งคำสั่งยกเลิกการตรวจสอบคือ SPD

1.5 จากนั้นคำสั่ง UNT ก็ส่งไปยังอุปกรณ์เพื่อยกเลิกการเป็นตัวส่ง ซึ่งถ้าหาก SQR ยังคงเป็น LOW อยู่ ก็จะมีการตรวจสอบไปยังอุปกรณ์ตัวอื่นๆ ต่อไปตามขั้นตอนเดิม

2) การตรวจสอบแบบขนาน สามารถทำได้เร็วกว่าแบบอนุกรมทั้งนี้เพราะสามารถอ่านข้อมูลเพียงไบต์เดียวก็สามารถรู้ได้ทันทีว่าอุปกรณ์ตัวใดเป็นตัวขอรับบริการ

2.3 VISA (Virtual Instrument Architecture)

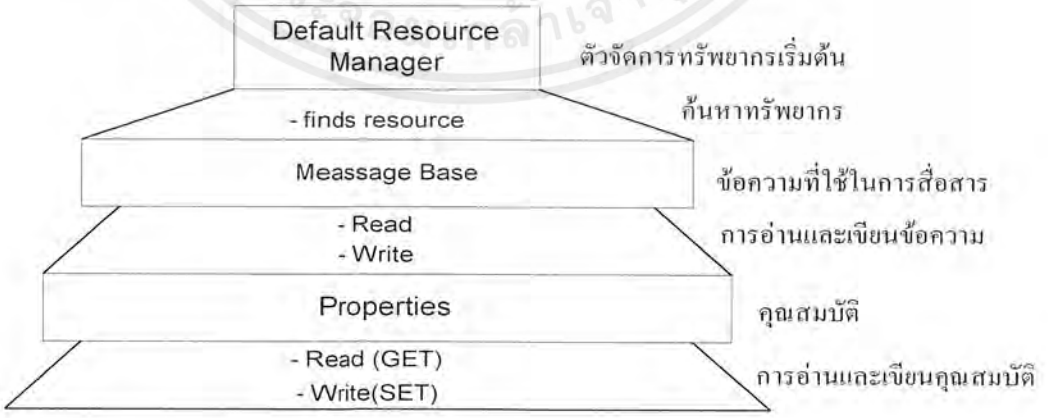
VISA คือมาตราฐานของ I/O Application Programming Interface (API) สำหรับการเขียนโปรแกรมเพื่อควบคุมอุปกรณ์เครื่องมือวัดซึ่งโดยตัว VISA แล้วไม่สามารถที่จะโปรแกรมอุปกรณ์เครื่องมือวัดได้แต่ VISA เป็นภาษา API ระดับสูงที่จะไปสั่งงานไดร์ฟเวอร์ซึ่งอยู่ระดับล่างลงไป โครงสร้างของ VISA สามรถแสดงได้ดังรูปที่ 2.44

รูปที่ 2.44 โครงสร้างของ VISA

VISA สามารถควบคุมพอร์ต VXI, GPIB และพอร์ตอนุกรมได้ โดยการเลือกไดร์ฟเวอร์ที่เหมาะสม ซึ่งขึ้นอยู่กับอุปกรณ์เครื่องมือวัดที่เราใช้ ดังนั้นในขณะที่เราแก้ไขโปรแกรมเราต้องใช้ความระมัดระวังเป็นอย่างสูง เพราะปัญหาอะไรก็ตามที่เกิดขึ้นจาก VISA คือปัญหาที่เกิดจากไดร์ฟเวอร์ระดับล่างที่ถูก VISA เรียกใช้งานอยู่ในขณะนั้น

2.3.1 พื้นฐานของ VISA

โครงสร้างภายในของ VISA สามารถแสดงให้เห็นได้ดังรูปที่ 2.45



รูปที่ 4.45 โครงสร้างภายในของ VISA

2.3.2 Default Resource Manager, Session, and Instrument descriptors

การจัดการทรัพยากรเริ่มต้น (Default Resource Manager) เป็นการปฏิบัติการอันดับแรกของ VISA ซึ่ง LabVIEW จะติดต่อสื่อสารกับอุปกรณ์ภายนอกโดยอัตโนมัติด้วยตัวจัดการทรัพยากรเริ่มต้น เป็นอันดับแรกหลังจากที่ VI มีการเรียกใช้ VISA ในขั้นตอนนี้จะทำให้เกิด 2 สิ่งที่เราจำเป็นต้องนิยามคือ ทรัพยากรและเซกชั่น

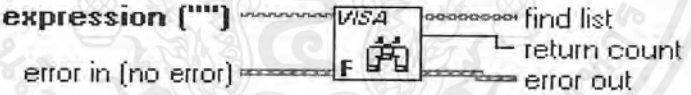
1. ทรัพยากร (Resource) คือสิ่งที่มีอยู่จริง ที่เราสามารถติดต่อสื่อสารด้วย เช่น เครื่องมือวัดต่าง ๆ

2. ค่าเซกชั่น (Session) คือการติดต่อสื่อสาร(link)กับอุปกรณ์รอบนอก

เราสามารถใช้ตัวจัดการทรัพยากรเริ่มต้นของ VISA ในการเปิดเซกชั่นเพื่อติดต่อกับอุปกรณ์ภายนอก และส่วนใหญ่แล้วเราจำเป็นต้องเปิดเซกชั่นของทรัพยากรต่าง ๆ ก่อนที่เราจะมีการติดต่อกับทรัพยากรเหล่านั้น และตัวจัดการทรัพยากรพื้นฐานยังสามารถใช้ค้นหาทัพยากรที่มีอยู่นะระบบด้วย

1) ฟังก์ชันของ VISA ที่ใช้ค้นหาทรัพยากรในระบบ(VISA Find Resource)

ฟังก์ชันของ VISA ที่ใช้ค้นหาทรัพยากรในระบบ แสดงให้เห็นได้ดังรูปที่2.50 ฟังก์ชันนี้ถือเป็นจุดเริ่มต้นของการโปรแกรม VISA เราสามรถใช้ฟังก์ชันนี้ในการกำหนดทรัพยากรที่จำเป็นสำหรับฟังก์ชันของเราได้

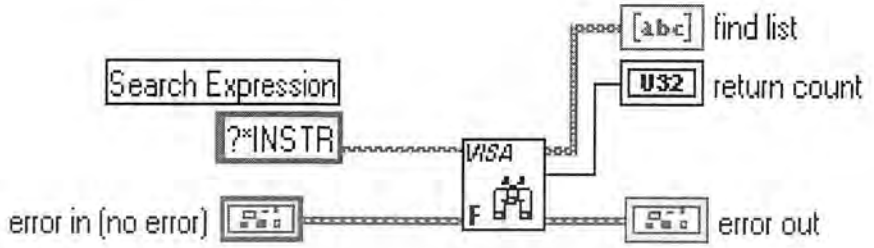


รูปที่ 2.46 ลักษณะของ VISA Find Resource

จากรูปที่ 2.46 เราจะเห็นได้ว่าอินพุตที่จำเป็นของฟังก์ชันคือ "อินพุตสตริง" ที่ถูกเรียกว่า Expression เป็นตัวกำหนดว่าทรัพยากรชนิดใดที่จะให้ฟังก์ชัน VISA Find Resource แสดงค่ากลับเมื่อมีการค้นพบทรัพยากรนั้นในระบบ ค่าที่จะแสดงกลับเมื่อมีการพบอุปกรณ์นั้นมีอยู่ 2 ค่า ดังนี้

1. Return count จะแสดงจำนวนทรัพยากรที่พบในระบบ
2. Find List เอาต์พุตนี้คืออาร์เรย์ของสตริง(Array string) แต่ละสตริงจะบ่งบอกถึงลักษณะ(description)ของทรัพยากรที่เราพบ

ตัวอย่าง ของ VI ที่สามารถค้นหาทรัพยากรของระบบทั้งหมดได้สามารถเขียนได้ตามรูปที่ 2.47



รูปที่ 2.47 ตัวอย่างการใช้ฟังก์ชัน Find Resource

2) ตัวบ่งบอกลักษณะของเครื่องมือวัด (Instrument Descriptor)

เป็นการแสดงถึงรายละเอียดและตำแหน่ง(location) ของทรัพยากรที่เราพบในระบบ โดยจะแสดงในรูปแบบของฟอร์แมตสตริง (Format string) ซึ่งมีรูปแบบดังนี้

Interface Type [board index]::Address::VISA Class

จากฟอร์แมตสตริงด้านบนเราสามารถแบ่งออกเป็น 3 ส่วนได้ ดังนี้

1. Interface Type [board index] คือ ส่วนที่บอกว่าระบบของเราใช้พอร์ตอะไรในการติดต่อสื่อสารและมีพอร์ตดังกล่าวนี้กี่พอร์ต เช่น ถ้าเรามีการ์ด GPIB 2 อันอยู่ในระบบของเรา ส่วนนี้จะแสดงเป็นฟอร์แมตสตริงดังนี้ GPIB0 สำหรับการ์ด GPIB ที่ 1 และ GPIB1 สำหรับการ์ด GPIB ที่ 2

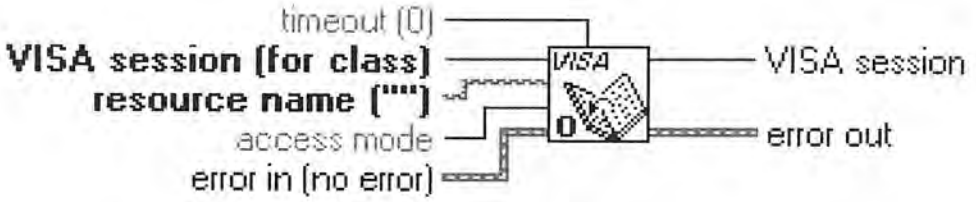
2. Address คือ ส่วนที่บอกว่าเครื่องมือวัดที่เชื่อมต่อระบบของเราอยู่นั้น อยู่ที่ตำแหน่งอะไร เช่น ถ้าในระบบของเรามีการ์ด GPIB หนึ่งการ์ดและเชื่อมต่อกับเครื่องมือวัดที่กำหนดค่าตำแหน่งไว้ที่ 1 ส่วนนี้จะมีฟอร์แมตสตริงดังนี้ GPIB0::1

3. VISA Class เป็นการรวมการปฏิบัติการบางอย่างหรือทั้งหมดของ VISA ซึ่ง คลาส INSTR เป็นคลาสพื้นฐานที่ครอบคลุมการปฏิบัติการพื้นฐานของ VISA สำหรับเครื่องมือวัด ปกติ LabVIEW ใช้คลาส INSTR เป็นคลาสเริ่มต้น(Class default)

3) การเปิดและปิดเซกชั่น(Opening and Closing Session)

3.1) การเปิดเซกชั่น

การเปิดเซกชั่นจะกระทำเมื่อเราต้องการใช้หรือจัดการทรัพยากรในระบบ ฟังก์ชั่นที่ใช้ในการเปิดเซกชั่นคือ VISA Open Function แสดงให้เห็นได้ดังรูปที่ 2.48



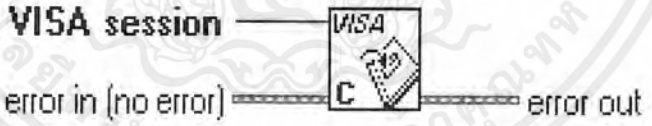
รูปที่ 2.48 VISA Open

อินพุตที่เรียกว่าชื่อของทรัพยากร(Resource Names input) เป็นสตริงที่ได้มาจากตัวบอกลักษณะเครื่องมือวัด

แอปพลิเคชั่นส่วนใหญ่ต้องการ การเปิดเซกชั่นเพียงครั้งเดียวสำหรับเครื่องมือวัดแต่ละเครื่อง ที่แอปพลิเคชั่นต้องการติดต่อด้วย

3.2) การปิดเซกชั่น

เมื่อ VISA ได้มีการเปิดเซกชั่นแล้ว VISA จะเข้าไปใช้ทรัพยากรภายในเครื่องคอมพิวเตอร์ของเรา จนกว่าจะมีการใช้คุณสมบัติการปิดโปแกรม VISA (End a Program VISA properly) ดังนั้นเมื่อมีการเปิดเซกชั่นแล้วจำเป็นต้องมีการปิดเซกชั่นด้วย มิฉะนั้น VISA จะไม่คืนทรัพยากรให้กับระบบของเรา VISA Close Function แสดงให้เห็นได้ดังรูปที่ 2.49



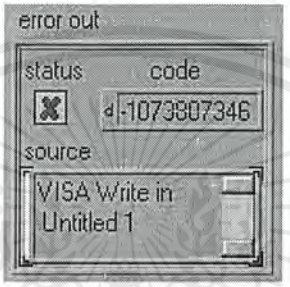
รูปที่ 2.49 VISA Close

2.3.3 การตรวจจับข้อผิดพลาดด้วย VISA (Error Handling with VISA)

การตรวจจับข้อผิดพลาดด้วย VISA VIs เหมือนกับการตรวจจับข้อผิดพลาดอื่น ๆ ใน LabVIEW โดยแต่ละ VISA VIs จะมีจุดต่อข้อผิดพลาดด้านเข้า (Error input Terminal) และจุดต่อข้อผิดพลาดด้านออก (Error output Terminal) ซึ่งถูกใช้ส่งค่าความผิดพลาดผ่านไปยังคลัสเตอร์แสดงความผิดพลาด (Error Cluster) ของ VI หนึ่งไปยังอีก VI หนึ่งในdiagram คลัสเตอร์แสดงค่าความผิดพลาดจะประกอบด้วย 3 ส่วนดังนี้

1. ค่าทางตรรก (Boolean) ใช้แสดงสถานของการเกิดข้อผิดพลาด
2. ค่าตัวเลข (Numeric) ใช้แสดงรหัสข้อผิดพลาดที่เกิดขึ้น (Error Code)
3. ค่าตัวอักษร (String) ใช้แสดงตำแหน่งของ VI ที่เกิดข้อผิดพลาดขึ้น

ถ้าเกิดข้อผิดพลาดขึ้นในอันดับต่อ ๆ ไป VI จะไม่เอ็กซ์คิว (Execute) โปรแกรมต่อ และVI จะผ่านค่าข้อผิดพลาดนั้น ๆ ไปยังคลัสเตอร์แสดงข้อผิดพลาด ซึ่งคลัสเตอร์แสดงข้อผิดพลาดด้าน Front Panel ของ VISA แสดงได้ดังรูปที่ 2.50



รูปที่ 2.50 คลัสเตอร์แสดงข้อผิดพลาดด้าน Front Panel

ใน LabVIEW มี VIs ที่ใช้ตรวจจับข้อผิดพลาดอีกแบบหนึ่งเป็นการตรวจจับข้อผิดพลาดอย่างง่าย ผู้ใช้สามารถเข้าไปเรียกใช้ได้จากเพลตย่อย (subpalette) time & Dialog ซึ่งภายใต้เพลต Function ถ้าเกิดข้อผิดพลาดขึ้น Vis นี้จะแสดงกล่องสนทนาขึ้น (pop up dialog box) ภายในกล่องสนทนาจะแสดงเหตุผลที่เป็นไปได้ในการเกิดข้อผิดพลาดนั้น ๆ การตรวจจับข้อผิดพลาดอย่างง่ายจะแจ้งข้อผิดพลาดต่าง ๆ คล้ายกันกับคลัสเตอร์แสดงความผิดพลาดแต่ Vis ตรวจจับข้อผิดพลาดอย่างง่ายจะแจ้งรายละเอียดของการเกิดข้อผิดพลาดได้มากกว่า กล่องสนทนาแสดงข้อผิดพลาดแสดงให้เห็นได้ดังรูปที่ 2.51

รูปที่ 2.51 กล่องสนทนาแสดงข้อผิดพลาด

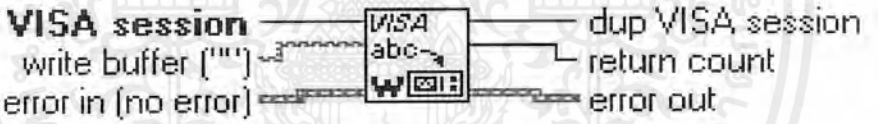
2.3.4 ข้อความในการติดต่อสื่อสาร (Message Based Communication)

ใน LabVIEW การที่จะติดต่อสื่อสารกับพอร์ต GPIB และพอร์ตอนุกรม จะใช้ข้อความในการติดต่อสื่อสาร (Message Based Communication) ที่แตกต่างกัน ซึ่งที่ระดับของ VISA จะเป็นเพียงโพรโตคอลที่ใช้ในการส่งข้อความไปยังเครื่องมือวัด โดยที่ผู้ใช้ไม่สามารถมองเห็นขั้นตอนในการติดต่อได้ ผู้ใช้จำเป็นต้องทราบเพียงว่า เวลาใดควรจะเขียน (ส่ง) ข้อความและเวลาใดควรอ่าน (รับ) ข้อความ LabVIEW จะใช้ฟังก์ชัน VISA Write ในการเขียนข้อความเพื่อสั่งงานเครื่องมือวัด และใช้ฟังก์ชัน VISA Read ในการอ่านข้อความต่าง ๆ จากเครื่องมือวัด

VISA Write เป็นฟังก์ชั่นที่ LabVIEW ใช้ในการส่งข้อความคำสั่งไปยังเครื่องมือวัด VISA Write แสดงให้เห็นได้ดังรูปที่ 2.52

นอกจาก VISA session แล้วมีอินพุตที่จำเป็นเพียงอินพุตเดียวคือ write buffer เป็นอินพุตที่รับข้อความกรือคำสั่งที่ต้องการส่งไปยังเครื่องมือวัดจากผู้ใช้ ลักษณะของ VISA Write แสดงให้เห็นได้ดังรูปที่ 2.52

เอาต์พุต Return count เป็นเอาต์พุตที่แสดงจำนวนไบต์ของข้อความที่เป็นอินพุต



รูปที่ 2.52 ลักษณะของ VISAWrite

VISA Read เป็นฟังก์ชั่นที่ LabVIEW ใช้ในการอ่านข้อความต่าง ๆ จากเครื่องมือวัดอินพุต Bytes count เป็นอินพุตที่ใช้กำหนดค่าจำนวนไบต์สูงสุดที่ VISA Read จะสามารถอ่านได้จากเครื่องมือวัด VISA Read จะหยุดการอ่านค่าเมื่อครบจำนวนไบต์ที่กำหนดหรือเมื่อสิ้นสุดค่าที่ได้จากเครื่องมือวัด VISA Read สามารถแสดงให้เห็นได้ดังรูปที่ 2.53

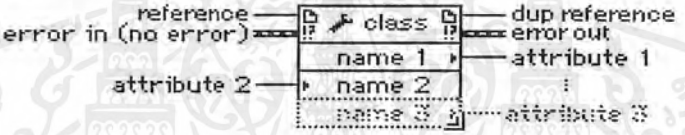
เอาต์พุต Read buffer เป็นเอาต์พุตที่ใช้แสดงข้อความต่าง ๆ ที่ VISA Read อ่านได้จากเครื่องมือวัด

เอาต์พุต Return count เป็นเอาต์พุตที่แสดงจำนวนไบต์ของข้อความที่อ่านได้จากเครื่องมือวัด

รูปที่ 2.53 ลักษณะของ VISA Read

2.3.5 คุณสมบัติของ VISA (VISA Properties)

ในการปฏิบัติการพื้นฐานต่าง ๆ เกี่ยวกับการติดต่อสื่อสารกับเครื่องมือวัด จะถูกครอบคลุมด้วยคำสั่งที่เป็นข้อความดังที่ได้กล่าวมาแล้วในเรื่องของ VISA แต่ผู้ใช้สามารถเพิ่มคุณสมบัติต่าง ๆ ของการปฏิบัติการที่นอกการปฏิบัติการพื้นฐานได้ โดยการอ่านหรือเขียนค่าของคุณสมบัติที่ต้องการได้ในโปรแกรม ซึ่ง LabVIEW จะใช้ Property Node ในการอ่านและกำหนดค่าคุณสมบัติของ VISA Property Node แสดงให้เห็นได้ดังรูปที่ 2.54



รูปที่ 2.54 Property Node

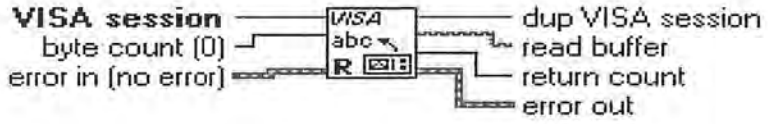
หลังจากที่วาง Property Node ไว้ที่บล็อกไดอะแกรมแล้วผู้ใช้จะสามารถกำหนดคุณสมบัติต่าง ๆ ของ VISA ได้ วิธีการในการกำหนดคุณสมบัติของ VISA มีดังต่อไปนี้

1. ต่อสาย VISA Session เข้าที่จุดต่ออ้างอิงด้านเข้า (Reference input terminal) ของProperty Node

2. คลิกขวาที่ Property Node และเลือก **Instr** ที่ตัวเลือก VISA Class

ขณะที่เราวาง Property Node ที่บล็อกไดอะแกรมเป็นครั้งแรก Property Node จะมีคุณสมบัติเพียงอย่างเดียว แต่ Property Node สามารถเปลี่ยนขนาดให้สามารถมีคุณสมบัติเพิ่มขึ้นได้โดยง่าย ค่าเริ่มต้นของ VSA Property Node จะเป็นจุดต่อที่ใช้สำหรับอ่านค่าคุณสมบัติ ซึ่งจุดต่อสำหรับการอ่านนี้จะแสดงให้เห็นเป็นรูปลูกศรเล็ก ๆ ด้านขวามือของ Property Node

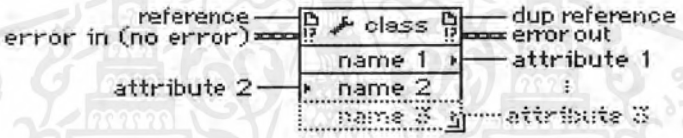
บางจุดต่อ (Terminal) สามารถเป็นได้ทั้งจุดต่อสำหรับการอ่านและจุดต่อสำหรับการเขียนค่าคุณสมบัติ การเปลี่ยนจากจุดต่อสำหรับการอ่านเป็นจุดต่อสำหรับการเขียนนั้นทำได้โดย คลิกขวาที่ Property Node แล้วเลือกเปลี่ยน (Change)



รูปที่ 2.53 ลักษณะของ VISA Read

2.3.5 คุณสมบัติของ VISA (VISA Properties)

ในการปฏิบัติการพื้นฐานต่าง ๆ เกี่ยวกับการติดต่อสื่อสารกับเครื่องมือวัด จะถูกครอบคลุมด้วยคำสั่งที่เป็นข้อความดังที่ได้กล่าวมาแล้วในเรื่องของ VISA แต่ผู้ใช้สามารถเพิ่มคุณสมบัติต่าง ๆ ของการปฏิบัติการที่นอกการปฏิบัติการพื้นฐานได้ โดยการอ่านหรือเขียนค่าของคุณสมบัติที่ต้องการได้ในโปรแกรม ซึ่ง LabVIEW จะใช้ Property Node ในการอ่านและกำหนดค่าคุณสมบัติของ VISA Property Node แสดงให้เห็นได้ดังรูปที่ 2.54



รูปที่ 2.54 Property Node

หลังจากที่วาง Property Node ไว้ที่บล็อกไดอะแกรมแล้วผู้ใช้จะสามารถกำหนดคุณสมบัติต่าง ๆ ของ VISA ได้ วิธีการในการกำหนดคุณสมบัติของ VISA มีดังต่อไปนี้

1. ต่อสาย VISA Session เข้าที่จุดต่ออ้างอิงด้านเข้า (Reference input terminal) ของProperty Node

2. คลิกขวาที่ Property Node และเลือก **Instr** ที่ตัวเลือก VISA Class

ขณะที่เราวาง Property Node ที่บล็อกไดอะแกรมเป็นครั้งแรก Property Node จะมีคุณสมบัติเพียงอย่างเดียว แต่ Property Node สามารถเปลี่ยนขนาดให้สามารถมีคุณสมบัติเพิ่มขึ้นได้โดยง่าย ค่าเริ่มต้นของ VSA Property Node จะเป็นจุดต่อที่ใช้สำหรับอ่านค่าคุณสมบัติ ซึ่งจุดต่อสำหรับการอ่านนี้จะแสดงให้เห็นเป็นรูปลูกศรเล็ก ๆ ด้านขวามือของ Property Node

บางจุดต่อ (Terminal) สามารถเป็นได้ทั้งจุดต่อสำหรับการอ่านและจุดต่อสำหรับการเขียนค่าคุณสมบัติ การเปลี่ยนจากจุดต่อสำหรับการอ่านเป็นจุดต่อสำหรับการเขียนนั้นทำได้โดย คลิกขวาที่ Property Node แล้วเลือกเปลี่ยน (Change)

2.4 ดาต้าซ็อกเก็ต (Datasocket)

ดาต้าซ็อกเก็ต เป็นเครื่องมือที่ใช้เขียนโปรแกรม โดยดาต้าซ็อกเก็ตจะอนุญาติให้อ่านหรือเขียนและแลกเปลี่ยนข้อมูลไปมาระหว่างแอปพลิเคชั่นที่แตกต่างกันได้ ดาต้าซ็อกเก็ตสามารถเข้าถึงข้อมูลที่เครื่องคอมพิวเตอร์ท้องถิ่น (Local Computer) หรือเครื่องคอมพิวเตอร์ที่ต่อยู่กับระบบเครือข่ายที่ใช้โพรโตคอล HTTP และ FTP Server ได้

สมมุติว่าถ้าเราใช้ฟังก์ชั่นทั่วไปของ LabVIEW เช่น File I/O Function, TCP/IP Function ในการเขียนโปรแกรมเพื่อขนถ่ายข้อมูลผ่านเครือข่ายอินเตอร์เน็ต เราจำเป็นต้องเขียนโค้ดโปรแกรมแยกกันแต่ละโพรโตคอล แต่ถ้าเราใช้ดาต้าซ็อกเก็ต ดาต้าซ็อกเก็ตจะจัดเตรียมส่วนในการเชื่อมต่อระหว่างแอปพลิเคชั่น (API) ของแต่ละโพรโตคอลไว้เป็นอันหนึ่งอันเดียวกัน ทำให้เราสามารถเชื่อมต่อระหว่างแหล่งกำเนิดข้อมูล (Source) และปลายทางของข้อมูล (Target) ที่ใช้โพรโตคอลที่แตกต่างกันได้ โดยไม่จำเป็นต้องเขียนโค้ดโปรแกรมเพื่อรองรับข้อมูลของแต่ละโพรโตคอล ในการระบุตำแหน่ง Datasocket ของเครื่องคอมพิวเตอร์ที่เป็นต้นทางและเครื่องคอมพิวเตอร์ที่เป็นปลายทาง ผู้ใช้สามารถกำหนดอยู่ในรูปแบบ URL (Uniform resource locators) ได้

2.4.1 การใช้ฟังก์ชั่นดาต้าซ็อกเก็ต (Using Datasocket Functionality)

ที่ Front panel ผู้ใช้สามารถนำ Datasocket Control มาได้โดยเลือกที่ Control >> Path & Reference Type.ctl

ที่บล็อกไดอะแกรม ผู้ใช้สามารถนำ Datasocket palette โดยการเลือกที่ Function >> Communication >> Datasocket ซึ่งในแพลตหลัก (Main palette) ของ Datasocket นี้จะเป็นการปฏิบัติการระดับสูง (High – level) เช่น การเปิด (Open), การปิด (Close), การเขียน (Writing) และการอ่าน (Reading) แพลต Datasocket นี้ยังมีแพลตย่อย (Subpalette) อีก 2 พาเลตดังนี้

- Function >> Communication >>Datasocket >> Advanced
- Function >> Communication >>Datasocket >> Read
- Function >> Communication >>Datasocket >> Write

ในปริญญานิพนธ์นี้จะกล่าวถึงฟังก์ชัน Datasocket ที่ได้ใช้บ่อย ๆ ในการสร้างปริญญานิพนธ์นี้เท่านั้น ฟังก์ชันที่ใช้บ่อย ๆ ในปริญญานิพนธ์นี้มีดังนี้คือ

1) Datasocket Open Connection

Datasocket Open Connection เป็นการสร้างดาต้าซ็อกเก็ตอ้างอิง (Datasocket Reference) และเริ่มต้นการเชื่อมต่อไปที่ URL ที่ผู้ใช้กำหนดขึ้น Datasocket Open Connection แสดงให้เห็นได้ดังรูปที่ 2.55

รูปที่ 2.55 ลักษณะของ Datasocket Open Connection

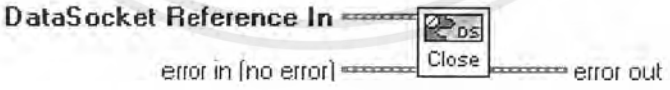
ขณะที่มีการเชื่อมต่อดาต้าซ็อกเก็ตไปที่ต้นทาง (Source) เพื่อการอ่าน (รับ) ข้อมูลจะมีการเปลี่ยนแปลงหนึ่งครั้งเมื่อการเชื่อมต่อเสร็จสมบูรณ์ การอ่านจะรอจนกว่าเวลาที่กำหนดไว้หมดลง (Timeout) เป็นหนึ่งรอบสำหรับการเปลี่ยนแปลง

ขณะที่มีการเชื่อมต่อดาต้าซ็อกเก็ตไปที่ปลายทาง (Target) เพื่อการเขียน (ส่ง) ข้อมูลที่ต้องเขียนจะถูกบันทึกไว้ในดาต้าซ็อกเก็ตและจะมีการเขียนไปจนกว่าการเชื่อมต่อจะถูกสร้างขึ้นมา

อินพุต AccessMode เป็นอินพุตสำหรับบอก Datasocket Open Connection ว่าขณะที่มีการเชื่อมต่อไปที่ต้นทางเพื่อการอ่านหรือเชื่อมต่อไปที่ปลายทางเพื่อการเขียน

2) Datasocket Close Connection

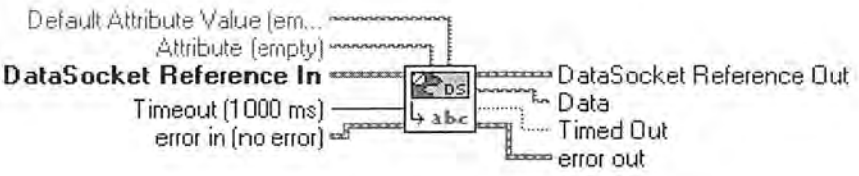
Datasocket Close Connection เป็นการยกเลิกการเชื่อมต่อดาต้าซ็อกเก็ตและทำลายดาต้าซ็อกเก็ตอ้างอิง Datasocket Close Connection แสดงให้เห็นได้ดังรูปที่ 2.56



รูปที่ 2.56 ลักษณะของ Datasocket Close Connection

3) Datasocket Read String

Datasocket Read String เป็นการอ่านค่าหรือคุณสมบัติจากดาต้าซ็อกเก็ตต้นทาง ข้อมูลที่ได้จะเป็นข้อมูลชนิดข้อความ (String) Datasocket Read String แสดงให้เห็นได้ดังรูปที่ 2.57



รูปที่ 2.57 ลักษณะของ Datasocket Read String

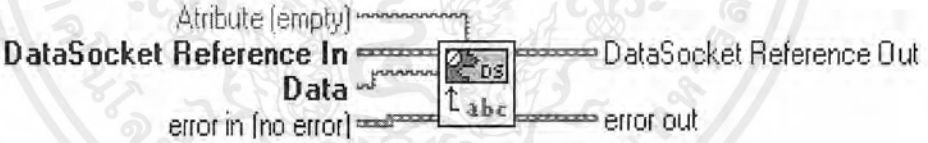
อินพุต Attribute (empty) ถ้ามีการต่ออินพุตนี้และให้คุณสมบัติของอินพุตนี้ว่าง (Empty) ค่าข้อมูลจะอ่าน

อินพุตอื่น ๆ จะเป็นการกำหนดคุณสมบัติต่าง ๆ ของค่าข้อมูลที่จะอ่าน ถ้าไม่มีการกำหนดคุณสมบัติ ค่าคุณสมบัติที่ Default Attribute Value จะถูกส่งกลับ

เอาต์พุต Data จะเป็นข้อมูลชนิดข้อความที่ Datasocket Read String สามารถอ่านได้

4) Datasocket Write String

Datasocket Write String เป็นการเขียนข้อมูลชนิดข้อความไปที่ไปที่ดาต้าซ็อกเก็ตปลายทาง Datasocket Write String แสงดงให้เห็นได้ดังรูปที่ 2.58



รูปที่ 2.58 ลักษณะของ Datasocket Write String

อินพุต Attribute (empty) ถ้ามีการต่ออินพุตนี้และให้คุณสมบัติของอินพุตนี้ว่าง (Empty) ค่าข้อมูลจะเขียน

อินพุตอื่น ๆ จะเป็นการกำหนดคุณสมบัติต่าง ๆ ของค่าข้อมูลที่จะเขียน

5) Datasocket Write String

Datasocket Server Control ใช้เปิดโปรแกรม Datasocket Server เป็น VI ที่จำเป็นต้องมีทุก ๆ ครั้งก่อนการใช้ฟังก์ชั่นดาต้าซ็อกเก็ต ซึ่ง Datasocket Server Control our มีการปฏิบัติการอยู่ 4 อย่างดังนี้

1. Launch เริ่มใช้โปรแกรม Datasocket Server
2. Show แสดงหน้าต่างของโปรแกรม Datasocket Server
3. Hide ซ่อนหน้าต่างโปรแกรม Datasocket Server
4. Close ยกเลิกการใช้งานโปรแกรม Datasocket Server

Datasocket Server Control สามารถแสดงให้เห็นได้ดังรูปที่ 2.59

รูปที่ 2.59 ลักษณะของ Datasocket Server Control

2.4.2 ตัวอย่าง การสร้าง DSReader โดยใช้ ActiveX Controls ใน VIsual Basic

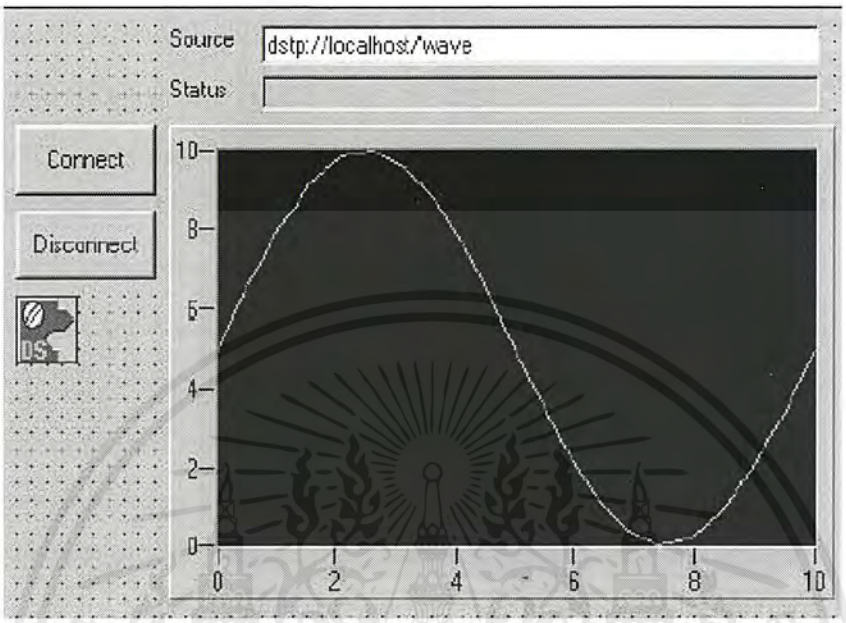
การสร้าง DSReader โดยใช้ ActiveX Controls ใน VIsual Basic เพื่อให้ DSReader ทำหน้าที่เชื่อมต่อ Datasocket Server เพื่ออ่านข้อมูลจาก Server และ แสดงผลที่ Web page มีขั้นตอนต่าง ๆ ดังต่อไปนี้

การเริ่มโปรเจ็คใหม่และการโหลด ActiveX Control

1. เริ่มใช้ VIsual Basic
2. ที่ New Project dialog ให้เลือก New ActiveX Control
3. คลิกเม้าส์ขวาที่ Toolbox และเลือก **Components** ในกรณีที่ VIsual Basic ไม่แสดง Toolbox ให้ไปเลือกที่ **Views >> Toolbox**
4. เลือก **National Instruments CW Datasocket** และ **National Instruments CW UI** ในกรณีที่ VIsual Basic ไม่แสดงรายชื่อดังกล่าว ให้กดปุ่ม **Brows** และเลือก cwds.ocx และ cwui.ocx จากไดเร็กทอรี่ของระบบวินโดว์
5. คลิก **OK**

การออกแบบส่วนประกอบ

จากรูปที่ 2.60 แสดงให้เห็นลักษณะของ DSReader ซึ่งลักษณะของ ฟอร์มจะเป็นหน้าต่างสีเทา หรือพื้นที่สำหรับวาง Controls และ Integrators เพื่อสร้างส่วนการติดต่อกับผู้ใช้ (User Interface) การสร้างส่วนติดต่อกับผู้ใช้มีขั้นตอน ดังต่อไปนี้



รูปที่ 2.60 ลักษณะของตัวอย่างโปรแกรม DSReader

1. วาง CommandButton บนฟอร์ม เปลี่ยน **Name** caption เป็น ConnectButton และ Caption property เป็น Connect

2. วาง CommandButton บนฟอร์ม เปลี่ยน **Name** caption เป็น DisconnectButton และ Caption property เป็น Disconnect

3. วาง TextBox บนฟอร์มสำหรับระบุ URL ต้นทาง จากนั้นเพิ่มป้ายชื่อ(label)เป็น Source

Label บนฟอร์มสำหรับแสดงสถานะการเชื่อมต่อ เปลี่ยน **Name** caption เป็น StatusLbl และ **Border Style** property เป็น **Fixed Single** จากนั้นลบตัวอักษรที่มีอยู่เดิมแล้วออก เพิ่มป้ายชื่อเป็น Status

5. วาง CwGraph control บนฟอร์ม

6. วาง Datasocket Control บนฟอร์ม (ขณะที่ RUN โปรแกรมจะไม่แสดงรูป)

2.4.3 การเขียนโค้ด

หลังจากที่วางส่วนประกอบต่าง ๆ ลงบนฟอร์มเรียบร้อยแล้ว จำเป็นต้องมีการเขียน VIsual Basic โค้ด เพื่อให้ส่วนประกอบที่วางไว้ตอบสนองตามเหตุการณ์ที่อาจจะเกิดขึ้น เช่นมีการเลื่อนเมาส์ หรือมีการเปลี่ยนค่าของส่วนประกอบ การเขียนโค้ดมีขั้นตอนต่าง ๆ ดังต่อไปนี้

1. ที่ **DSReader** component –ถ้าต้องการเชื่อมต่อ Datasocket Server โดยอัตโนมัติเพื่ออ่านและเปลี่ยนข้อมูลขณะที่มีเหตุการณ์คลิกเมาส์ที่ปุ่ม **Connect** ทำได้โดยดับเบิลคลิกที่ ปุ่ม **Connect** และเขียนโค้ดดังต่อไปนี้

```
Private Sub ConnectButton_Click()
      CWDatasocket1.ConnectTo Text1.Text, cwdsReadAutoUpdate
End Sub
```

2. ขณะที่ Datasocket Control เปลี่ยนแปลงข้อมูลเป็นข้อมูลใหม่ ต้องการพล็อตกร๊าฟทำได้โดยดับเบิลคลิกที่ Datasocket Control และเขียนโค้ดดังต่อไปนี้

```
Private Sub CWDatasocket1_OnDataUpdated(ByVal Data As CWDSLib.CWData)
      If IsArray(Data.Value) Then
      CWGraph1.PlotY Data.Value
      End If
End Sub
```

3. แสดงสถานะขณะมีการเชื่อมต่อที่และการยกเลิกการเชื่อมต่อที่ Status field ทำได้โดยดับเบิลคลิกที่ Datasocket Control แล้วเลือก OnstatusUpdated ที่ช่องเลือกเหตุการณ์ (Event list) และเขียนโค้ดดังต่อไปนี้

```
Private Sub CWDatasocket1_OnStatusUpdated(ByVal Status As Long, ByVal
Error AsLong, ByVal Message As String)
      StatusLbl.Caption = Message
End Sub
```

4. ในกรณีที่ต้องการยกเลิกการเชื่อมต่อโดยการกดปุ่ม **Disconnect** เพื่อยกเลิกการเชื่อมต่อกับ Datasocket Server และคืนทรัพยากรที่ Datasocket Server ใช้ให้ระบบ ทำได้โดยดับเบิลคลิกที่ Disconnect และเขียนโค้ดดังต่อไปนี้

```
Private Sub DisconnectButton_Click()
      CWDatasocket1.Disconnect
End Sub
```

2.4.4 การบันทึกไฟล์

1) การตั้งชื่อ Project และ Control

1. เปิด Project Explorer ซึ่งสามารถเข้าถึงได้โดย **View >> Project Explorer**
2. เลือก Project1 ใน. Project Explorer และเปลี่ยน **Name** property เป็น **Reader**
3. คลิกฟอร์ม และเปลี่ยน **Name** property เป็น **DSReader**

2) การบันทึก Project

1. เลือก File >> Save Project As
2. บันทึก Control เป็น DSReader.ctl
3. บันทึก Project เป็น DSReader.vbp

3) การสร้าง ActiveX Control ใน VIsual Basic

1. เลือก File >> Make DSReader.ocx VIsual Basic จะทำการสร้าง ActiveX control และบันทึกไว้ในไดเร็กทอรี่เดียวกับที่ผู้ใช้ได้ทำการบันทึก Project ไว้

2. เลือก **Project >> Reader Properties >> Component** จากนั้นเลือกเปลี่ยน Version Compatibility เป็น **Binary Compatibility**

3.บันทึก Project

2.5 การสร้าง HTML ไฟล์

การสร้าง HTML ไฟล์จาก ActiveX Control ทำได้โดยใช้โปรแกรม Application Setup Wizard ของ VIsual Basic 5.0 และโปรแกรม Package & Deployment Wizard ของ VIsual Basic 6.0 โปรแกรมทั้งสองนี้จะสร้าง Internet distribution package ซึ่งใน Internet distribution package นี้จะมีไฟล์ที่เรียกว่า Cabinet หรือไฟล์ที่มีส่วนขยายเป็น(. cab) และ HTML ไฟล์

ภายใน Cabinet ไฟล์นี้จะมีข้อมูลต่าง ๆ เกี่ยวกับ DSReder component ซึ่ง Cabinet ไฟล์นี้สามารถ Download และ ติดตั้งตัวเองได้โดยอัตโนมัติผ่านโปรแกรม Internet Explorer ทำให้ผู้ใช้สามารถเข้าไปใช้งาน DSReder ด้วย Web browser

หลังจากสร้าง HTML ไฟล์และ Cabinet ไฟล์ จาก Package and Deployment Wizard แล้ว ให้บันทึกไฟล์ทั้ง 2 ไว้ใน Root Directory ของ HTTP Server ซึ่งในโครงงานนี้ก็คือ C:\Program File\National Instruments\LabVIEW\WWW

หมายเหตุ การใช้โปรแกรม Package & Deployment Wizard สามารถศึกษาได้จากหนังสือ VIsual Basic 6.0 ทั่ว ๆ ไป

บทที่ 3
การออกแบบและการสร้าง

3.1 หลักการออกแบบโปรแกรม

การออกแบบโปรแกรมเพื่อควบคุมเครื่องมือวัดผ่านเครือข่ายอินเตอร์เน็ตโดย GPIB พอร์ตนั้น เป็นการผสมผสานเทคโนโลยีและความรู้หลาย ๆ อย่างรวมกัน เช่นการใช้โปรแกรม LabVIEW,โครงสร้างและการใช้งาน GPIB พอร์ต, เครือข่ายอินเตอร์เน็ต, ActiveX Control, และการสร้างโปรแกรมประยุกต์บนเครือข่ายอินเตอร์เน็ตโดย Visual Basic 6.0 ดังนั้นเพื่อให้การเขียนโปรแกรมง่ายขึ้น โครงงานนี้จึงได้แยกส่วนประกอบใหญ่ ๆ ของโครงงานเป็น 3 ส่วน แต่ละส่วนแยกกันโดยอิสระ ดังนี้

1. เป็นส่วนที่ใช้โปรแกรม LabVIEW สร้างขึ้นมา เพื่อควบคุมเครื่องมือวัดโดยไม่ผ่านเครือข่ายอินเตอร์เน็ต

2. Mix_server และ Mix_work เป็นส่วนที่ใช้โปรแกรม LabVIEW สร้างขึ้นมา เพื่อควบคุมเครื่องมือวัดโดยผ่านเครือข่ายอินเตอร์เน็ต โดยแบ่งเป็น 2 โปรแกรม ดังนี้

2.1 Mix_work ทำหน้าที่เป็นเครื่องลูกข่ายในการควบคุมเครื่องมือวัดผ่านเครือข่ายอินเตอร์เน็ต

2.2 Mix_server ทำหน้าที่เป็นเครื่องแม่ข่ายในการควบคุมเครื่องมือวัดผ่านเครือข่ายอินเตอร์เน็ต

3. Web_Control แบ่งออกเป็น 2 โปรแกรม ดังนี้

3.1 Server_www เป็นส่วนที่ใช้โปรแกรม LabVIEW สร้างขึ้นมา เพื่อทำหน้าที่เป็นเครื่องแม่ข่ายสำหรับ Web_Control

3.2 Web_Control เป็นส่วนที่ใช้ Visual Basic 6.0 สร้างขึ้นมาเพื่อเป็น Web Site สำหรับควบคุมเครื่องมือวัดผ่านเครือข่ายอินเตอร์เน็ต

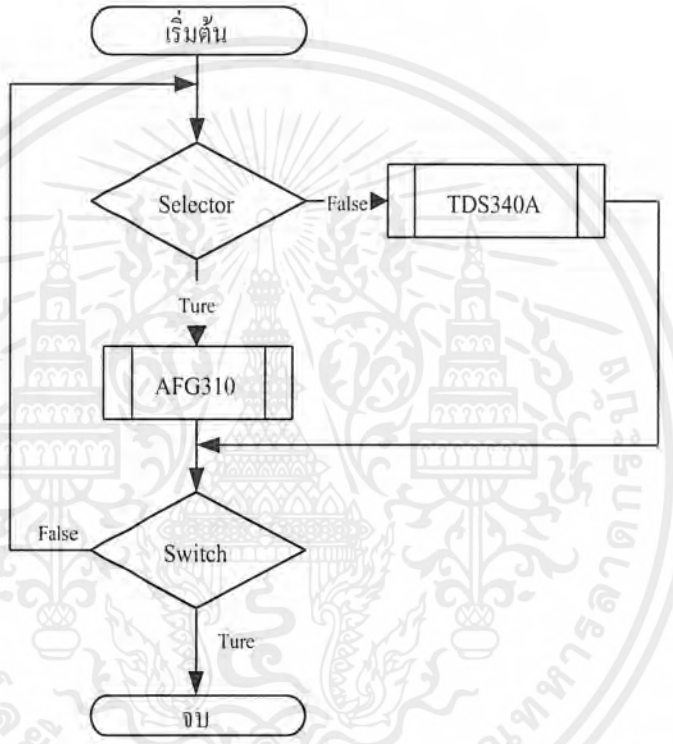
3.2 หลักการสร้างโปรแกรม

3.2.1 หลักการสร้างโปรแกรม Local_Control

เพื่อให้ง่ายแก่การทำความเข้าใจ จึงแบ่งโปรแกรม Local_Control เป็นส่วนดังต่อไปนี้

1) โปรแกรมหลัก (Main program)

โปรแกรมหลักแสดงเป็นรูปผังการทำงานได้ดังรูปที่ 3.1



รูปที่ 3.1 ผังการทำงานโปรแกรม Local_Control

หน้าที่การทำงานของส่วนประกอบต่าง ๆ มีดังต่อไปนี้

Selector เมื่อเริ่มต้นเข้าสู่โปรแกรม Selector จะทำหน้าที่เป็นตัวเลือกว่าจะควบคุมเครื่องมือวัดชนิดใด ซึ่งในโครงงานนี้ก็คือการเลือกว่าจะควบคุมเครื่องกำเนิดสัญญาณ AFG310 หรือเครื่องวัดสัญญาณ TDS340A นั้นเอง

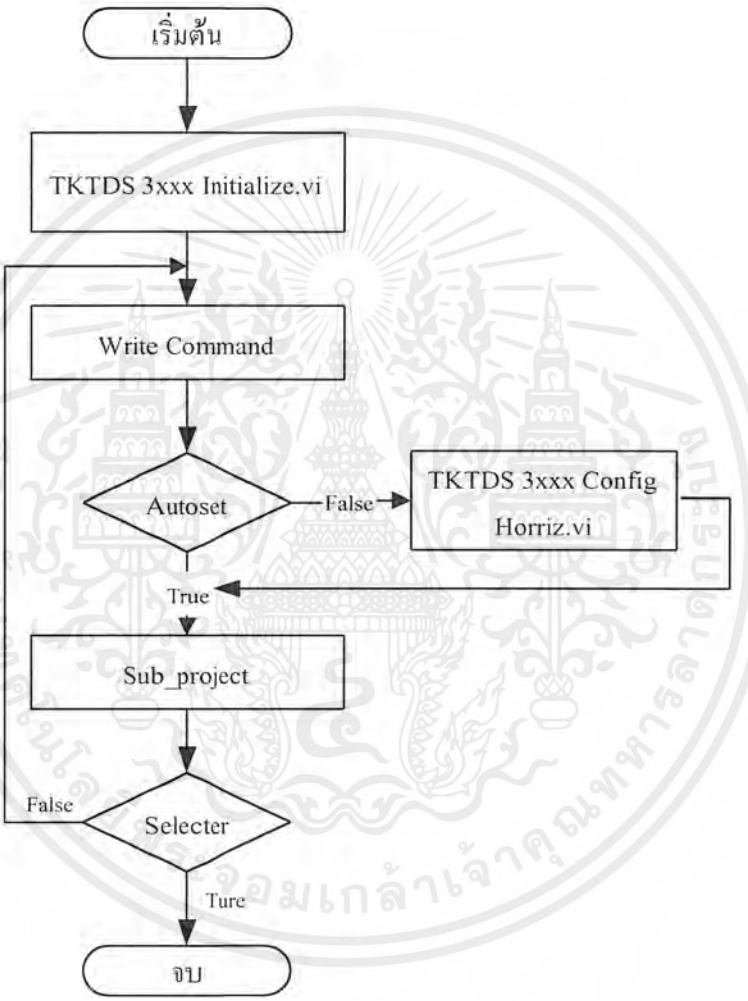
TDS340A เป็นโปรแกรมย่อย (sub VI) เพื่อควบคุมเครื่องวัดสัญญาณ TDS340A

AFGS310 เป็นโปรแกรมย่อยเพื่อควบคุมเครื่องกำเนิดสัญญาณ AFGS310

Switch ปกติเป็นค่าเท็จ (False) คือให้ย้อนกับไปทำการเลือกควบคุมเครื่องมือวัดใหม่ ถ้าเป็นจริง (True) ให้ออกจากโปรแกรม Local_Control

2) โปรแกรมย่อย TDS340A

ผังการทำงานของโปรแกรมย่อย TDS340A แสดงให้เห็นได้ดังรูปที่ 3.2



รูปที่ 3.2 ผังการทำงานของโปรแกรมย่อย TDS340A

หน้าที่การทำงานของส่วนประกอบต่าง ๆ มีดังต่อไปนี้

TKTDS 3xx Initialize.vi เป็นโปรแกรมย่อย เพื่อกำหนดค่าเริ่มต้นก่อนการควบคุมเครื่องกำเนิดสัญญาณ TDS340A เช่นการกำหนดค่าตำแหน่ง (Address) ของเครื่องมือวัด,การรีเซต (Reset) เครื่องมือวัดก่อนการใช้งาน

Write Command เป็นการเขียนคำสั่งเพิ่มเติม

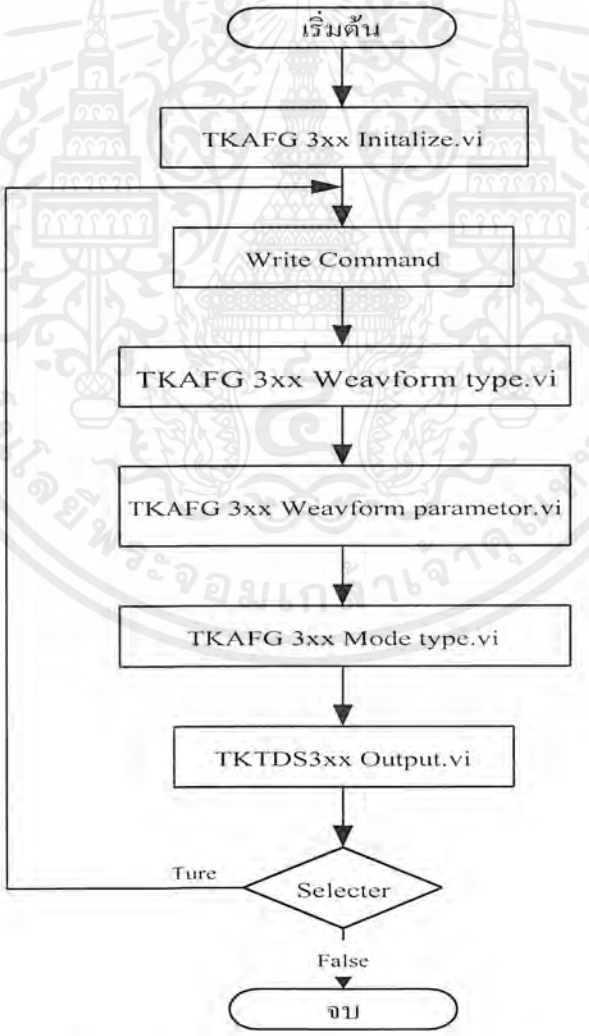
TKTSD 3xx Config Horiz.vi เป็นโปรแกรมย่อย เพื่อกำหนดค่าพารามิเตอร์ทางแนวนอนของเครื่องวัดสัญญาณ

Sub_project เป็นโปรแกรมย่อยเพื่อกำหนดค่า Measurements ต่าง ๆ เช่นกำหนดการวัดความถี่,วัดแอมปลิจูด และอ่านค่าที่ได้จากเครื่องวัดสัญญาณ

Selector เป็น Selector เดียวกันกับโปรแกรมหลัก ถ้าเป็นค่าจริง(มีการกดปุ่ม Selector ที่ Front Panel)จะเป็นการออกจากโปรแกรมย่อย ถ้าเป็นค่าเท็จ (ไม่มีการกดปุ่ม Selector ที่ Front Panel)จะปฏิบัติการในโปรแกรมย่อยต่อไป

3) โปรแกรมย่อย AFG310

ผังการทำงานของโปรแกรมย่อย AFG310 แสดงให้เห็นได้ดังรูปที่ 3.3



รูปที่ 3.3 ผังการทำงานของโปรแกรมย่อย AFG310

หน้าที่การทำงานของส่วนประกอบต่าง ๆ มีดังต่อไปนี้

TKAFG 3xx Initalize.vi เป็นโปรแกรมย่อยเพื่อกำหนดค่าเริ่มต้นก่อนการควบคุมเครื่องวัดสัญญาณ AFG310 เช่นการกำหนดค่าตำแหน่ง (Address) ของเครื่องมือวัด, การรีเซต (Reset) เครื่องมือวัดก่อนการใช้งาน

Write Command เป็นการเขียนคำสั่งเพิ่มเติม

TKAFG 3xx Weavform type.vi เป็นโปรแกรมย่อย เพื่อกำหนดชนิดของสัญญาณ เช่น สัญญาณซายน์,สัญญาณพัลส์

TKAFG 3xx Weavform parametor.vi เป็นโปรแกรม เพื่อกำหนดค่าพารามิเตอร์ต่าง ๆ ของเครื่องกำเนิดสัญญาณเช่น ค่าความถี่, ค่าแรงดัน, ออฟเซต (Offset) และเฟส (Phase)

TKAFG 3xx Mode type.vi เป็นโปรแกรมย่อย เพื่อกำหนดโหมดการทำงานของเครื่องกำเนิดสัญญาณ เช่น Countinuous, Tring, Burst และ withcount

TKTDS3xx Output.vi เป็นโปรแกรมย่อย ที่จำเป็นต้องมี เพื่อใช้สั่งให้เครื่องกำเนิดสัญญาณผลิตสัญญาณตามค่าพารามิเตอร์ต่าง ๆ ที่กำหนดไว้ออกมาทางเอาต์พุต

Selector ทำงานเช่นเดียวกันกับ Selector ที่โปรแกรมหลัก

Block Diagram ของโปรแกรม Local_Control แสดงให้เห็นได้จากรูปที่ ก.1 และ ก.2

3.2.2 หลักการสร้างโปรแกรม Mix_server และ Mix_work

การสร้างโปรแกรมในส่วนนี้สามารถแบ่งออกได้ 2 โปรแกรม ตามที่ได้ออกแบบไว้ดังนี้

1) โปรแกรม Mix_server

เพื่อให้ง่ายแก่การทำความเข้าใจ จึงแบ่งโปรแกรม Mix_server ออกเป็น 3 ส่วน ดังนี้

1.1) โปรแกรมหลักของ Mix_server

ผังการทำงานโปรแกรมหลักของ Mix_server แสดงให้เห็นได้ ดังรูปที่ 3.4 หน้าที่การทำงานของส่วนประกอบต่าง ๆ มีดังต่อไปนี้

URL (Uniform Resource Locator) เป็นชื่อที่ใช้แทน IP Address เพื่อความสะดวก ในโครงงานนี้ได้ทำเป็นตัวเลือก 2 ทิศทางไว้สำหรับเลือกให้งาน นั้นคือ Local host และ DNS host

Open Connection ReadAutoupdete และ Open Connection WriteAutoupdate เป็นการบอกให้ LabVIEW เริ่มการเชื่อมต่อโพรโตคอล Dstp ซึ่งโปรแกรม Mix_server นี้จำเป็นต้องมีการอ่านและเขียนข้อมูล ดังนั้นโครงงานนี้จึงใช้ การเชื่อมต่อโพรโตคอล Dstp ทั้งแบบอ่าน (Read) และแบบเขียน (Write) โดยแยก URL ของทั้ง 2 แบบออกโดยอิสระ

Datasocket Read String เป็นการอ่านข้อมูลที่เป็นข้อความจากเครื่องคอมพิวเตอร์ลูกข่าย (Work station)

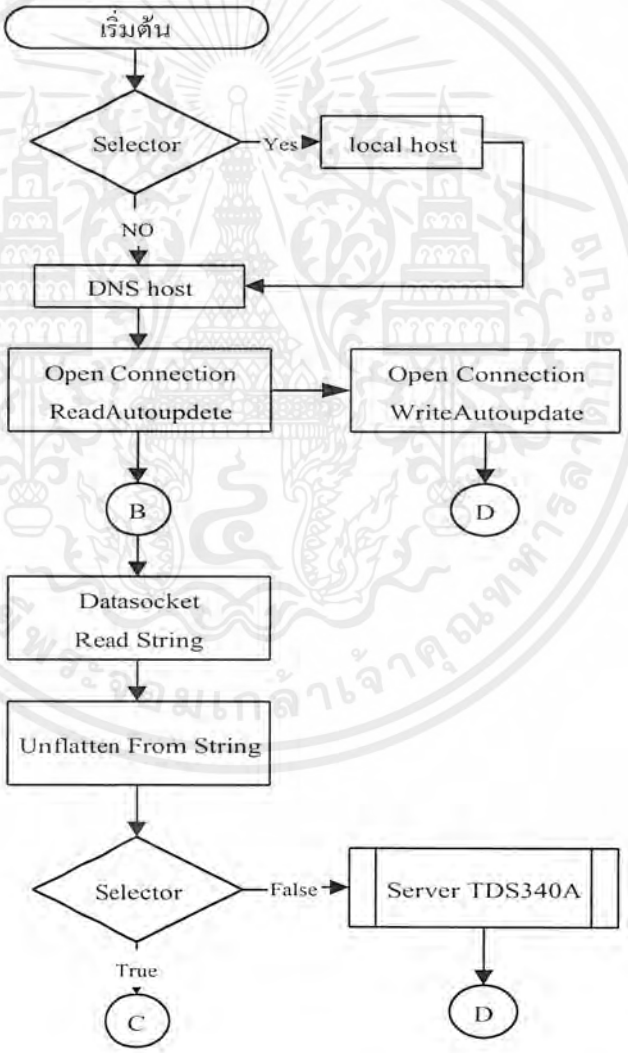
Unflatten From String แปลงข้อมูลที่เป็นข้อความให้กลับมาเป็นข้อมูลจริง จะกล่าวโดยละเอียดในเรื่อง Flatten to String

Selector เป็นตัวเลือก 2ทิศทาง ซึ่งค่าบูลีน (Boolean) ที่ได้จะขึ้นอยู่กับข้อมูลที่ส่งมาจากเครื่องลูกข่าย Selector จะทำหน้าที่เลือกว่าจะให้เครื่องมือวัดที่ต่ออยู่กับเครื่องคอมพิวเตอร์แม่ข่าย เครื่องใดทำงาน โดยค่าปกติคือเครื่องกำเนิดสัญญาณ AFG310

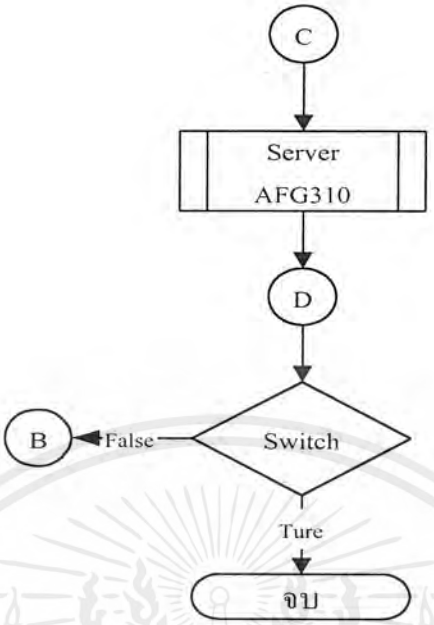
Server TDS340A เป็นโปรแกรมย่อย ใช้ควบคุมเครื่องวัดสัญญาณ

Server AFG310 เป็นโปรแกรมย่อย ใช้ควบคุมเครื่องกำเนิดสัญญาณ

Switch เช่นเดียวกันกับ Switch ของโปรแกรมหลัก Local_Control



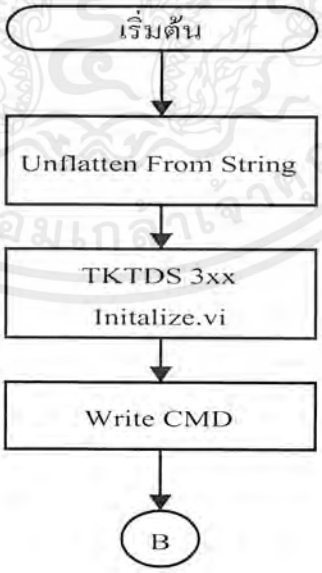
รูปที่ 3.4 ผังการทำงานโปรแกรมหลัก Mix_server



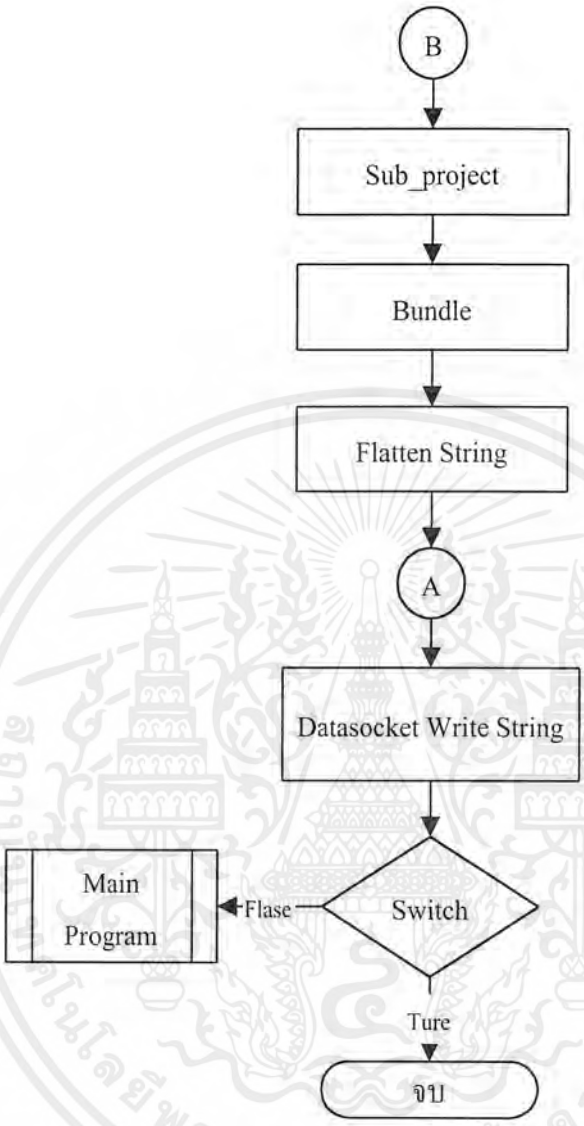
รูปที่ 3.4 ผังการทำงานโปรแกรมหลัก Mix_server (ต่อ)

1.2) โปรแกรมย่อย Server TDS340A

ผังการทำงานของโปรแกรมย่อย Server TDS340A แสดงให้เห็นได้ดังรูปที่ 3.5



รูปที่ 3.5 ผังการทำงานโปรแกรมย่อย Server TDS340A



รูปที่ 3.5 ผังการทำงานโปรแกรมย่อย Server TDS340A (ต่อ)

หน้าที่การทำงานของส่วนประกอบต่าง ๆ มีดังต่อไปนี้

Bundle ทำหน้าที่รับผูกข้อมูลต่าง ๆ ที่ได้จากโปรแกรมย่อย Sub_project ให้เป็นข้อมูลชนิดเดียว เนื่องจากข้อมูลที่ได้จากโปรแกรมย่อย Sub_project เป็นข้อมูลหลายชนิด จึงจำเป็นต้องมีการรวมข้อมูลเหล่านั้น เพื่อให้สามารถแปลงข้อมูลนั้นให้เป็นข้อมูลแบบข้อความได้ในครั้งเดียว รายละเอียดของ Bundle ดูได้จาก บทที่ 2 หัวข้อการใช้โปรแกรม LabVIEW

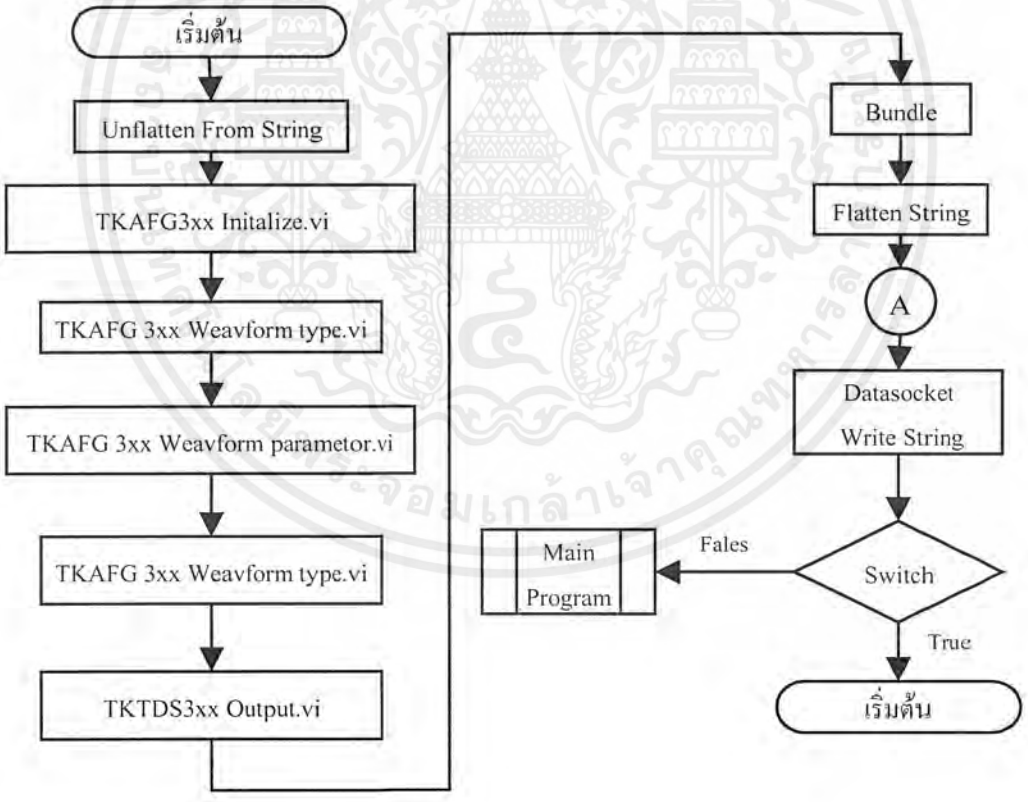
Flatten to String เป็นการนำข้อมูลที่ได้จาก Bundle มาแปลงให้เป็นข้อมูลแบบข้อความเนื่องจากการเขียนข้อมูลผ่าน Datasocket นั้นURL หนึ่ง URL สามารถอ่านหรือเขียนข้อมูล

ได้เพียงชนิดเดียว ดังนั้นเพื่อให้สามารถเขียนข้อมูลทั้งหมดผ่าน Datasocket ได้โดยใช้เพียงหนึ่ง URL จึงจำเป็นต้องใช้ Fatten to String แปลงข้อมูลเหล่านั้นให้เป็นข้อมูลแบบข้อความ

Datasocket Write String เป็นการเขียนข้อมูลชนิดข้อความ ซึ่งได้จาก Flatten to String ผ่าน Datasocket ไปให้เครื่องลูกข่าย

1.3) โปรแกรมย่อย Server AFG310

ผังการทำงานของโปรแกรมย่อย Server AFG310 แสดงให้เห็นได้ ดังรูปที่ 3.6 หน้าที่การทำงานของส่วนประกอบต่าง ๆ คล้ายกับโปรแกรมย่อย Server TDS340A คือทำการแปลงข้อมูลที่อ่านได้จากเครื่องคอมพิวเตอร์ลูกข่าย ที่เป็นข้อมูลชนิดข้อความ ให้เป็นข้อมูลชนิดต่าง ๆ โดยใช้ Unflatten From String จากนั้นนำข้อมูลเหล่านั้นไปประมวลตามขั้นตอน แล้วนำผลลัพธ์ที่ได้จากการประมวลผลนั้น แปลงให้เป็นข้อมูลชนิดข้อความโดยใช้ Bundle และ Flatten to String แล้วเขียนกลับไปยังเครื่องคอมพิวเตอร์ลูกข่ายอีกครั้ง



รูปที่ 3.6 ผังการทำงานของโปรแกรมย่อย Server AFG310

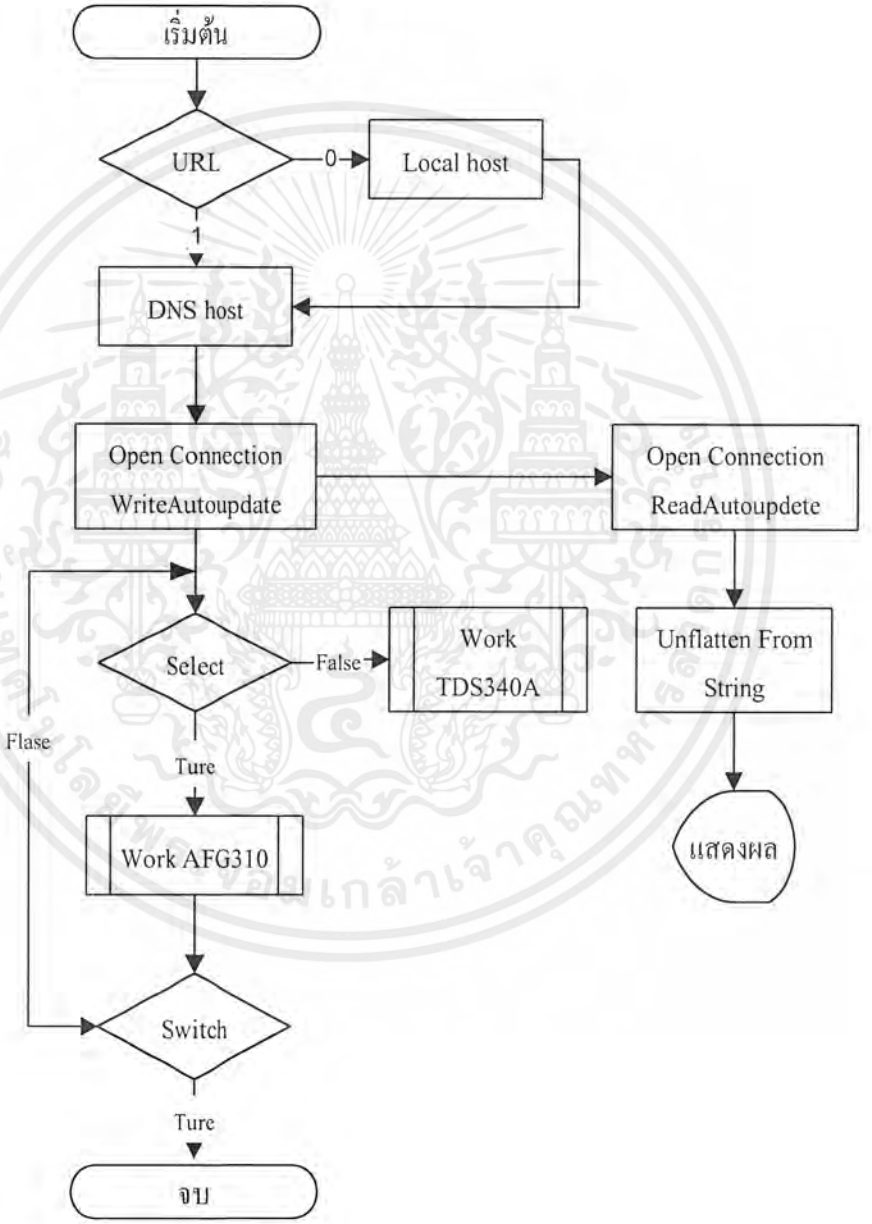
Block Diagram ของโปรแกรม Mix_server แสดงให้เห็นได้จากรูปที่ ก.3 และ ก.4

2) **โปรแกรม Mix_wrok**

เพื่อให้ง่ายแก่การทำความเข้าใจ จึงแบ่งโปรแกรม Mix_work ออกเป็น 2 ส่วน ดังนี้

2.1) โปรแกรมหลัก Mix_work

ผังการทำงานโปรแกรมหลักของ Mix_work แสดงให้เห็นได้ดังรูปที่ 3.7



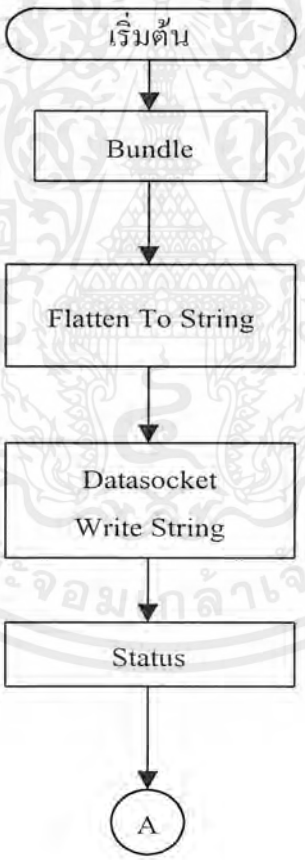
รูปที่ 3.7 ผังการทำงานของโปรแกรมหลัก Mix_work

หน้าที่การทำงานของส่วนประกอบต่าง ๆ มีดังต่อไปนี้

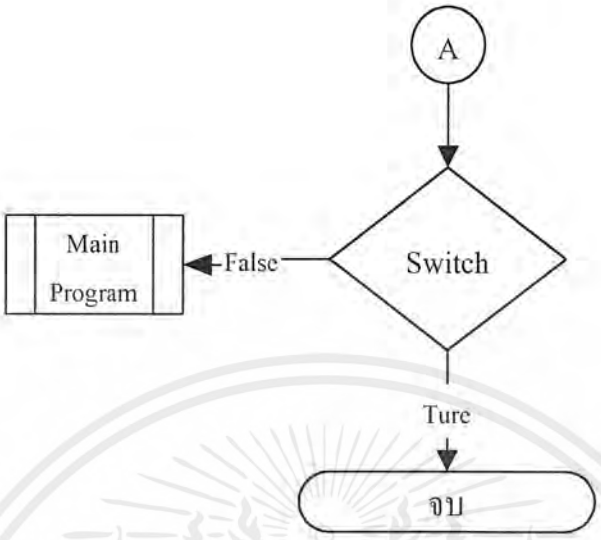
จากรูปผังการทำงานของโปรแกรมหลัก Mix_work จะเห็นว่าคล้ายกันกับรูปผังการทำงานของโปรแกรมหลัก Mix_server มาก โดยจะมีข้อแตกต่างกันที่โปรแกรมย่อย และมีข้อสังเกตว่าโปรแกรม Mix_work จะทำการเขียนข้อมูลไปให้โปรแกรม Mix_server ก่อน โดยการเขียนข้อมูลจะทำที่โปรแกรมย่อย work TDS340A หรือ work AFG310 จากนั้นก็จะอ่านค่าจากเครื่องคอมพิวเตอร์แม่ข่ายกลับมา แล้วทำการแปลงข้อมูลโดย Unflatten From String เพื่อแสดงผลที่เครื่องคอมพิวเตอร์ลูกข่าย ซึ่งเป็นการทำงานที่ตรงข้ามกับโปรแกรม Mix_server

2.2) โปรแกรมย่อย work TDS340A และ work AFG310

ผังการทำงานโปรแกรมหลักของ work TDS340A แสดงให้เห็นได้ดังรูปที่ 3.8



รูปที่ 3.8 ผังการทำงานโปรแกรมย่อย Work TDS34A และ work AFG310



รูปที่ 3.8 ผังการทำงานโปรแกรมย่อย Work TDS34A และ works AFG310 (ต่อ)

หน้าที่การทำงานของส่วนประกอบต่าง ๆ มีดังต่อไปนี้

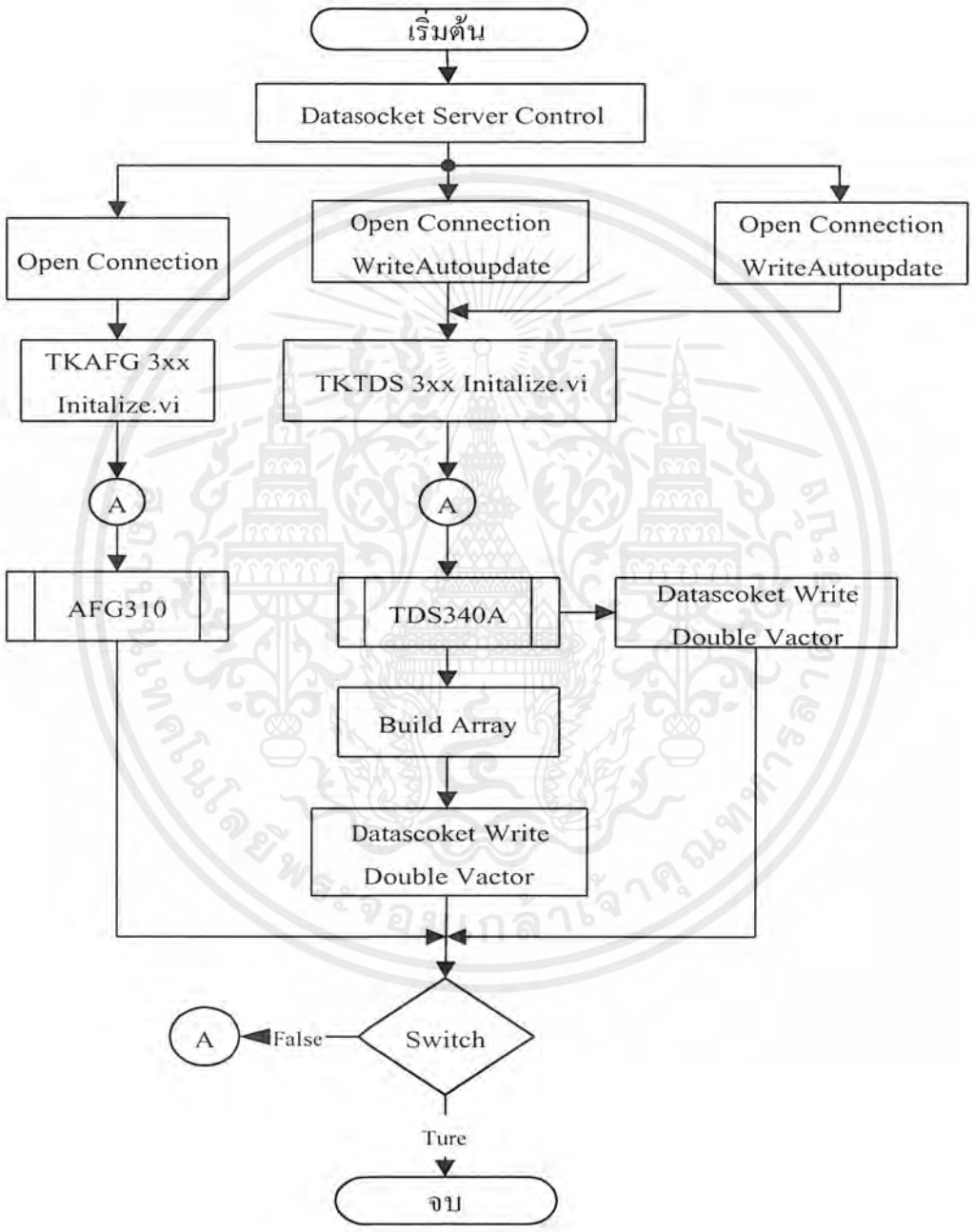
โดยปกติแล้วโปรแกรม Mix_work จะทำงานที่เครื่องคอมพิวเตอร์ลูกข่ายที่ไม่ได้ต่ออยู่กับเครื่องมือวัด การทำงานของโปรแกรมจึงมีเฉพาะส่วนของการเขียนข้อมูล ไปสั่งงานเครื่องมือวัดที่ต่ออยู่กับเครื่องคอมพิวเตอร์แม่ข่าย และอ่านค่าที่ได้กลับมาแสดงผล โดยโปรแกรมย่อย work TDS340A และ work AFG310 ซึ่งมีลำดับการทำงานดังนี้คือ เมื่อโปรแกรมย่อยได้รับข้อมูลจากโปรแกรมหลักแล้วก็จะแปลงข้อมูลต่าง ๆ นั้นให้เป็นข้อมูลชนิดข้อความจากโดย Bundle และ Flatten to String จากนั้นจะเขียนข้อมูลไปยังเครื่องแม่ข่ายข้อมูลโดย Datasocket Write String การทำงานอันดับสุดท้ายของโปรแกรมย่อย TDS340A คือการตรวจสอบค่าของ Switch ถ้าปกติเป็นค่า “เท็จ” คือให้ย้อนกลับไปโปรแกรมหลักอีกครั้ง แต่ถ้ามีการกด Switch ค่าจะเปลี่ยนเป็น “จริง” เป็นการจบโปรแกรมMix_work ทั้งหมด

Block Diagram ของโปรแกรม Mix_work แสดงให้เห็นได้จากรูปที่ ก.5 และ ก.6

3.2.3 หลักการสร้างโปรแกรม Server_www และ Web_Control

1) โปรแกรม Server_www

ผังการทำงานโปรแกรมหลักของ Server_www แสดงให้เห็นได้ดังรูปที่ 3.9



รูปที่ 3.9 ผังการทำงานโปรแกรม Server_www

โปรแกรม Server_www เป็นโปรแกรมที่ทำหน้าที่เป็น Web Server ทำการรับข้อมูลจากจากโปรแกรม Web_Control ของเครื่องคอมพิวเตอร์ลูกข่าย และประมวลผลต่าง ๆ ตามข้อมูลที่ได้รับมาแล้วเขียนผลลัพธ์ที่ได้จากการประมวลผลนั้นกลับไปยังคอมพิวเตอร์ลูกข่ายอีกครั้งหนึ่ง ในโปรแกรมส่วนที่ 3 นี้เครื่องคอมพิวเตอร์ลูกข่ายสามารถควบคุมได้เฉพาะเครื่องกำเนิดสัญญาณเท่านั้น แต่ในการอ่านค่ากลับนั้นจะอ่านค่าจากเครื่องวัดสัญญาณแทน โดยที่การสร้างโปรแกรม Server_www นั้นได้แยก URL สำหรับการอ่านข้อมูลจากเครื่องคอมพิวเตอร์ลูกข่าย 1 URL และการเขียนข้อมูลที่ได้จากเครื่องวัดสัญญาณ อีก 2 URL ซึ่งทั้ง 3 โปรแกรมนี้ทำงานแยกจากการโดนอิสระ หน้าที่การทำงานของส่วนประกอบต่าง ๆ มีดังต่อไปนี้

Open Connection 1 URL และ Open Connection WriteAutoupdete 2 URL เป็นการการเริ่มต้นใช้โพรโตคอล Dstp เช่นเดียวกันกับโปรแกรม Mix_sever และ Mix_work

โปรแกรมย่อย AFG310 เป็นโปรแกรมย่อยที่ใช้กำหนดค่าต่างๆ ของเครื่องกำเนิดสัญญาณ เช่นเดียวกันกับโปรแกรมย่อย AFG310 ของโปรแกรมต่าง ๆ ที่ได้กล่าวไปแล้ว

Sub_project เป็นโปรแกรมย่อยที่ใช้อ่านค่าต่าง ๆ ของเครื่องวัดสัญญาณ เช่นเดียวกันกับโปรแกรมย่อย TDS340A ของโปรแกรมต่าง ๆ ที่ได้กล่าวไปแล้ว แต่มีข้อสังเกตอยู่ว่าโปรแกรมย่อย TDS340A จะไม่มีการรับข้อมูลจากเครื่องคอมพิวเตอร์ลูกข่าย แต่จะมีการเขียนค่าที่วัดได้จากเครื่องวัดสัญญาณกลับไปให้เครื่องคอมพิวเตอร์ลูกข่าย โดยการเขียนค่ากลับจะเขียนผ่าน 2 URL ซึ่ง URL ที่ 1 เขียนค่าพารามิเตอร์ต่าง ๆ เช่น ค่าแรงดัน, ค่าความถี่, ค่าแรงดัน RMS กลับ ไปให้เครื่องลูกข่าย ส่วน URL ที่ 2 จะเขียนรูปคลื่นสัญญาณกลับให้เครื่องลูกข่าย

Datasocket Write Double Vector เป็นการเขียนข้อมูลผ่านโพรโตคอล Dstp คล้ายกับ Datasocket write String แต่ Datasocket Write String จะเป็นการเขียนข้อมูลชนิดข้อความ ส่วน Datasocket Write Double vector จะเป็นการเขียนข้อชนิดจำนวนจริงที่เป็นอาเรย์

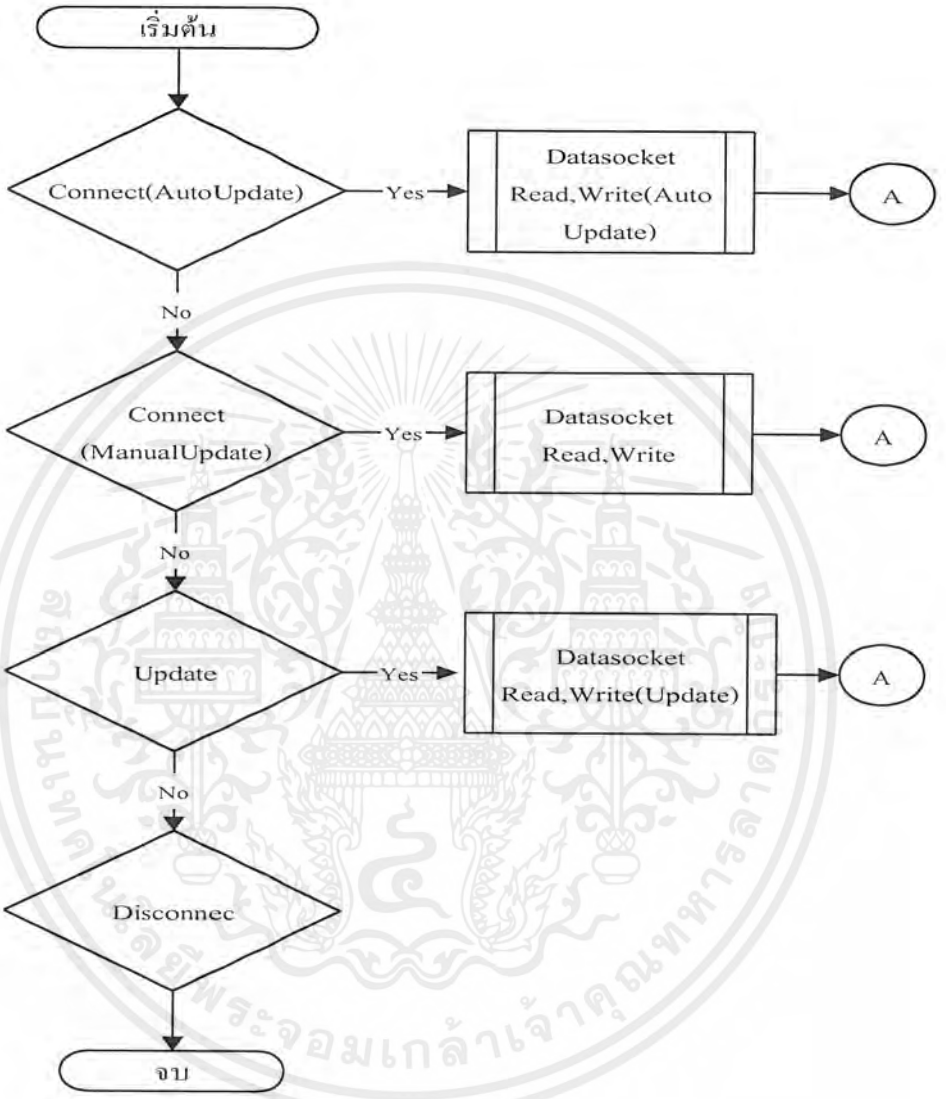
Switch ทำงานเช่นเดียวกับ Switch ในทุก ๆ โปรแกรมที่ได้กล่าวไปแล้ว

Block Diagram ของโปรแกรม Server_www แสดงให้เห็นได้จากรูปที่ ก.7

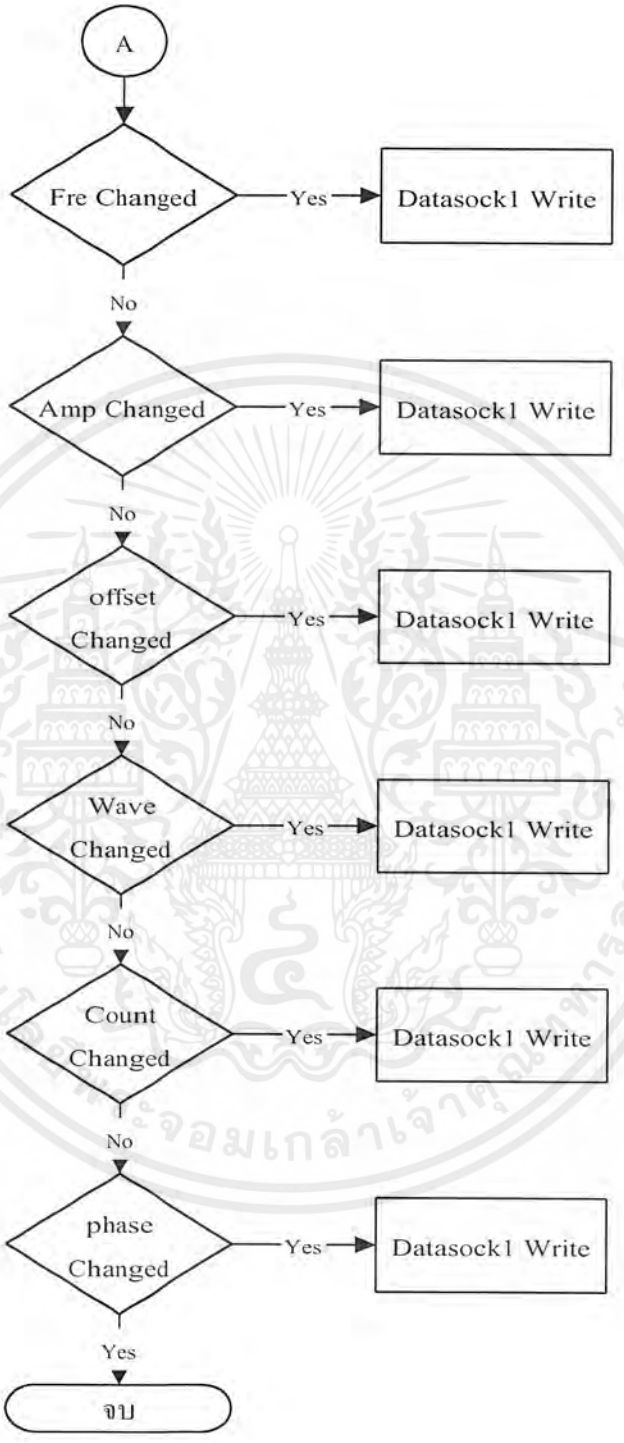
2) โปรแกรม Web_Control

ผังการทำงานของโปรแกรมย่อย Web_Control แสดงให้เห็นได้ ดังรูปที่ 3.10

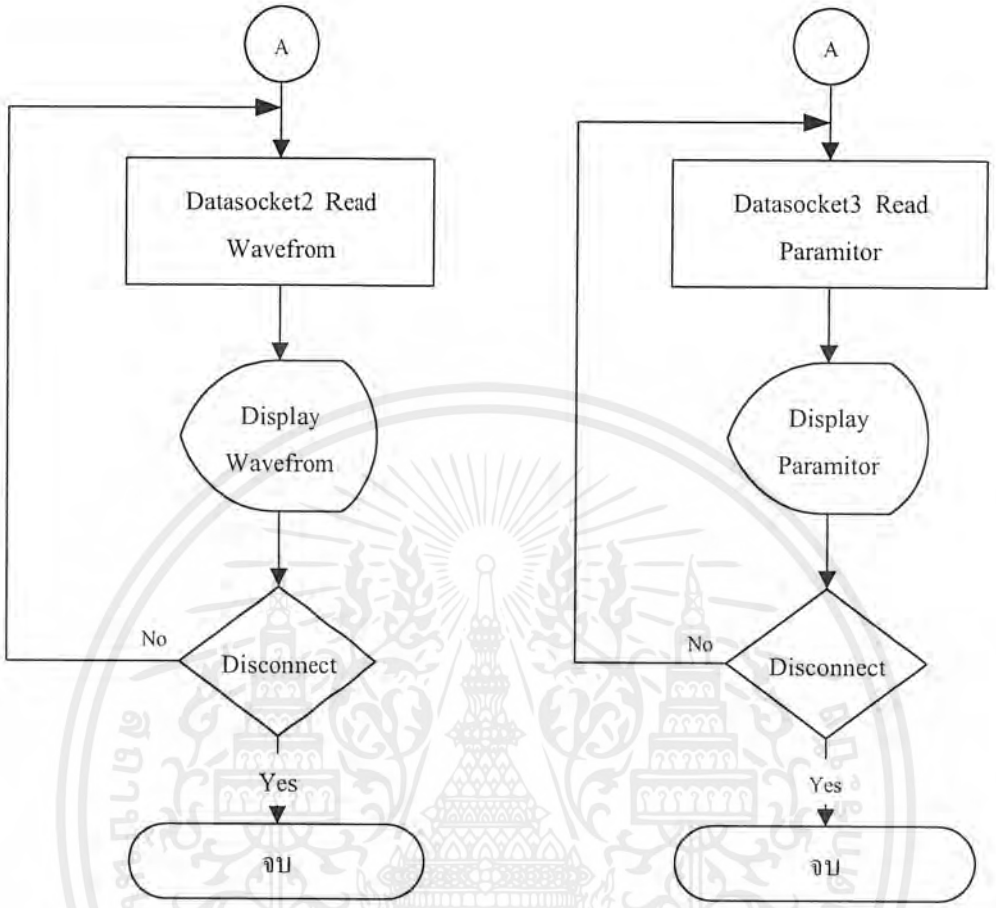
เป็นโปรแกรมที่สร้างขึ้นโดย Visual Basic 6.0 ซึ่งเป็นโปรแกรมที่คอมไพล์เป็น ActiveX และใช้ Package & Deployment Wizard สร้าง Web Site จาก ActiveX เพื่อทำหน้าที่เป็น Web Site สำหรับควบคุมเครื่องมือวัดผ่านเครือข่ายอินเตอร์เน็ต ผังการทำงานของโปรแกรม Web_Contorl แสดงได้ ดังรูปที่ 3.10 รูปที่ 3.11 และรูปที่ 3.12



รูปที่ 3.10 ผังการทำงานโปรแกรม Web_Control



รูปที่ 3.11 ผังการทำงานโปรแกรมย่อย Datasocket Write (Autoupdate)



รูปที่ 3.12 ผังการทำงานโปรแกรมย่อย Datasocket Read

โค้ดของโปรแกรม Web_control แสดงให้เห็นได้โดยภาคผนวกที่ ก.8

บทที่ 4

การทดลองและผลการทดลอง

จากบทที่แล้วได้กล่าวถึงการออกแบบและการสร้างโครงงาน ซึ่งโครงงานนี้ได้แบ่งโปรแกรมสำหรับควบคุมเครื่องมือวัดออกเป็น 3 ส่วนดังนี้

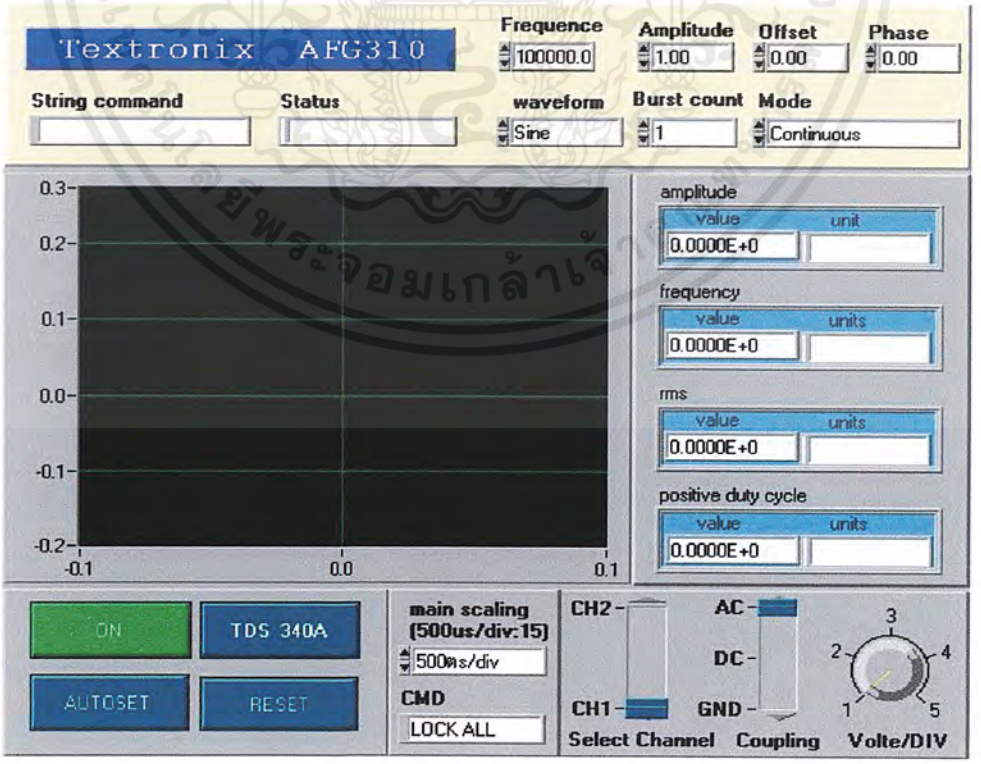
1. โปรแกรม Local_Control
2. โปรแกรม Mix_server และ Mix_work
3. โปรแกรม Server_www และ Web_Control

ในบทนี้จะกล่าวถึง ลักษณะโปรแกรมและการใช้งานโปรแกรม, ลักษณะอุปกรณ์และการติดตั้ง, การทดลองและผลการทดลองของโปรแกรมทั้ง 3 ส่วนที่กล่าวมาแล้วนั้นโดยละเอียด

4.1. ลักษณะโปรแกรมและการใช้งานโปรแกรม

4.1.1. โปรแกรม Local_Control

1) โปรแกรม Local_Control มีลักษณะโปรแกรมดังรูปที่ 4.1



รูปที่ 4.1 ลักษณะของโปรแกรม Local_Contol

การใช้งานโปรแกรม

เมื่อเข้าสู่โปรแกรม Local_Control โปรแกรมจะทำงานเองโดยอัตโนมัติ ปุ่มปรับและช่องแสดงผลต่าง ๆ มีความหมายและการใช้งานดังนี้

ปุ่ม ON กดเมื่อต้องการหยุดโปรแกรม

ปุ่ม TDS340A ถ้าไม่มีการกดจะเป็นการเลือกควบคุมเครื่องวัดสัญญาณ TDS340A ถ้ากดจะเป็นการเลือกควบคุมเครื่องกำเนิดสัญญาณ AFG310

ปุ่ม AUTOSET เป็นปุ่มที่สั่งให้เครื่องวัดสัญญาณปรับค่าต่าง ๆ ให้เหมาะสมโดยอัตโนมัติ

ปุ่ม RESET กดเมื่อต้องการ Reset เครื่องวัดสัญญาณ TDS340A

Main Scaling เป็นการปรับ Time Division ของเครื่องวัดสัญญาณ TDS340A

CMD เป็น String Control ไว้รับคำสั่งพิเศษจากผู้ใช้งาน

Select Channel ใช้เลือกช่องวัดสัญญาณ (Channel) ของเครื่องวัดสัญญาณ TDS340A

Coupling. ใช้กำหนดแบบการคลัปปลิ้งของเครื่องวัดสัญญาณ TDS340A

Volt/DIV. ใช้เป็นการกำหนดค่า Volt Division ของเครื่องวัดสัญญาณ TDS340A

Graph ใช้แสดงผลรูปสัญญาณที่ได้จากเครื่องวัดสัญญาณ TDS340A

Amplitude ใช้แสดงผลค่าแอมปลิจูดที่อ่านได้ เป็นค่าจากยอดถึงยอด (Peak to Peak) มีหน่วยเป็นโวลต์ (V)

Frequency ใช้แสดงผลค่าความถี่ที่อ่านได้ มีหน่วยเป็นเฮิร์ต (Hz)

Rms. ใช้แสดงผลค่าโวลต์ RMS มีหน่วยเป็นโวลต์ (V)

Positive duty cycle ใช้แสดงผลค่า Duty Cycle มีหน่วยเป็นเปอร์เซ็น (%)

String Command เป็น String Control ไว้รับคำสั่งพิเศษจากผู้ให้ สำหรับเครื่องกำเนิดสัญญาณ AFG310

Status ใช้แสดงสถานะการทำงานของโปรแกรมเช่น กำลังปฏิบัติงาน (Inprocess), ปฏิบัติการเสร็จสมบูรณ์ (Complete)

Frequency เป็นช่องรับค่าความถี่ต้องการให้เครื่องกำเนิดสัญญาณ AFG310 ผลิตขึ้นมา

Amplitude เป็นช่องรับค่าแรงเคลื่อนไฟฟ้าที่ต้องการให้เครื่องกำเนิดสัญญาณ AFG310 ผลิตขึ้นมา

Offset เป็นการกำหนดค่า Offset ให้แก่เครื่องกำเนิดสัญญาณ AFG310

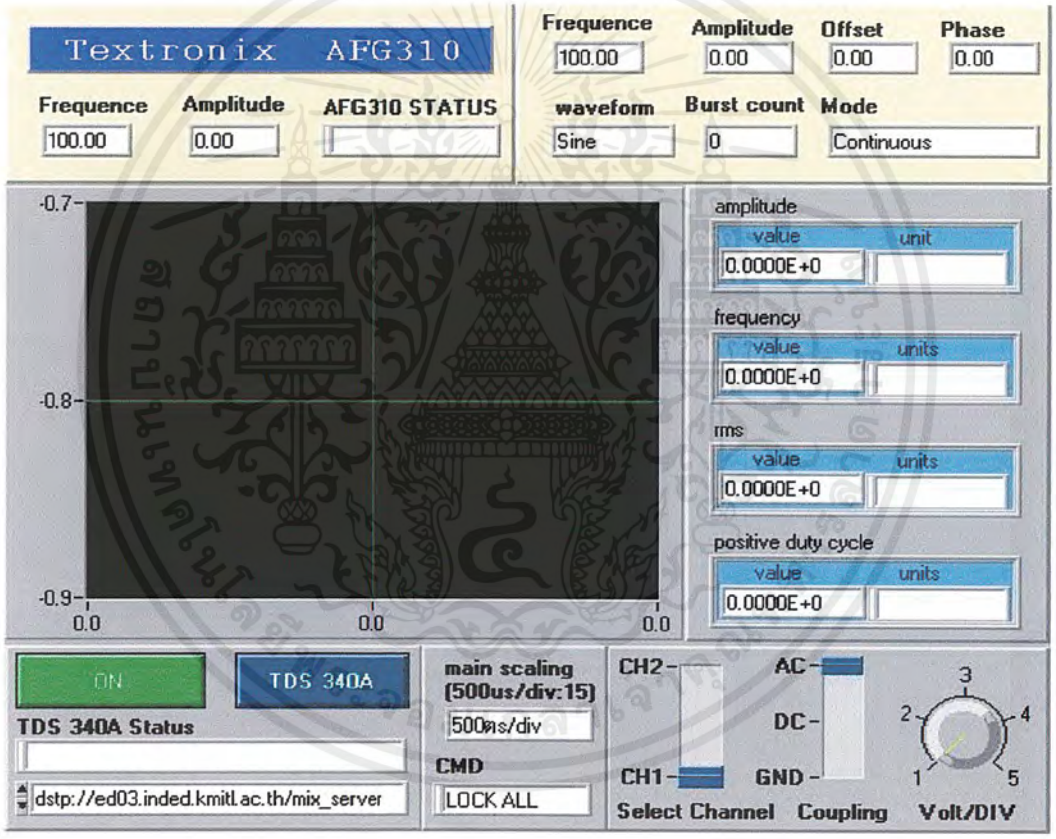
Phase เป็นการกำหนดค่าเฟสให้กับเครื่องกำเนิดสัญญาณ AFG310

Wave from เป็นการเลือกชนิดของรูปคลื่นสัญญาณที่ต้องการให้เครื่องกำเนิดสัญญาณ AFG310 ผลิต

Mode เป็นการกำหนดโหมดการทำงานของเครื่องกำเนิดสัญญาณ AFG310

4.1.2. โปรแกรม Mix_server และ Mix_work

1.) โปรแกรม Mix_server มีลักษณะโปรแกรมดังรูปที่ 4.2



รูปที่ 4.2 ลักษณะของโปรแกรม Mix_server

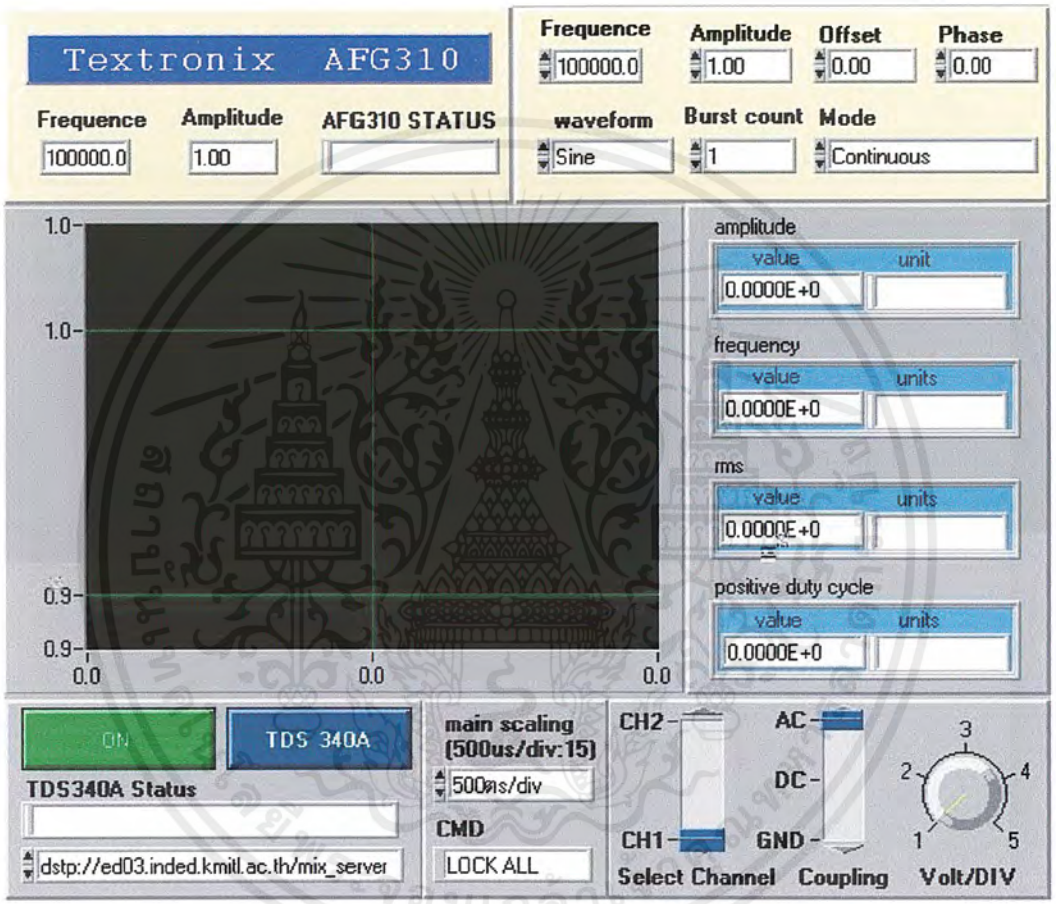
การใช้งานโปรแกรม

โปรแกรม Mix_server นี้จะใช้งานที่เครื่องคอมพิวเตอร์แม่ข่าย โดยการรันโปรแกรมเพียงครั้งเดียว จากนั้นโปรแกรม Mix_server จะค่อยรับข้อมูลจากเครื่องคอมพิวเตอร์ลูกข่ายและประมวลผลเองโดยอัตโนมัติ ปุ่มปรับและการแสดงผลต่าง ๆ เหมือนกับโปรแกรม Local_Control (ในกรณีของ Mix_server จะไม่สามารถปรับค่าใด ๆ ได้) ส่วนที่แตกต่างจาก Local_Control มีดังนี้

URL ใช้เลือก URL ก่อนการเริ่มโปรแกรม

TDS340A Status และ AFG310 Status เป็นการแสดงสถานะการเชื่อมต่อ Datasocket ผ่านเครือข่ายอินเตอร์เน็ต โดยแยกสถานะของเครื่องมือวัดทั้ง 2 ออกจากกันโดยอิสระ

2.) โปรแกรม Mix_work มีลักษณะโปรแกรมดังรูปที่ 4.3



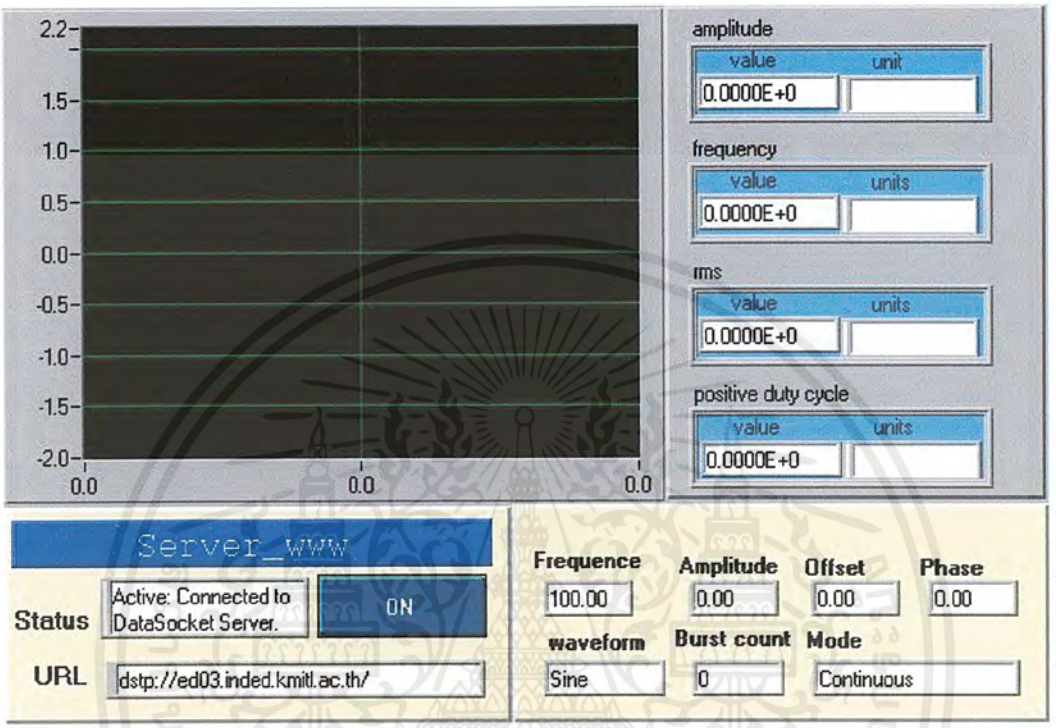
รูปที่ 4.3 ลักษณะของโปรแกรม Mix_work

การใช้งานโปรแกรม

โปรแกรม Mix_work นี้จะใช้งานที่เครื่องคอมพิวเตอร์ลูกข่าย เมื่อรันโปรแกรมแล้ว Mix_work จะทำหน้าที่เขียนข้อมูลไปที่เครื่องคอมพิวเตอร์แม่ข่าย และอ่านข้อมูลที่ได้จากการประมวลผลของแม่ข่ายกลับมาเพื่อแสดงผลที่เครื่องคอมพิวเตอร์ลูกข่ายอีกครั้ง ปุ่มปรับและการแสดงผลต่าง ๆ เช่นเดียวกับโปรแกรม Mix_server

4.1.3. โปรแกรม Server_www และ Web_Control

1.) โปรแกรม Server_www มีลักษณะโปรแกรมดังรูปที่ 4.4

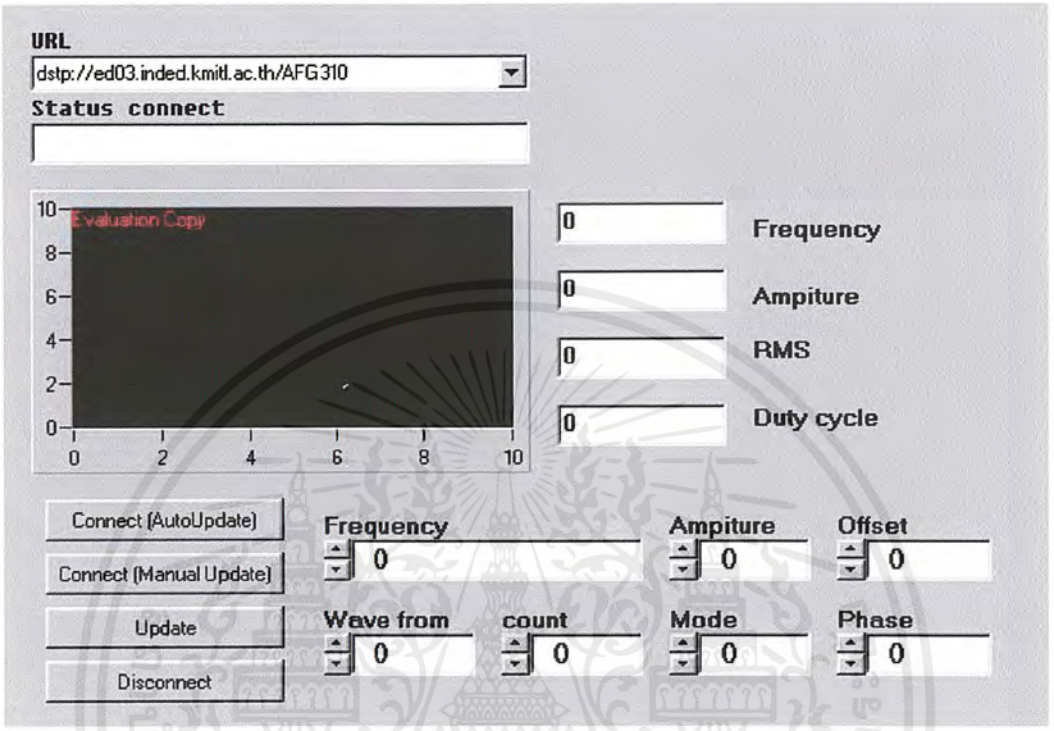


รูปที่ 4.4 ลักษณะโปรแกรม Serve_www7

การใช้งานโปรแกรม

จากรูปที่ 4.4 จะเห็นได้ว่าโปรแกรม Server_www ไม่มีปุ่มของการควบคุมเครื่องวัดสัญญาณ TDS340A และเครื่องกำเนิดสัญญาณ AFG310 ทั้งนี้เนื่อง โปรแกรม Server_www ทำหน้าที่เป็นเครื่องแม่ข่ายที่ค่อยรับคำสั่งจาก Web Page (โปรมแกรม Web_Control) เท่านั้น ดังนั้นจึงไม่จำเป็นต้องมีการปรับค่าที่โปรแกรม Server_www อีก การแสดงผลต่าง ๆ มีความหมายเช่นเดียวกับโปรแกรม Mix_server

2.) โปรแกรม Web_Control มีลักษณะโปรแกรมดังรูปที่ 4.5



รูปที่ 4.5 ลักษณะโปรแกรม Web_Control

การใช้งานโปรแกรม

ก่อนที่จะใช้โปรแกรม Web_Control จำเป็นต้องรันโปรแกรม Server_www ที่เครื่องคอมพิวเตอร์แม่ข่ายก่อนเสมอ เนื่องจากโปรแกรม Web_Control ทำงานบน Web Brower ดังนั้นจึงจำเป็นต้องรัน โปรแกรม Server_www เพื่อให้โปรแกรม Server_www ทำงานเป็น Web server เมื่อรันโปรแกรม Server_www ที่เครื่องคอมพิวเตอร์แม่ข่ายแล้ว ก็สามารถใช้โปรแกรม Web_Control ได้เหมือนกับการเข้า Web Page ทั่ว ๆ ไป ซึ่งในโครงงานนี้ได้กำหนด URL ของโปรแกรม Web_Control เป็น http://www.ed03.inded.kmitl.ac.th/Web_Control ปุ่มปรับและการแสดงผลต่าง ๆ คล้ายกับโปรแกรม Mix_work แต่ปุ่มที่แตกต่างไปบางปุ่ม ดังนี้

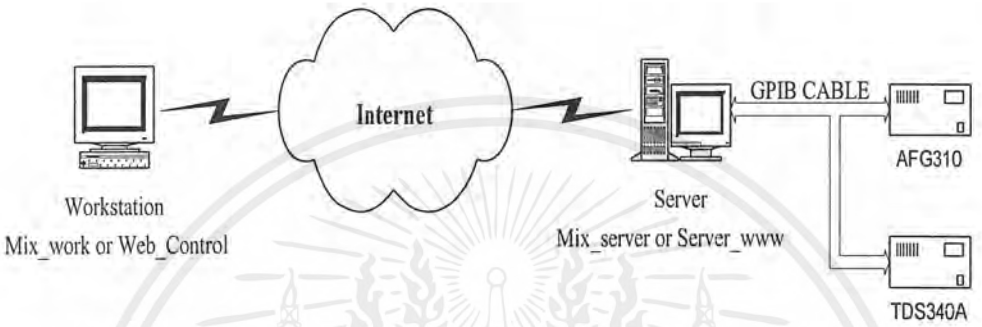
Connect (AutoUpdate) ระบุการเชื่อมต่อโพรโตคอล dstp ให้มีการเปลี่ยนแปลงข้อมูลเองได้โดยอัตโนมัติ

Connect (ManualUpdate) เช่นเดียวกับปุ่ม Connect (AutoUpdate) แต่จะเปลี่ยนแปลงข้อมูลเมื่อมีการคลิกที่ปุ่ม Update เท่านั้น

Update ใช้ร่วมกับปุ่ม Connect (ManualUpdate)

Disconnect คลิกเมื่อต้องการยกเลิกการเชื่อมต่อโพรโตคอล Dstp

4.1.4 ภาพรวมลักษณะการใช้งานโปรแกรม Mix_server และ Mix_work หรือ โปรแกรม Server_www และ Web_control



รูปที่ 4.5 ภาพรวมลักษณะการใช้โปรแกรม Mix_server และ Mix_work หรือโปรแกรม Server_www และ Web_control

4.2 ลักษณะอุปกรณ์และการติดตั้ง

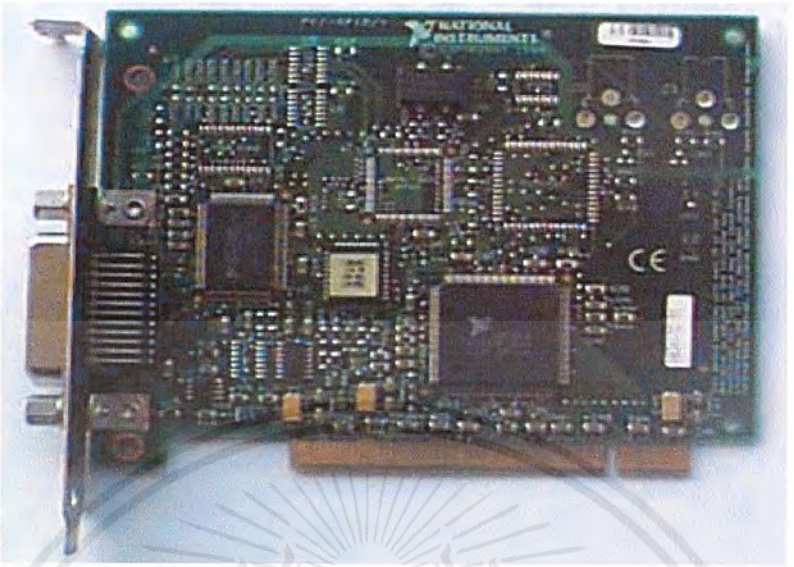
อุปกรณ์ที่ใช้ในโครงงานนี้มีอยู่ 3 ชนิดคือ

1. การ์ด GPIB
2. เครื่องวัดสัญญาณ TDS340A และเครื่องกำเนิดสัญญาณ AFG310
3. เครื่องคอมพิวเตอร์

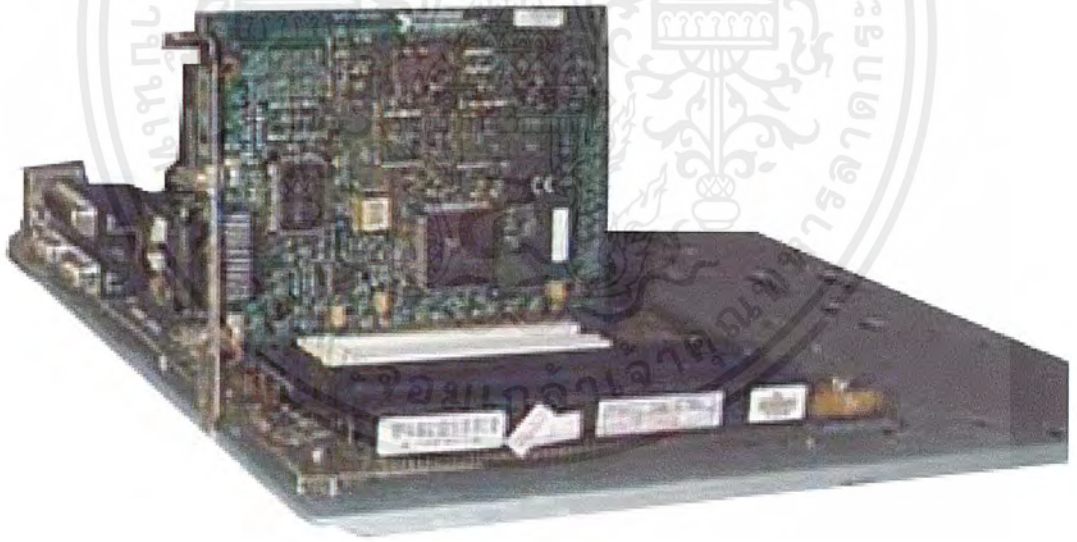
ลักษณะอุปกรณ์ต่าง ๆ และวิธีการติดตั้งอุปกรณ์ แสดงให้เห็นได้ดังนี้

4.2.1. ลักษณะของการ์ด GPIB

ลักษณะของการ์ด GPIB และการติดตั้งการ์ด GPIB บน Main Board ของเครื่องคอมพิวเตอร์แสดงให้เห็นได้ดังรูปที่ 4.6 และ รูปที่ 4.7 ตามลำดับ ดังนี้



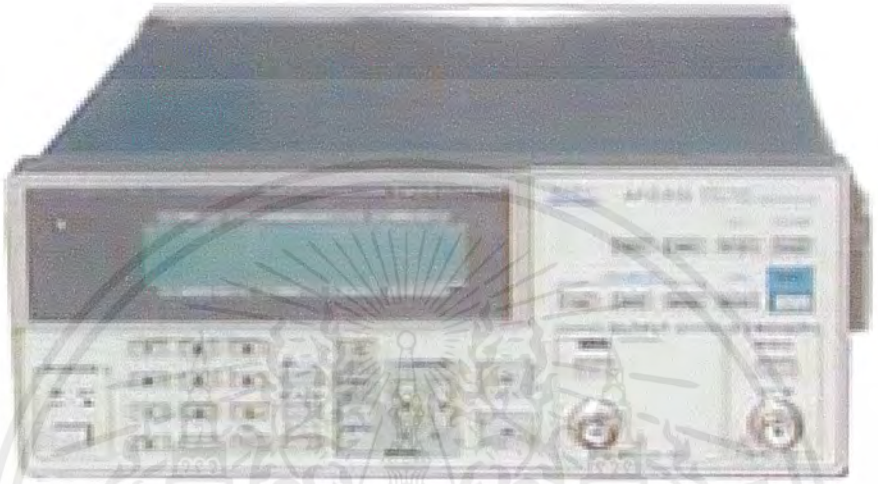
รูปที่ 4.6 ลักษณะของการ์ด GPIB



รูปที่ 4.7 การติดตั้งการ์ด GPIB บน Main Board ของเครื่องคอมพิวเตอร์

4.2.2. ลักษณะของเครื่องกำเนิดสัญญาณ AFG310 และเครื่องวัดสัญญาณ TDS340A

ลักษณะของเครื่องกำเนิดสัญญาณ AFG310 เครื่องวัดสัญญาณ TDS340A สามารถแสดงเห็นได้ดังรูปที่ 4.8 และ รูปที่ 4.9 ตามลำดับ ดังนี้



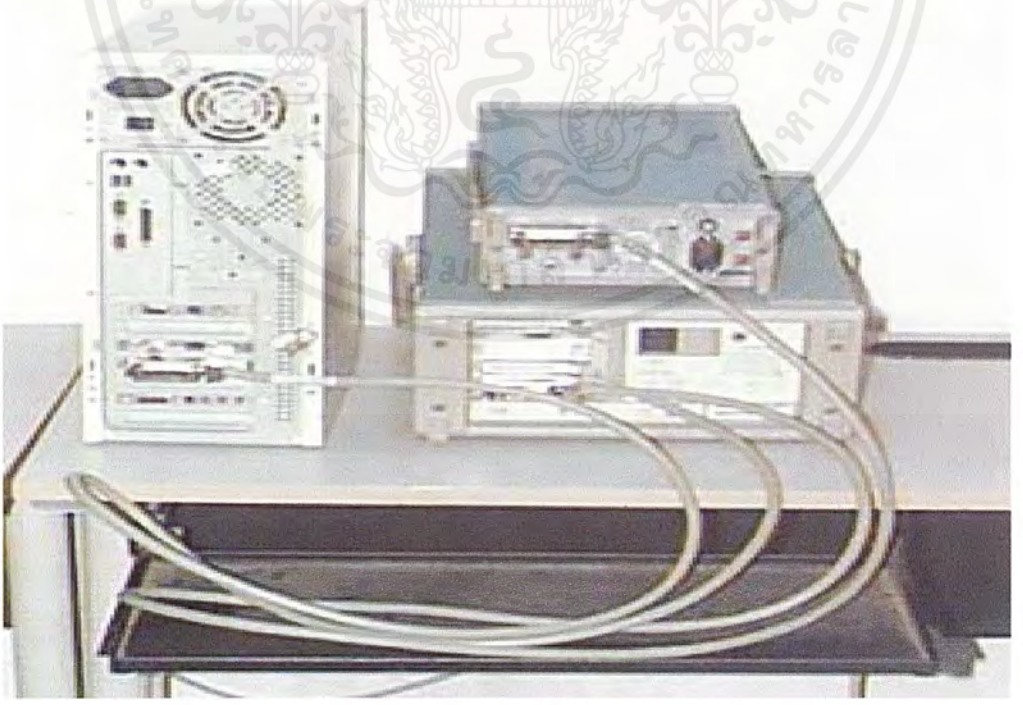
รูปที่ 4.8 ลักษณะของเครื่องกำเนิดสัญญาณ AFG310

รูปที่ 4.9 ลักษณะของเครื่องวัดสัญญาณ TDS340A

4.2.3. ลักษณะการเชื่อมต่อเครื่องมือวัดเข้ากับเครื่องคอมพิวเตอร์

ลักษณะการเชื่อมต่อเครื่องมือวัดเข้ากับเครื่องคอมพิวเตอร์ แสดงให้เห็นได้ดังรูปที่ 4.10

รูปที่ 4.10(ก) ลักษณะการเชื่อมต่อเครื่องมือวัดเข้ากับเครื่องคอมพิวเตอร์



รูปที่ 4.10(ข) ลักษณะการเชื่อมต่อเครื่องมือวัดเข้ากับเครื่องคอมพิวเตอร์

4.3. การทดลองและผลการทดลอง

เพื่อให้สอดคล้องกับโปรแกรมต่าง ๆ ที่ได้กล่าวไว้ในตอนต้น โครงงานนี้ได้แบ่งการทดลองและผลการทดลองของโครงงานออกเป็น 3 ส่วน โดยมีรายละเอียดดังต่อไปนี้

4.3.1. การทดลองและผลการทดลองโปรแกรม Local_Control

ขั้นที่ 1 เปิดเครื่องกำเนิดสัญญาณ AFG310 และเครื่องวัดสัญญาณ TDS340A บันทึกค่าต่าง ๆ ของเครื่องมือวัดทั้งสอง ดังตารางที่ 4.1

ตารางที่ 4.1 ผลการทดลองขั้นที่ 1

เครื่อง	Frequency	Amplitude	Offset	Phase	Wave from	Bust count	Mode
TDS340A	-	-	-	-	-	-	-
AFG310	100KHz	1V	0V	0	SINE	OFF	CONT

ขั้นที่ 2 ต่อสัญญาณเอาต์พุตของเครื่องกำเนิดสัญญาณ AFG310 เข้าเครื่องวัดสัญญาณ TDS340A จากนั้นกดปุ่ม Output ของเครื่องกำเนิดสัญญาณ AFG310 บันทึกผลการทดลอง ดังตารางที่ 4.2

ตารางที่ 4.2 ผลการทดลองขั้นที่ 2

เครื่อง	Frequency	Amplitude	Offset	Phase	Wave from	Bust count	Mode
TDS340A	100.1KHz	2.1V	-	-	-	-	-
AFG310	100KHz	1V	0V	0	SINE	OFF	CONT

ขั้นที่ 3 รันโปรแกรม Local_Control จากนั้นปรับปุ่มต่าง ๆ ภายในโปรแกรมLocal_Control ดังนี้

ขณะเลือก TDS340A

- กดปุ่ม RESET บันทึกผลการทดทดลอง ดังตารางที่ 4.3

ตารางที่ 4.3 ผลการทดลองขั้นที่ 3 ขณะกดปุ่ม RESET

เครื่อง	Frequency	Amplitude	Offset	Phase	Wave from	Bust count	Mode
TDS340A	25KHz	200mV	-	-	-	-	-
AFG310	100KHz	1V	0V	0	SINE	OFF	CONT
Local_ Control	25KHz	200mV	-	-	-	-	-

- กดปุ่ม AUTOSET บันทึกผลการทดลอง ดังตารางที่ 4.4

ตารางที่ 4.4 ผลการทดลองขั้นที่ 3 ขณะกดปุ่ม AUTOSET

เครื่อง	Frequency	Amplitude	Offset	Phase	Wave from	Bust count	Mode
TDS340A	100.2KHz	2.08V	-	-	SINE	-	-
AFG310	100KHz	1V	0V	0	SINE	OFF	CONT
Local_ Control	100.2KHz	2.08V	-	-	SINE	-	-

- เลือกช่องวัดสัญญาณที่ 2 (CH2) บันทึกผลการทดลอง ดังตารางที่ 4.5

ตารางที่ 4.5 ผลการทดลองขั้นที่ 3 ขณะเลือก CH2

เครื่อง	Frequency	Amplitude	Offset	Phase	Wave from	Bust count	Mode
TDS340A	อ่านค่าไม่ได้	อ่านค่าไม่ได้	-	-	-	-	-
AFG310	100KHz	1V	0V	0	SINE	OFF	CONT
Local_ Control	อ่านค่าไม่ได้	อ่านค่าไม่ได้	-	-	-	-	-

- ปรับปุ่ม Main scaling เป็น 1Us/DIV บันทึกผลการทดลอง ดังตารางที่ 4.6

ตารางที่ 4.6 ผลการทดลองขั้นที่ 3 ขณะเลือก Main scaling เป็น 1us/DIV

เครื่อง	Frequency	Amplitude	Offset	Phase	Wave from	Bust count	Mode
TDS340A	100.2KHz	2.08V	-	-	SINE	-	-
AFG310	100KHz	1V	0V	0	SINE	OFF	CONT
Local_ Control	99.8KHz	2.12V	-	-	SINE	-	-

- ปรับปุ่ม Volt/DIV เป็น 5 บันทึกผลการทดลอง ดังตารางที่ 4.7

ตารางที่ 4.7 ผลการทดลองขั้นที่ 3 ขณะเลือก Volt /DIV เป็น 5

เครื่อง	Frequency	Amplitude	Offset	Phase	Wave from	Bust count	Mode
TDS340A	100.2KHz	2.08V	-	-	-	-	-
AFG310	100KHz	1V	0V	0	SINE	OFF	CONT
Local_ Control	100.2KHz	2.08V	-	-	SINE	-	-

หมายเหตุ ที่เครื่องวัดสัญญาณ TDS340A ไม่สามารถแสดงรูปสัญญาณได้เนื่องจากสัญญาณมีแอมปลิจูดเล็กเกินไป

ขณะเลือก AFG310

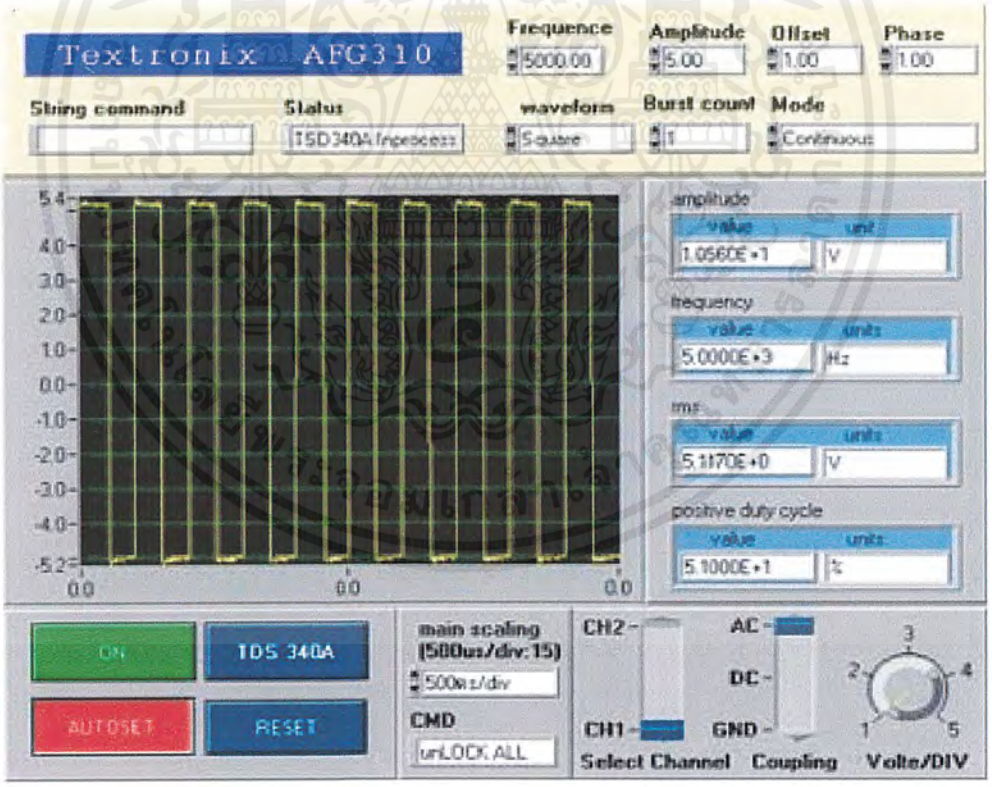
- ปรับ Frequency เป็น 5000Hz
- ปรับ Amplitude เป็น 5 V
- ปรับ Offset เป็น 1
- ปรับ Phase เป็น 1
- ปรับ Waveform เป็น square
- ปรับ Burst count เป็น 2
- ปรับ Mode เป็น Continuous

อ่านค่าจากโปรแกรม Local_Control บันทึกผลการทดลอง ดังตารางที่ 4.8

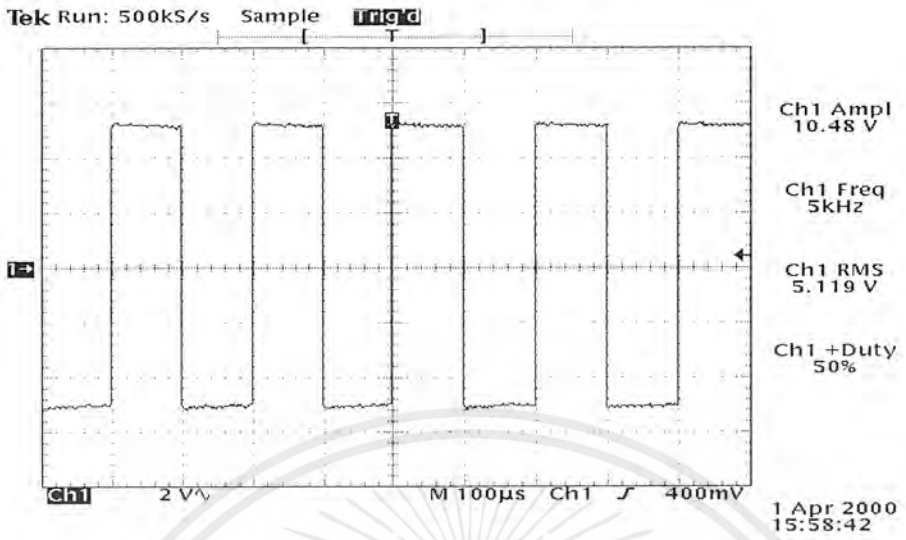
ตารางที่ 4.8 ผลการทดลองขั้นที่ 3 ขณะเลือก AFG310 หลังจากปรับค่าต่าง ๆ แล้ว

เครื่อง	Frequency	Amplitude	Offset	Phase	Wave from	Bust count	Mode
TDS340A	5KHz	10.6V	-	-	SQUARE	-	-
AFG310	5KHz	5V	1V	1	SQUA	OFF	CONT
Local_ Control	5KHz	10.6V	-	-	SQUARE	-	-

รูปเปรียบเทียบผลการทดลองที่ 4.3.1 ระหว่างเครื่องคอมพิวเตอร์ที่ รันโปรแกรม Local_Control กับเครื่องวัดสัญญาณ TDS 340A แสดงให้เห็นได้ ดังรูปที่ 4.11 และรูปที่ 4.12 ตามลำดับ



รูปที่ 4.11 โปรแกรม Local_Control หลังขั้นการทดลองที่ 3



รูปที่ 4.12 ภาพเครื่องวัดสัญญาณ TDS340A หลังขั้นการทดลองที่ 3

4.3.2. การทดลองและผลการทดลองโปรแกรม Mix_server และ Mix_work

ขั้นที่ 1 รันโปรแกรม Mix_server ที่เครื่องคอมพิวเตอร์แม่ข่าย

ขั้นที่ 2 รันโปรแกรม Mix_work ที่เครื่องคอมพิวเตอร์ลูกข่าย

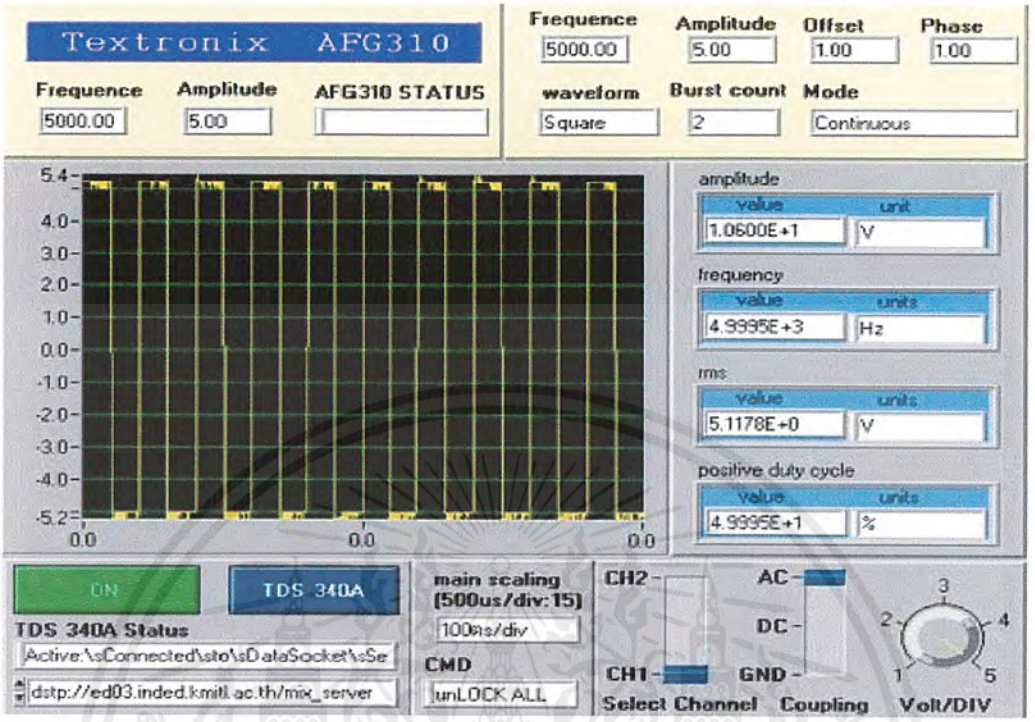
ขั้นที่ 3 Volt/DIV เป็น 5 และ main scaling เป็น 100us/div

ขั้นที่ 4 ทดลองปรับค่าตามการทดลองที่ 4.3.1. ขั้นที่ 3 ขณะเลือก AFG310 บันทึกผลการทดลอง ดังตารางที่ 4.9

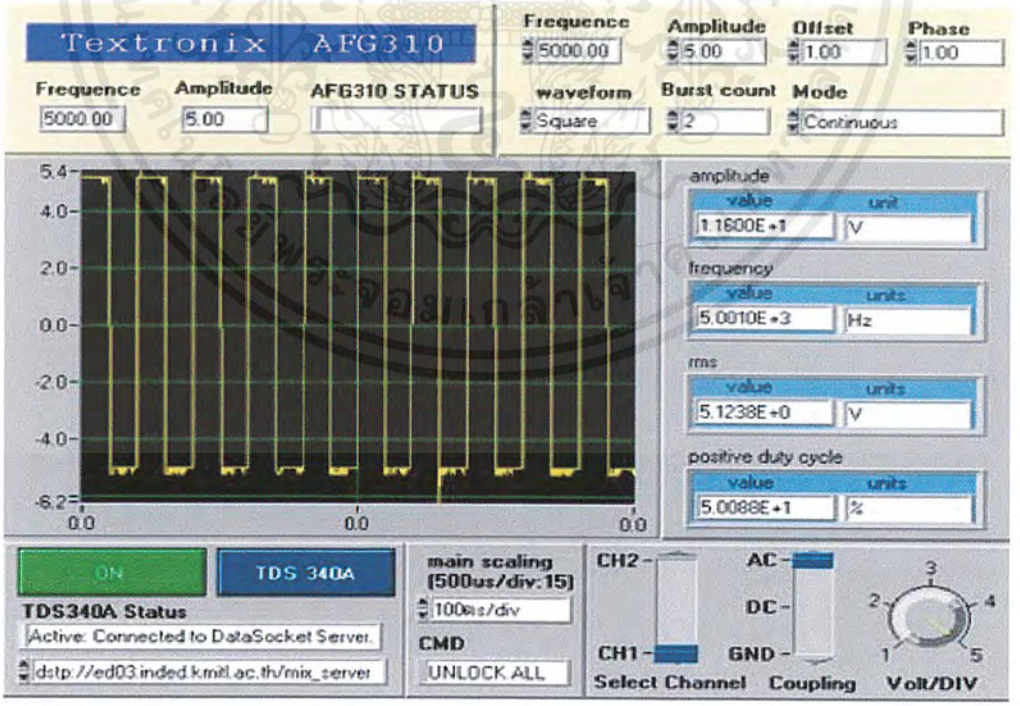
ตารางที่ 4.9 ผลการที่ 4.3.2 ขั้นตอนการทดลองที่ 3

เครื่อง	Frequency	Amplitude	Offset	Phase	Wave from	Bust count	Mode
TDS340A	5KHz	10.56V	-	-	-	-	-
AFG310	5KHz	5V	1V	1	SQUA	OFF	CONT
Mix_server	5000Hz	10V	1	1	SQUARE	2	CONT
Mix_work	5000Hz	10V	1	1	SQUARE	2	CONT

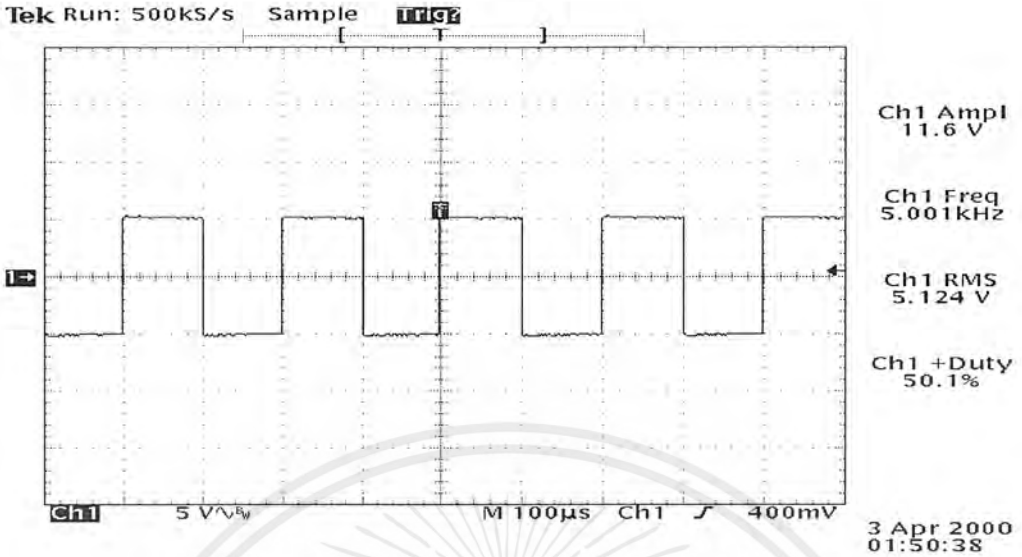
รูปเปรียบเทียบผลการทดลองที่ 4.3.2. ระหว่างเครื่องคอมพิวเตอร์แม่ข่ายและเครื่องคอมพิวเตอร์ลูกข่ายแสดง และ ภาพที่อ่านได้จากเครื่องวัดสัญญาณ TDS340A ให้เห็นได้ดังรูปที่ 4.13, รูปที่ 4.14 และ รูปที่ 4.15 ตามลำดับ



รูปที่ 4.13 ภาพของโปรแกรม Mix_server หลังจากทำการทดลองขั้นที่ 3



รูปที่ 4.14 ภาพของโปรแกรม Mix_work หลังจากทำการทดลองขั้นที่ 3



รูปที่ 4.15 ภาพของเครื่องวัดสัญญาณ TDS340A หลังจากทำการทดลองขั้นที่ 3

4.3.3. การทดลองและผลการทดลองโปรแกรม Server_www และ Web_control

ขั้นที่ 1 RUN โปรแกรม Server_www ที่เครื่องคอมพิวเตอร์แม่ข่าย

ขั้นที่ 2 ที่เครื่องคอมพิวเตอร์ลูกข่าย เข้าสู่ Web Page ที่ URL ดังนี้

http://www.ed03.inded.kmitl.ac.th/web_control

ขั้นที่ 3 ทดลองปรับค่าตามการทดลองที่ 4.3.1.ขั้นที่ ขณะเลือก AFG310 โดยมีข้อแตกต่างคือ

ปรับ Phase เป็น 1, ปรับ Waveform เป็น 1, ปรับ Burst count เป็น 2

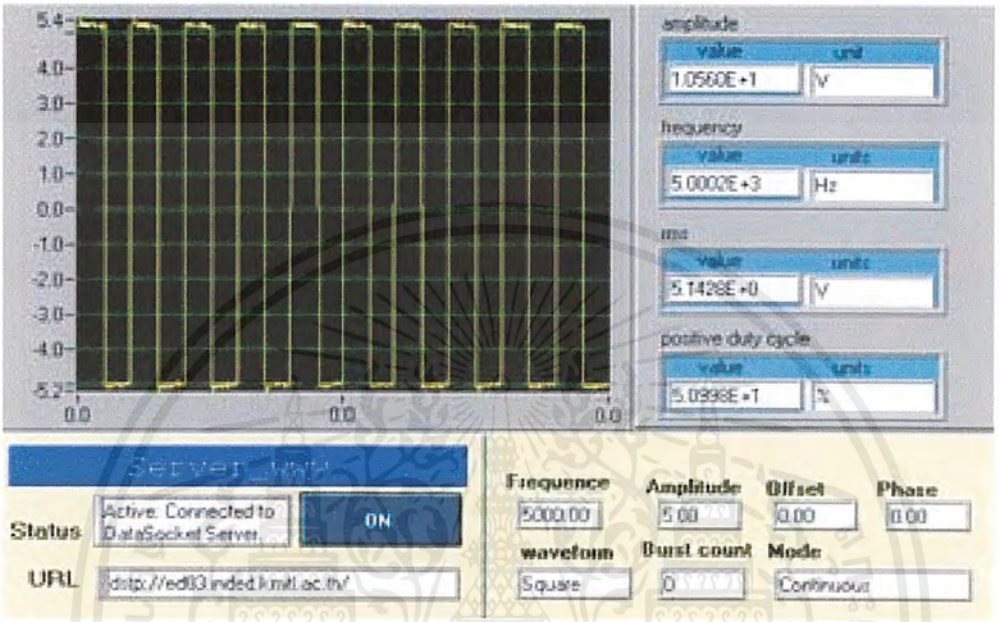
และ ปรับ Mode เป็น 0

บันทึกผลการทดลอง ดังตารางที่ 4.10

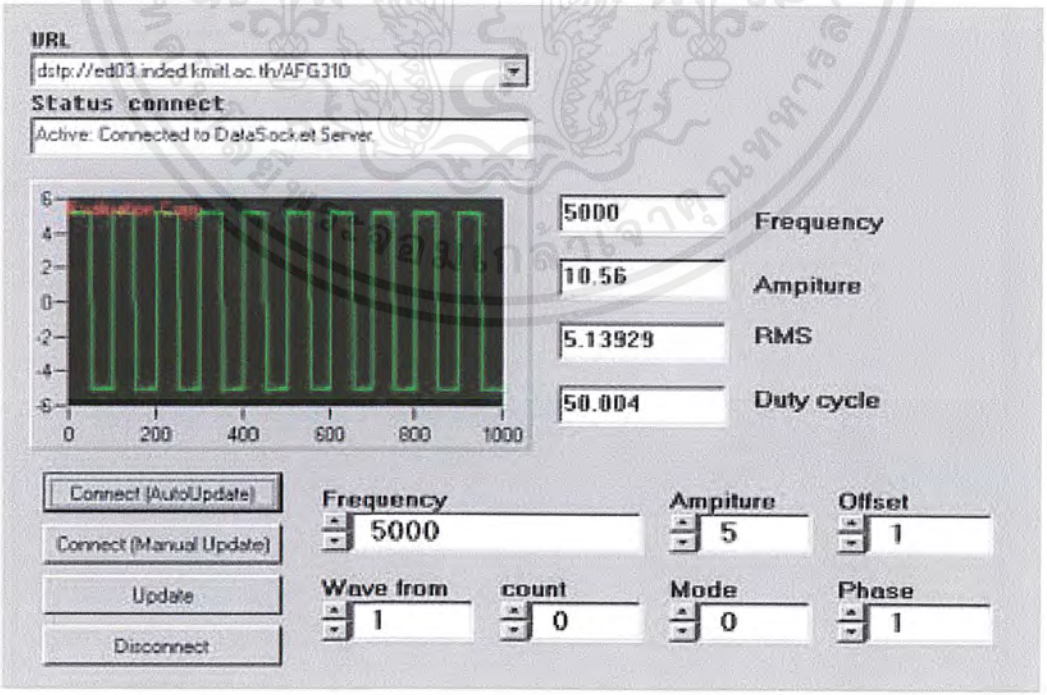
ตารางที่ 4.10 ผลการที่ 4.3.3 ขั้นตอนการทดลองที่ 3

เครื่อง	Frequency	Amplitude	Offset	Phase	Wave from	Bust count	Mode
TDS340A	5KHz	10.56V	-	-	-	-	-
AFG310	5KHz	5V	1V	1	SQUA	OFF	CONT
Server_www	5000Hz	10V	1	1	Square	2	CONT
Web_control	5000Hz	5V	1	1	Square	2	CONT

รูปเปรียบเทียบผลการทดลองที่ 4.3.3 ระหว่างเครื่องคอมพิวเตอร์แม่ข่ายและเครื่องคอมพิวเตอร์ลูกข่ายแสดงให้เห็นได้ ดังรูปที่ 4.16 และรูปที่ 4.17 ตามลำดับ โดยภาพของเครื่องวัดสัญญาณ TDS340A ที่ได้จากการทดลองนี้เหมือนกับภาพที่ 4.15



รูปที่ 4.16 ภาพของโปรแกรม Server_www หลังจากทำการทดลองขั้นที่ 3



รูปที่ 4.17 ภาพของโปรแกรม Web_control หลังจากทำการทดลองขั้นที่ 3

บทที่ 5
สรุปปัญหา แนวทางแก้ไขปัญหาและการพัฒนา

5.1. บทสรุป

ปริญญานิพนธ์ การควบคุมเครื่องมือวัดผ่านเครือข่ายอินเตอร์เน็ตโดย GPIB พอร์ตนี้ เป็นการนำเอาเทคโนโลยีหลายอย่างมารวมกันเช่น ActiveX และ LabVIEW เพื่อให้สามารถควบคุมเครื่องมือวัดโดยไม่ผ่านและผ่านเครือข่ายอินเตอร์เน็ตได้

การควบคุมเครื่องมือวัดโดยผ่านพอร์ต GPIB นั้นเป็นที่นิยมมากในโรงงานอุตสาหกรรม แต่ปริญญานิพนธ์นี้ เป็นการควบคุมเครื่องมือวัด 2 ชนิดแค่นั้น นั้นคือ เครื่องวัดสัญญาณ TDS340A และเครื่องกำเนิดสัญญาณ AFG310 ทำให้เหมือนว่าปริญญานิพนธ์นี้ไม่สามารถนำไปใช้ประโยชน์ได้จริง แต่ปริญญานิพนธ์นี้ได้เน้นหนักไปที่กรรมวิธีในควบคุมเครื่องมือวัดโดยผ่านพอร์ต GPIB และกรรมวิธีในการควบคุมเครื่องมือวัดผ่านเครือข่ายอินเตอร์เน็ต มากกว่า ดังนั้นปริญญานิพนธ์นี้จึงเหมาะที่จะเป็นแนวทางในการพัฒนาโปรแกรมเพื่อควบคุมเครื่องมือวัด มากกว่าการที่จะนำปริญญานิพนธ์นี้ไปใช้งานจริง

ขีดความสามารถของปริญญานิพนธ์นี้

1. สามารถควบคุมเครื่องมือวัดโดยไม่ผ่านเครือข่ายอินเตอร์เน็ตได้จากโปรแกรม Local_Contorl
2. สามารถควบคุมเครื่องมือวัดโดยผ่านเครือข่ายอินเตอร์เน็ตได้ โดยเทคโนโลยี Data Socket ของโปรแกรม LabVIEW จากโปรแกรม Mix_server และ Mix_work
3. สามารถควบคุมเครื่องมือวัดผ่านเครือข่ายอินเตอร์เน็ตได้ โดยเทคโนโลยี ActiveX จากโปรแกรม Server_www และ Web_Control

5.2. ปัญหา และแนวทางแก้ไขปัญหา

ในการทำปริญญานิพนธ์นี้ ได้มีปัญหาเกิดขึ้นขณะดำเนินงาน ซึ่งเป็นปัญหาที่สามารถแก้ไขได้และปัญหาที่ไม่สามารถแก้ไขได้ ดังนั้นคณะผู้จัดทำจึงได้แบ่งปัญหาและแนวทางในการแก้ไขปัญหาออกเป็น 2 ส่วน ดังนี้

5.2.1. ปัญหาและการแก้ไขปัญหาที่สามารถแก้ไขได้

ปัญหาที่ 1 เนื่องจากปริญญานิพนธ์นี้เป็นเรื่องที่ค่อนข้างใหม่ และข้อมูลที่จำเป็นส่วนใหญ่เป็นข้อมูลทางเทคนิคที่ปกปิดไว้เพื่อผลประโยชน์ทางการค้า ทำให้ข้อมูลในการทำปริญญานิพนธ์น้อยมาก

การแก้ไขปัญหา คณะผู้จัดทำใช้อยู่หลายวิธีเช่น สอบถามจากผู้รู้, สืบค้นข้อมูลในอินเตอร์เน็ตและติดต่อขอข้อมูลทางเทคนิคจากบริษัทที่ทำงานเกี่ยวกับเครื่องมือวัดโดยตรง

ปัญหาที่ 2 ข้อมูลที่ได้มาส่วนใหญ่เป็นภาษาอังกฤษ ทำให้ยากในการแปลความหมาย

การแก้ไขปัญหา คณะผู้จัดทำแก้ปัญหาโดย ทดลองเขียนโปรแกรมตามตัวอย่างโปรแกรม

ปัญหาที่ 3 เครื่องมือและอุปกรณ์ที่ใช้ในการทำปริญญานิพนธ์มีราคาแพง

การแก้ไขปัญหา คณะผู้จัดทำแก้ปัญหาโดย ขอความอนุเคราะห์จากอาจารย์ที่ปรึกษา

ปัญหาที่ 4 จุดประสงค์ในการทำปริญญานิพนธ์ มีข้อหนึ่งที่ต้องการศึกษาการเขียนโปรแกรม CGI ด้วยแต่เนื่องจากการรวมระหว่าง LabVIEW และ CGI จำเป็นต้องมีโปรแกรม Internet ToolKit ของ LabVIEW ซึ่งเป็นโปแกรมที่มีราคาแพงไม่สามารถจัดซื้อได้

การแก้ไขปัญหา คณะผู้จัดทำแก้ปัญหาโดย เปลี่ยนมาใช้เป็น ActiveX แทน CGI ซึ่งมีตัวอย่างของ ActiveX ให้สามารถ Down load จากอินเตอร์เน็ตได้

5.2.2. ปัญหาและการแก้ไขปัญหาที่ไม่สามารถแก้ไขได้

ปัญหาที่ 1 เนื่องจากการควบคุมเครื่องมือวัดผ่านพอร์ต GPIB โดยใช้โปรแกรม LabVIEW นั้นมีความซับซ้อนมากทำให้กาทำงานของโปรแกรมช้า ไม่สามารถทำให้โปรแกรมทำงานแบบเวลาจริง(Real time) ได้

แนวทางการแก้ไขปัญหา เขียนโปรแกรมจากภาษาโปรแกรมอื่น ๆ ที่มีความเร็วในการทำงานมากกว่า LabVIEW เช่น ภาษาโปรแกรม C, Pascal และ Visual C++

ปัญหาที่ 2 เนื่องจากข้อจำกัดทางโครงสร้างของเครื่องมือวัดทำให้การเขียนโปรแกรมควบคุมเครื่องมือวัดไม่เหมาะที่จะทำงานแบบตอบสนองตลอดเวลา (Inter Active)

แนวทางการแก้ไขปัญหา เปลี่ยนจากการเขียนโปรแกรมแบบตอบสนองตลอดเวลา เป็นแบบตอบสนองเมื่อต้องการ เช่นตัวอย่างโปรแกรม Wave Start ของ Tektronix

ปัญหาที่ 3 เนื่องจาก ActiveX ที่ใช้ในปริญญานิพนธ์นี้เป็นActiveX ตัวอย่าง ทำให้สามารถใช้งานได้แค่ 5 นาที

แนวทางการแก้ไขปัญหา สร้าง ActiveX ขึ้นมาใช้เอง อาจจะใช้ภาษาโปรแกรม Visual C++ สร้างขึ้นมาก็ได้

ปัญหาที่ 4 การ์ด GPIB มีราคาแพง

แนวทางการแก้ไขปัญหา เปลี่ยนการควบคุมเครื่องมือวัดผ่านพอร์ต GPIB เป็นผ่านพอร์ตอนุกรมแทนได้

5.3. การพัฒนา

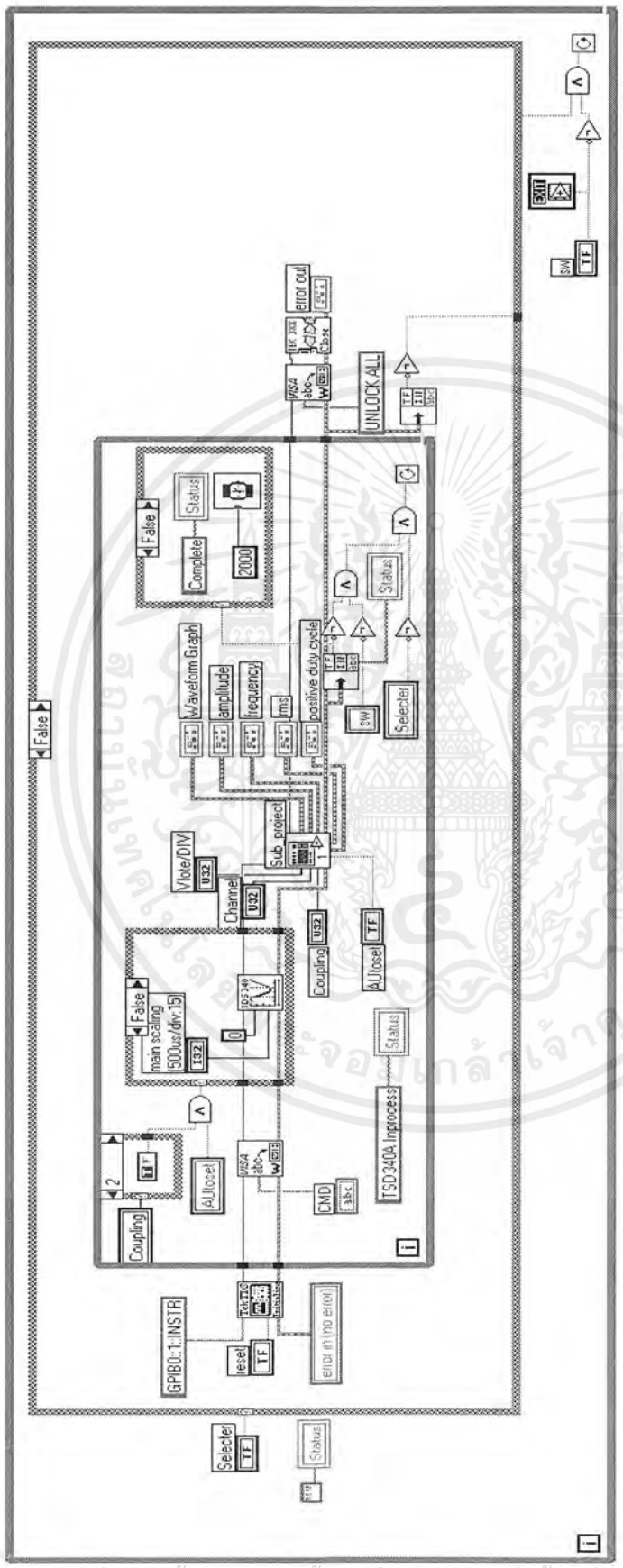
จากที่ปริญญานิพนธ์นี้เน้นไปที่กรรมวิธีในการควบคุมเครื่องมือวัด มากกว่าการมากที่จะให้ปริญญานิพนธ์นี้สามารถนำไปใช้งานจริงได้ ดังนี้การที่จะพัฒนาปริญญานิพนธ์นี้ต่อไปจึงสามารถทำได้หลายอย่าง ดังนี้

1. ติดตั้งการ์ด DAQ เพิ่ม และเขียนโปรแกรมควบคุมโดยผ่านและไม่ผ่านเครือข่ายอินเตอร์เน็ต เพื่อประยุกต์ใช้โรงงานอุสาหกรรมจริง

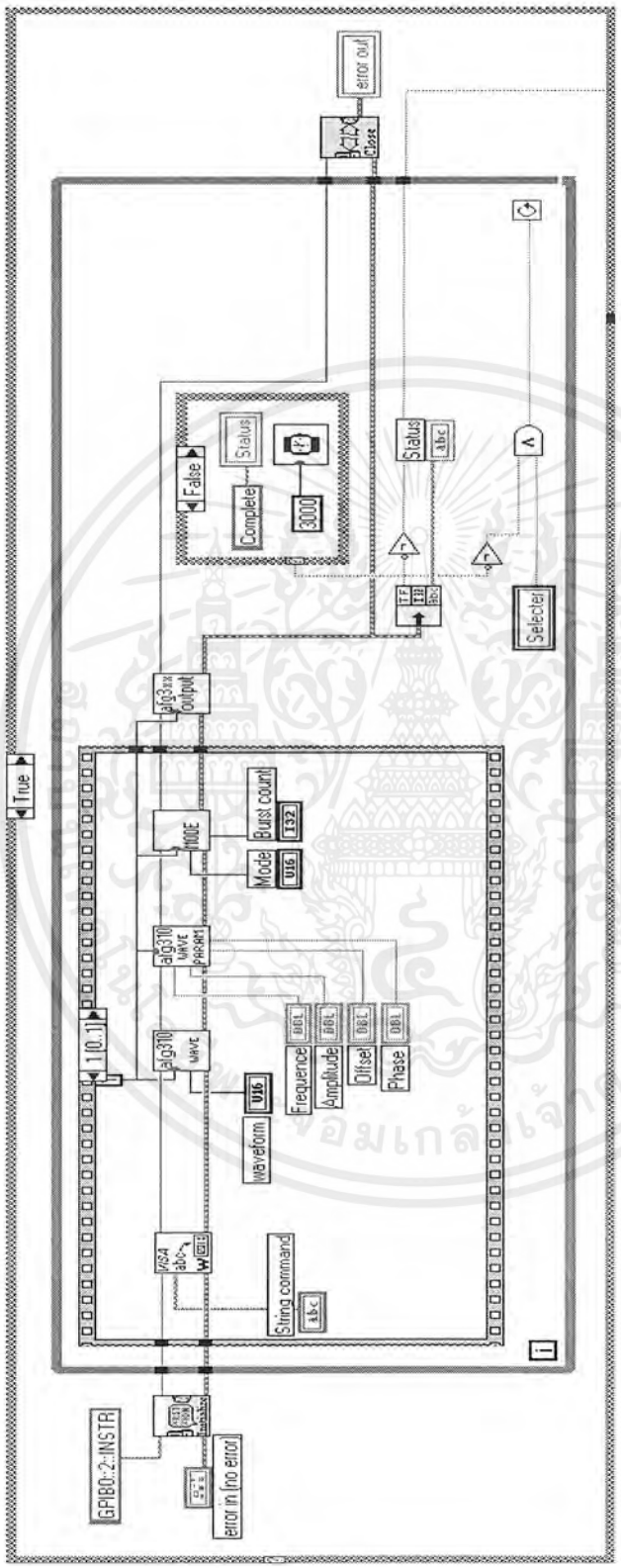
2. เขียนโปรแกรมควบคุมเครื่องมือวัดผ่านเครือข่ายอินเตอร์เน็ตโดยไม่ต้องมีการเขียนข้อมูลของเครื่องคอมพิวเตอร์ลูกข่ายกลับไปยังเครื่องคอมมพิวเตอร์แม่ข่าย ทำให้สามารถดูค่าต่าง ๆ ของเครื่องมือวัดผ่านเครือข่ายอินเตอร์เน็ตได้หลายเครื่องลูกข่ายพร้อมกัน เพื่อเป็นสื่อในการเรียนการสอน ทำเขียนโปรแกรมควบคุมเครื่องมือวัดผ่านเครือข่ายอินเตอร์เน็ตโดยไม่ต้องมีการเขียนข้อมูล

ภาคผนวก ก.

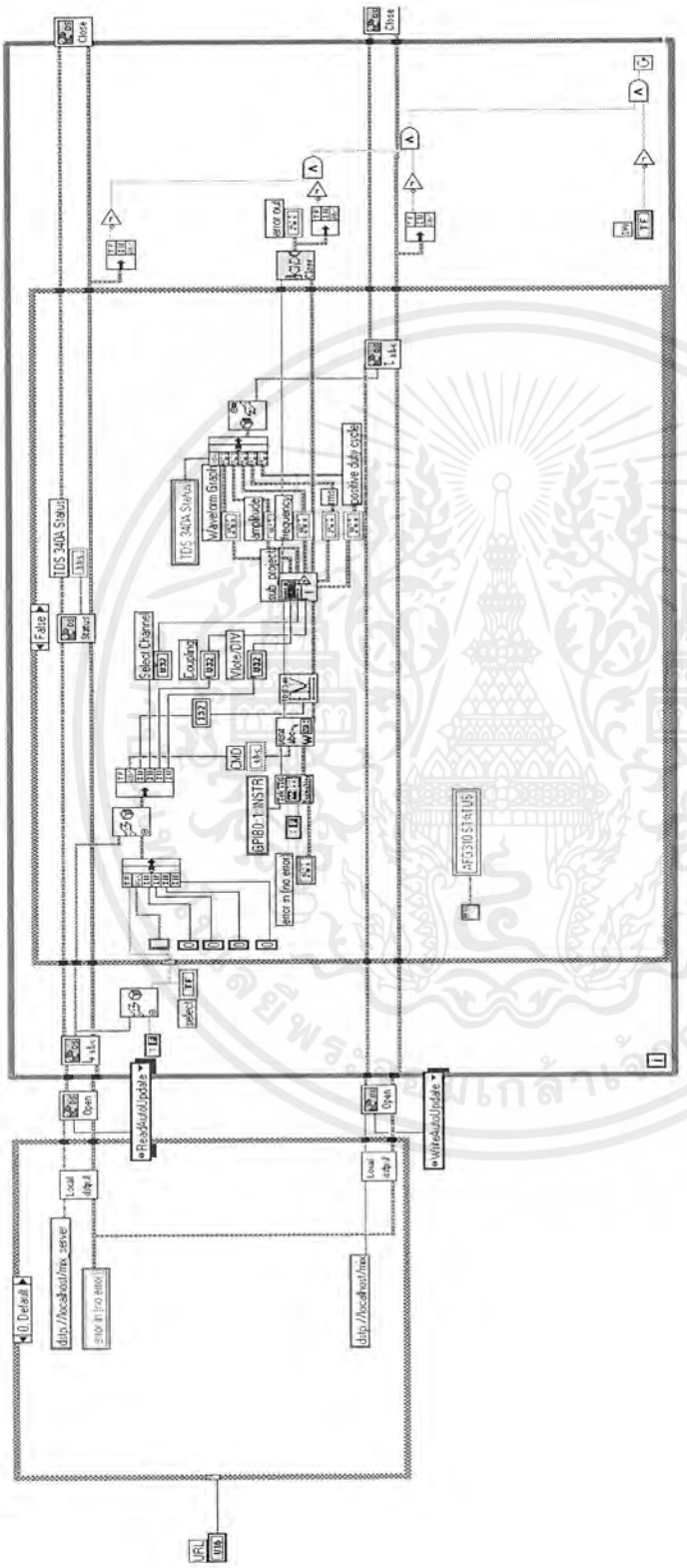
Block Diagram และ Source Code ของโปรแกรม



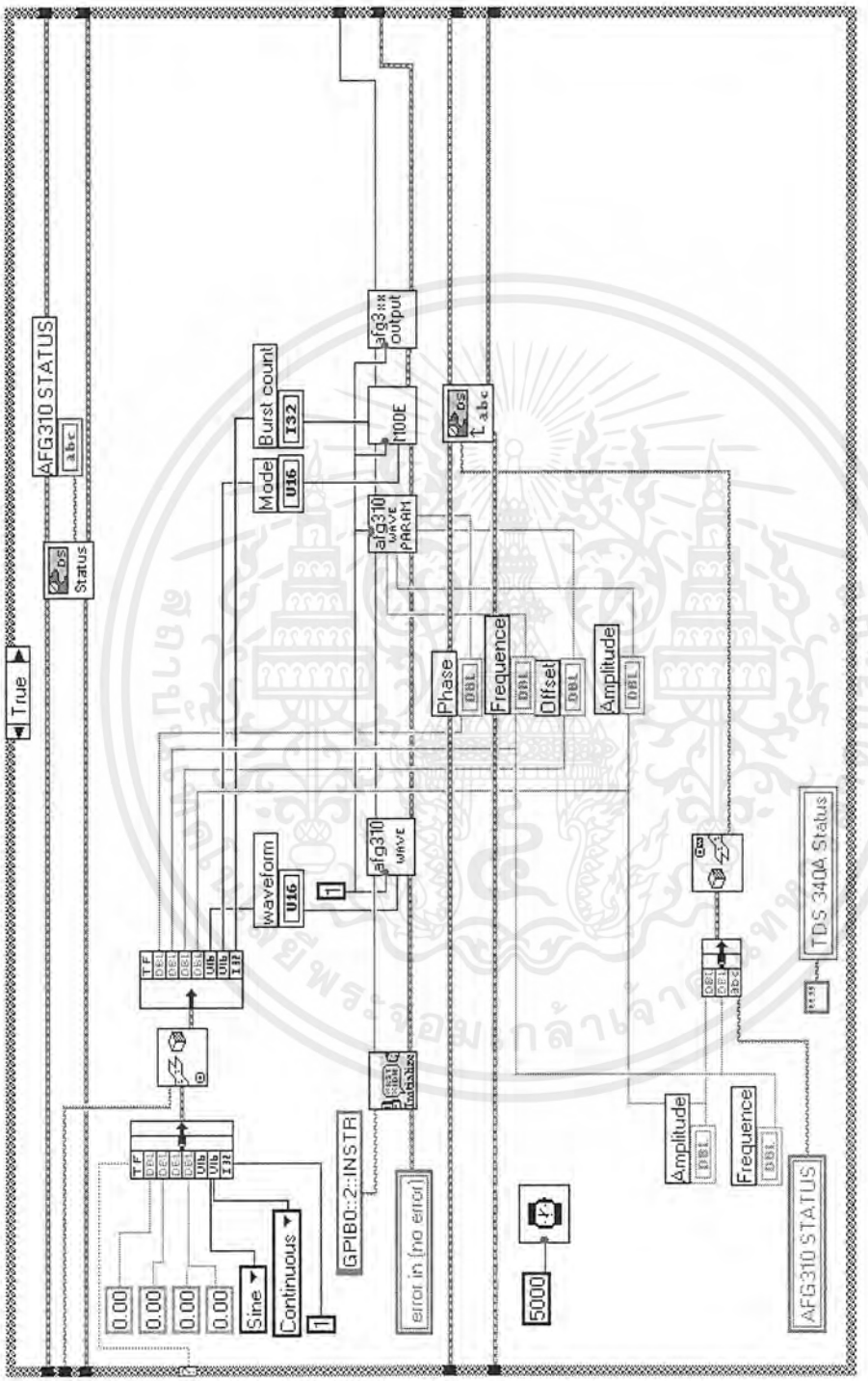
รูปที่ ก.1 Block Diagram ของโปรแกรม Local_control ขณะเลือกโปรแกรมย่อย TDS340A



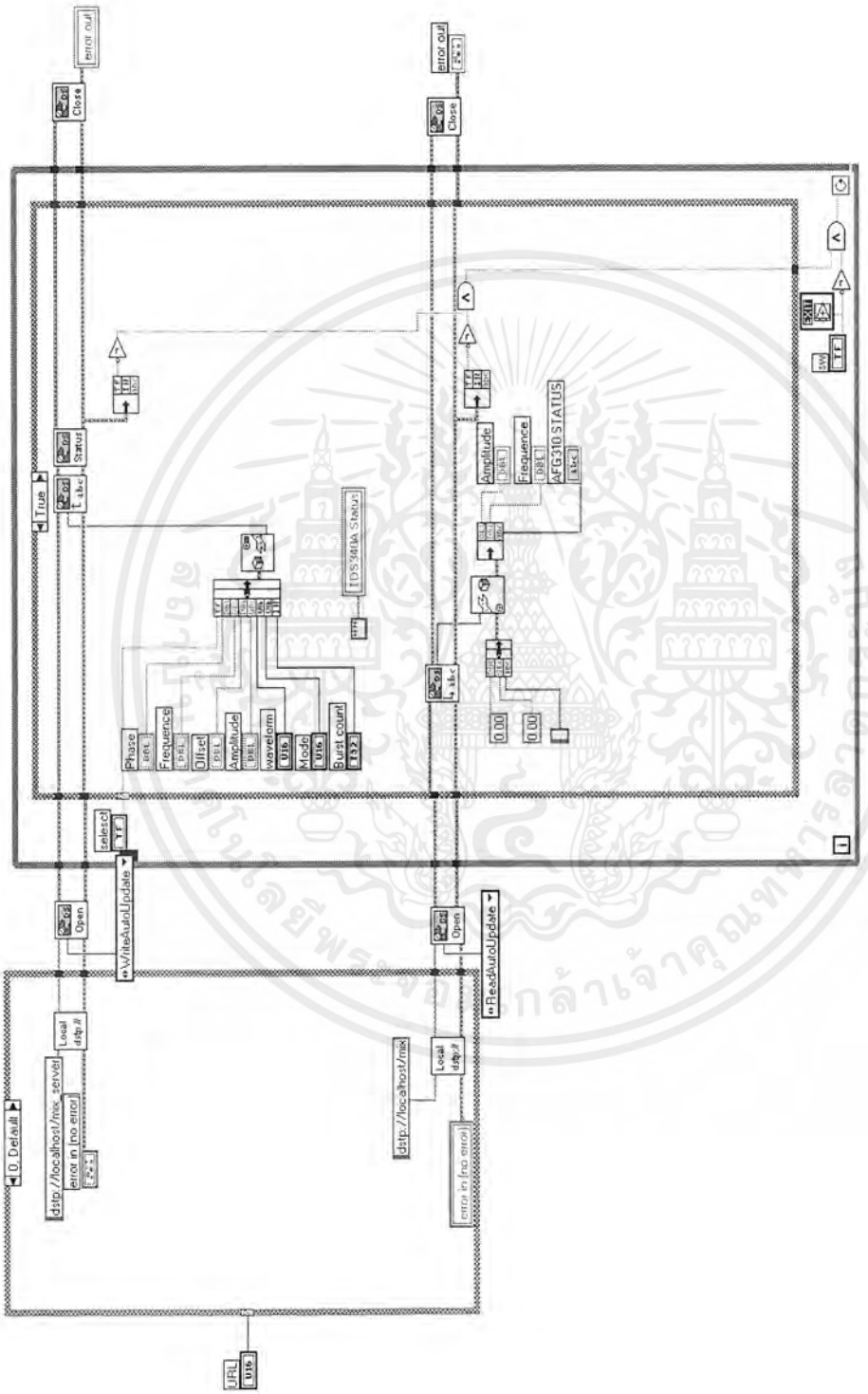
รูปที่ ก.2 Block Diagram ของโปรแกรม local_control ขณะเลือกโปรแกรมย่อย AFG310



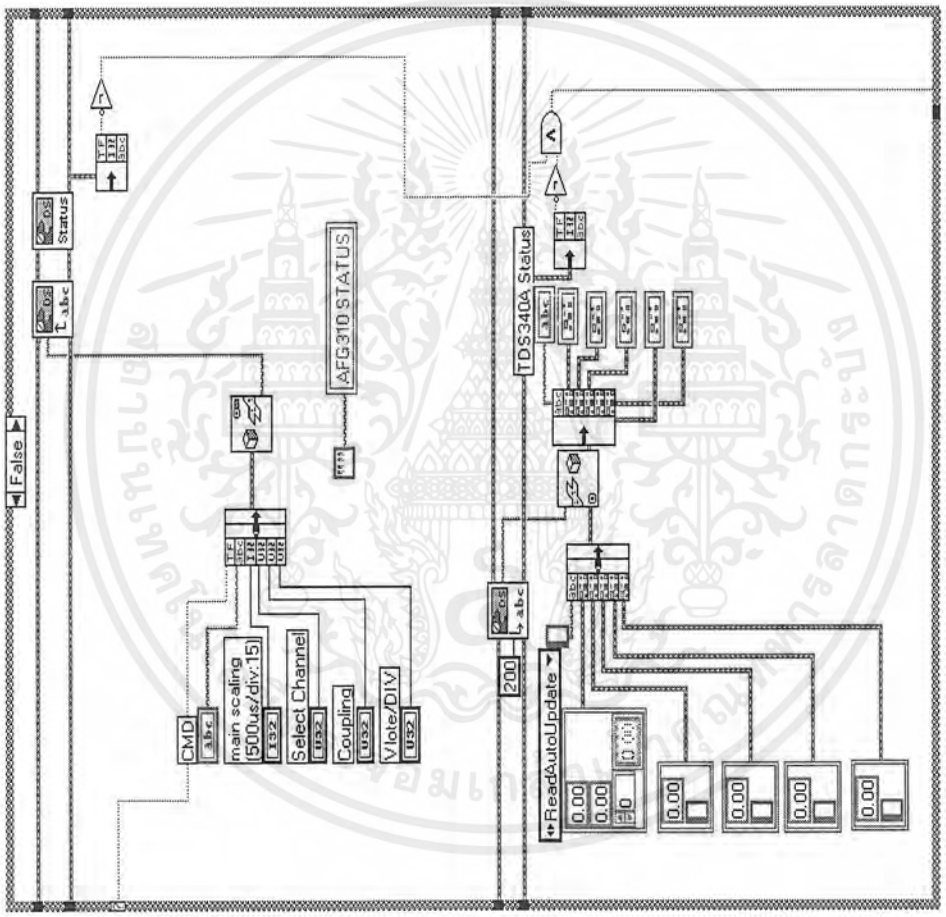
รูปที่ ค.3 Block Diagram ของโปรแกรม Mix_server ขณะเลือกโปรแกรมย่อย Server TDS340A



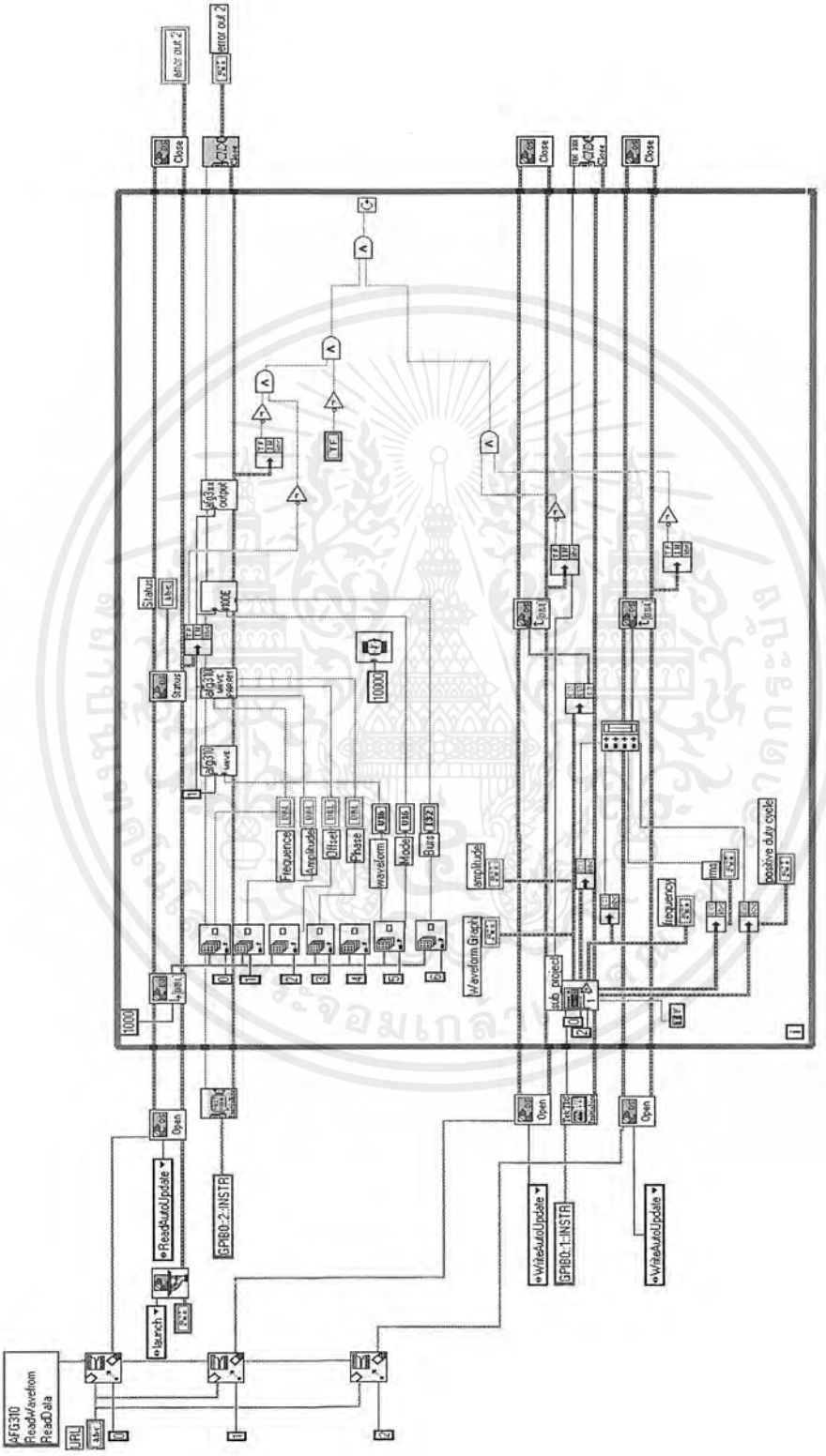
รูปที่ ก.4 Block Diagram ของโปรแกรม Mix_server ขณะเลือกโปรแกรมย่อย Server AFG310



รูปที่ ก.5 Block Diagram ของโปรแกรม Mix_workขณะเลือกโปรแกรมย่อย Work TDS340A



รูปที่ ก.6 Block Diagram ของโปรแกรม Mix_work ขณะเลือกโปรแกรมย่อย Work AFG310



รูปที่ ก.7 Block Diagram ของโปรแกรม Server_www

```
Option Explicit
Dim Data_Array(6) As Double
Private Sub amp_ValueChanged(Value As Variant, PreviousValue As Variant, ByVal OutOfRange As
Boolean)
   Data_Array(1) = Value
   CWDataSocket1.Data.Value = Data_Array()
End Sub
Private Sub cmdConnectAutoUpdate_Click()

   CWDataSocket1.ConnectTo cdURL(0), cwdsWriteAutoUpdate
   CWDataSocket2.ConnectTo cdURL(2), cwdsReadAutoUpdate
   CWDataSocket3.ConnectTo cdURL(1), cwdsReadAutoUpdate
End Sub

Private Sub cmdConnectManualUpdate_Click()
   CWDataSocket1.ConnectTo cdURL(0), cwdsWrite
   CWDataSocket2.ConnectTo cdURL(2), cwdsRead
   CWDataSocket3.ConnectTo cdURL(1), cwdsRead
End Sub

Private Sub cmdDisconnect_Click()

   CWDataSocket1.Disconnect
   CWDataSocket2.Disconnect
   CWDataSocket3.Disconnect
End Sub

Private Sub cmdUpdate_Click()

   CWDataSocket1.Update
   CWDataSocket2.Update
   CWDataSocket3.Update
End Sub
```

```
Private Sub cont_ValueChanged(Value As Variant, PreviousValue As Variant, ByVal OutOfRange As
Boolean)
Data_Array(5) = Value
  CWDataSocket1.Data.Value = Data_Array()
End Sub

Private Sub CWDataSocket1_OnStatusUpdated(ByVal Status As Long, ByVal Error As Long, ByVal
Message As String)
   edStatus.Text = Message
End Sub

Private Sub CWDataSocket2_OnDataUpdated(ByVal Data2 As CWDSLib.CWData)
If (IsArray(Data2)) Then
      CWGraph1.PlotY Data2.Value
      End If
End Sub
Private Sub CWDataSocket3_OnDataUpdated(ByVal Data_3 As CWDSLib.CWData)
Dim Wavefrom
If (IsArray(Data_3)) Then
   Wavefrom = Data_3.Value
   Rfre.Value = Wavefrom(1)
   Ramp.Value = Wavefrom(0)
   Rrms.Value = Wavefrom(2)
   RDuty.Value = Wavefrom(3)
   End If
End Sub

Private Sub fre_ValueChanged(Value As Variant, PreviousValue As Variant, ByVal OutOfRange As
Boolean)
   Data_Array(0) = Value
      CWDataSocket1.Data.Value = Data_Array()
End Sub
```

```
Private Sub mode_ValueChanged(Value As Variant, PreviousValue As Variant, ByVal OutOfRange As
Boolean)
Data_Array(6) = Value
  CWDataSocket1.Data.Value = Data_Array()
End Sub

Private Sub off_ValueChanged(Value As Variant, PreviousValue As Variant, ByVal OutOfRange As
Boolean)
  Data_Array(2) = Value
  CWDataSocket1.Data.Value = Data_Array()
End Sub

Private Sub ph_ValueChanged(Value As Variant, PreviousValue As Variant, ByVal OutOfRange As
Boolean)
    Data_Array(3) = Value
    CWDataSocket1.Data.Value = Data_Array()
End Sub

Private Sub UserControl_Initialize()
cdURL(0).ListIndex = 0
cdURL(1).ListIndex = 0
cdURL(2).ListIndex = 0
End Sub

Private Sub wave_ValueChanged(Value As Variant, PreviousValue As Variant, ByVal OutOfRange As
Boolean)
 Data_Array(4) = Value
 CWDataSocket1.Data.Value = Data_Array()
End Sub
```

ภาคผนวก ข.
ใบงานการใช้ LabVIEW ควบคุมเครื่องมือวัดผ่านเครือข่ายอินเตอร์เน็ต

ใบงานที่ 1

การตรวจสอบความพร้อมของอุปกรณ์ภายนอกและการ์ด GPIB

จุดประสงค์เชิงพฤติกรรม

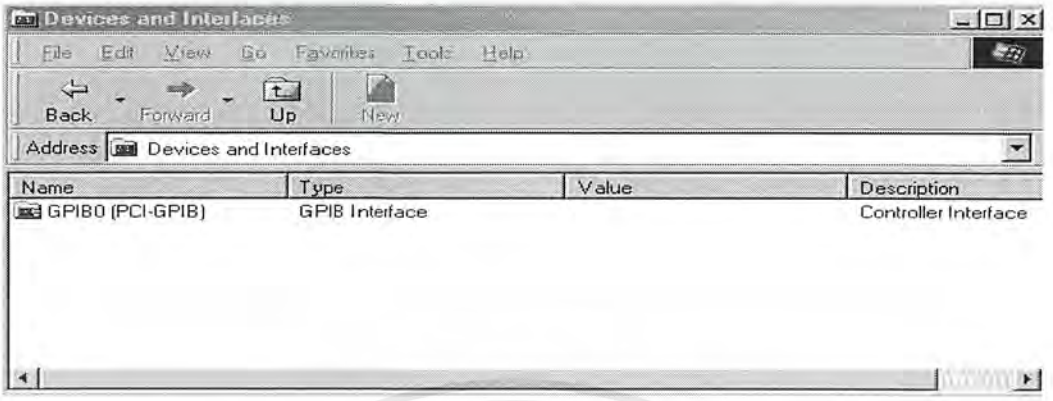
1. สามารถตรวจสอบความพร้อมในการทำงานของอุปกรณ์ภายนอกที่ต่ออยู่กับการ์ด GPIB ได้
2. สามารถใช้คำสั่งในการควบคุมอุปกรณ์ภายนอกอย่างง่าย ๆ ได้

อุปกรณ์ในการทดลอง

1. เครื่องกำเนิดสัญญาณ AFG 310	1	เครื่อง
2. เครื่องวัดสัญญาณไฟฟ้า TDS 340A	1	เครื่อง
3. คู่มือการใช้เครื่องกำเนิดสัญญาณ(User Manual AFG310)	1	เล่ม
4. คู่มือการใช้เครื่องวัดสัญญาณ(User Manual TDS 340A)	1	เล่ม
5. GPIB Card	1	การ์ด
6. GPIB Cable	2	เส้น

ลำดับขั้นการทดลอง

1. ติดตั้งการ์ด GPIB
2. ติดตั้งโปรแกรม LabVIEW 5.1
3. เมื่อผ่าน 2 ขั้นตอนนั้นแล้วที่ Desktop ของคอมพิวเตอร์จะปรากฏไอคอน Measurement & Automation
4. คลิดขวาที่ไอคอน Measurement & Automation เลือก open
5. ดับเบิลคลิกที่ Devices and Interface จะปรากฏไอคอน GPIB[PCI-GPIB] ให้ดับเบิลคลิกที่ไอคอน ดังรูปที่ 1



รูปที่ 1 หน้าต่าง Devices and Interface

6. เปิดเครื่องกำเนิดสัญญาณและเครื่องวัดสัญญาณ ดับเบิลคลิกที่ GPIB[PCI-GPIB] แล้วหน้าจอจะปรากฏ **tool bar Scan For Instruments** เป็น tool bar ที่ใช้สำหรับค้นหาอุปกรณ์ภายนอกที่ Interface โดยการ์ด GPIB ดับเบิลคลิกที่ **tool bar Scan For Instruments**

7. บันทึกผลการทดลองพร้อมสรุปผลการทดลอง

8. ดับเบิลคลิกที่ Insturment0 จะปรากฏหน้าต่าง NI-488.2 Communicator ดังรูปที่ 2 ซึ่งเป็นหน้าต่างสำหรับส่งคำสั่งหรือการอ่านสถานะต่าง ๆ ของอุปกรณ์ที่ติดตั้งในระบบ ขณะนี้เราสามารถที่จะส่งสตริงต่าง ๆ (ตามคู่มือที่ให้มา) เพื่อควบคุมการทำงานของอุปกรณ์ภายนอกได้แล้ว การทดลองควบคุมอุปกรณ์ภายนอก

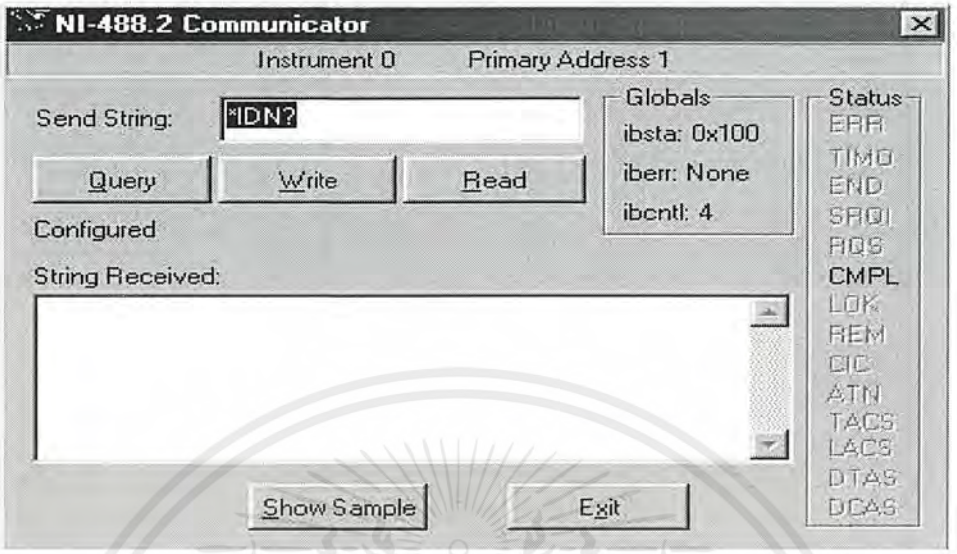
8.1 ทดลองปรับค่าต่าง ๆ ของเครื่องกำเนิดสัญญาณ ดังต่อไปนี้

- Frequency = 50 KHz
- Amplitude = 3.2 V
- Function = RAMP

8.2 ที่ช่อง Send String ให้พิมพ์ข้อความ *IDN? แล้วคลิกที่ Query, Write และ Read บันทึกผลการทดลองตามลำดับ

8.3 ที่ช่อง Send String ให้พิมพ์ข้อความ *RST แล้วคลิกที่ Query, Write และ Read บันทึกผลการทดลองตามลำดับ

8.4 ที่ช่อง Send String ให้พิมพ์ข้อความ *TST? แล้วคลิกที่ Query, Write และ Read บันทึกผลการทดลองตามลำดับ



รูปที่ 2 หน้าต่าง NI-488.2 Communicator

9. ดับเบิลคลิกที่ Instrument1 ทำตามขั้นตอนการทดลองที่ 8 และ 9

หมายเหตุ

การส่งสตริงเพื่อไปควบคุมและอ่านค่าของอุปกรณ์ภายนอก รูปแบบของสตริง(Format String) ของแต่ละอุปกรณ์จะไม่เหมือนกัน แต่ก็มีบ้างคำสั่งที่เป็นคำสั่งมาตราฐานสามารถใช้ได้กับทุก ๆ อุปกรณ์ เราสามารถศึกษาคำสั่งต่าง ๆ ได้จากคู่มือของอุปกรณ์นั้น ๆ

คำถามท้ายการทดลอง

1. เมื่อปิดเครื่องกำเนิดสัญญาณและเครื่องวัดสัญญาณหรืออย่างใดอย่างหนึ่งแล้ว ใช้ Tool bar Scan For Instruments ผลจะเป็นอย่างไร อธิบายสาเหตุที่เกิดผลเช่นนั้นด้วย

2. ถ้าต้องการควบคุมอุปกรณ์ภายนอกโดยการควบคุมผ่านเครื่องคอมพิวเตอร์เพียงอย่างเดียว (ตัดการควบคุมของปุ่มกดของเครื่องมือวัด) เราจะต้องส่งใช้คำสั่งใด และคำสั่งสตริงนั้นใช้ได้กับอุปกรณ์ทั้งสองชนิดได้หรือไม่

3. จากข้อ 2 ถ้าต้องการคืนการควบคุมให้กับปุ่มกดของเครื่องมือวัด เราจะต้องใช้คำสั่งใด

ใบงานที่ 2
การใช้โปรแกรม LabVIEW ควบคุมเครื่องกำเนิดสัญญาณ AFG310

จุดประสงค์เชิงพฤติกรรม

1. สามารถใช้ Function VISA Find Resource เพื่อค้นหาอุปกรณ์ในระบบได้
2. สามารถใช้ Function VISA Open, Close, Read, Write เพื่อติดต่อ, สั่งงานและอ่านค่าอุปกรณ์ภายนอกได้
3. สามารถเขียนโปรแกรมโดยใช้ LabVIEW เพื่อควบคุมเครื่องกำเนิดสัญญาณ AFG310ได้

อุปกรณ์ที่ใช้ในการทดลอง

1. เครื่องกำเนิดสัญญาณ AFG 310, 1 เครื่อง
2. คู่มือการใช้เครื่องกำเนิดสัญญาณ(User Manual AFG310) 1 เล่ม
3. เครื่องคอมพิวเตอร์ 1 เครื่อง
4. เหมือนกับข้อ 5 และ 6 ในใบงานที่ 1

การทดลองที่ 2.1

ลำดับขั้นการทดลอง

1. เปิดโปรแกรม LabView 5.1
2. ไปที่ *Windows >> Show Diagram* เลือก *Function palette>> Instrument I/O >> VISA >> Easy VISA Find Resource .vi* มาวางไว้ที่ Diagram ดับเบิลคลิก เมื่อ Front panel ของ ***Easy VISA Find Resource .vi*** ปรากฏ ให้ RUN Program
3. บันทึกผลการทดลองพร้อมสรุปผลการทดลอง
4. ไปที่ *Windows >> Show Diagram* บันทึกสิ่งที่ปรากฏขึ้นใน ***Easy VISA Find Resource .vi Diagram***

การทดลองที่ 2.2

ลำดับขั้นการทดลอง

ที่ Front Panel

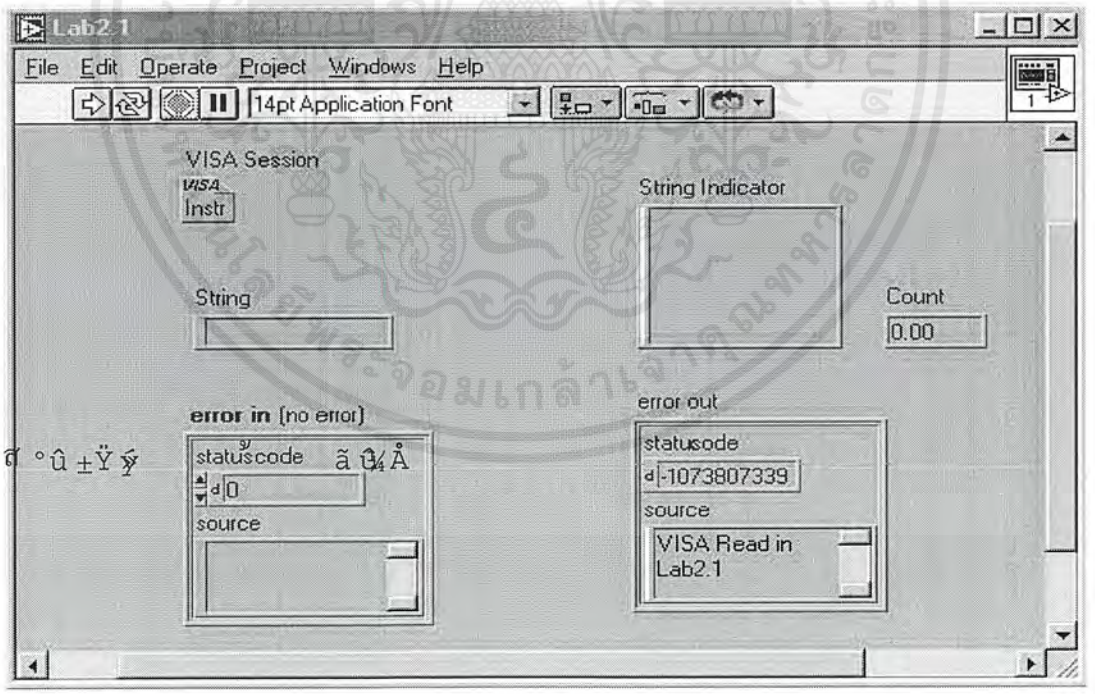
1. เขียนโปรแกรมดังรูปที่ 1 โดยลำดับดังนี้

- ***VISA Session*** เพื่อใช้ VISA ฟังก์ชั่น ซึ่งอยู่ใน *Path & Refnum Subpalette*

- ***String Control*** และ ***String Indicator*** โดย String Control ใช้สำหรับส่งคำสั่งที่เราต้องการไปควบคุมอุปกรณ์ภายนอก String Indicator ใช้สำหรับแสดงสถานะต่าง ๆ ของอุปกรณ์ภายนอก ซึ่งอยู่ใน *String & Table Subpalette*

- ***Count*** แสดงจำนวนตัวอักษรที่แสดงใน String Indicator ซึ่งอยู่ใน *Numeric Subpalette >> Digital Indicator*

- ***error in*** และ ***error out*** ตัวตรวจจับข้อผิด พลาดต่าง ๆ ของ Function VISAซึ่งอยู่ใน *Array & Cluster Subpalette*



รูปที่ 1 Front Panel

ที่ Block Diagram

2. เขียนโปรแกรมดังรูปที่ 2 โดยลำดับดังนี้

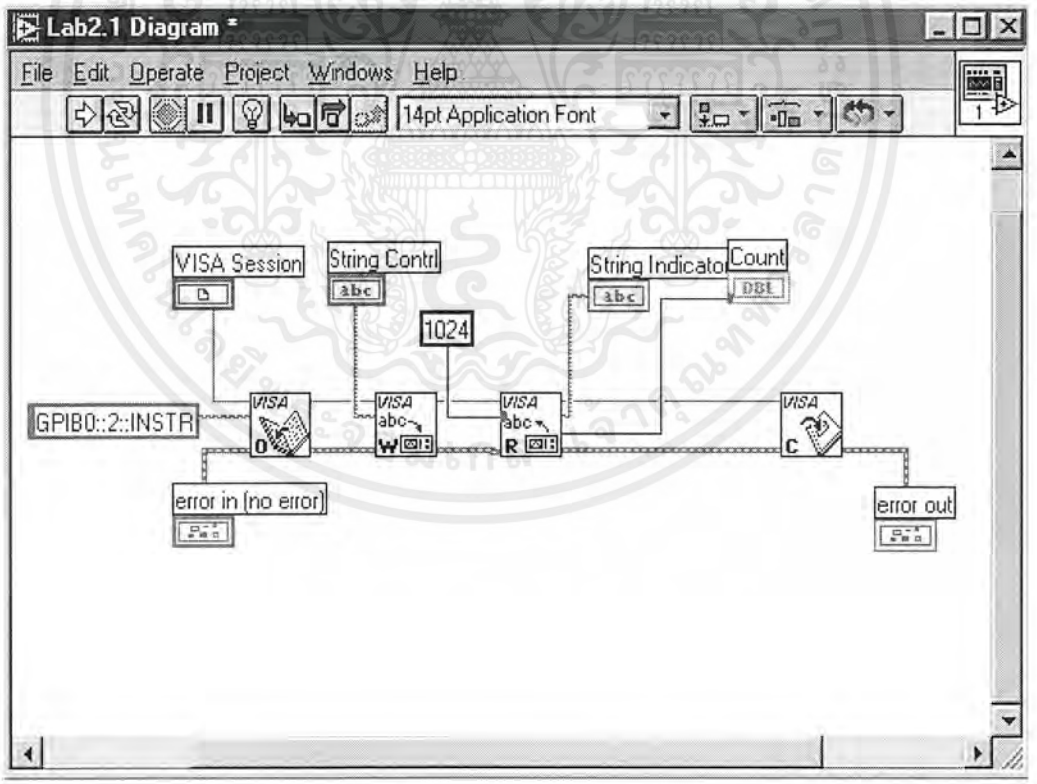
- ***GPIB0::2::INSTR*** เป็นชื่อของอุปกรณ์ภายนอกที่ต้องการจะติดต่อด้วย โดยถูกเขียนให้อยู่ในรูปแบบ VISA Format Sting แล้ว ซึ่งอยู่ใน *String Subpaltte >> String Constant*

- ***VISA Open*** เป็นการเปิด Session ของ VISA เพื่อเริ่มต้นการใช้ฟังก์ชั่น VISA อยู่ใน *Instrument I/O >> VISA>> VISA Open*

- ***VISA Close*** เป็นการปิด Session ของ VISA เพื่อคืนทรัพยากรของระบบ อยู่ใน *Instrument I/O >> VISA>> VISA Close*

- ***VISA Write*** เป็นการเขียนข้อความคำสั่ง(String Command) เพื่อควบคุมอุปกรณ์ภายนอก โดยจะรับคำสั่งจาก String Control ซึ่งอยู่ใน *Instrument I/O >> VISA>> VISA Write*

- ***VISA Read*** เป็นการอ่านสถานะของอุปกรณ์ภายนอกแล้วจะ แสดงค่าที่ได้ไปที่ String Indicator ซึ่งอยู่ใน *Instrument I/O >> VISA>> VISA Read*



รูปที่ 2 Block Diagram

3. พิมพ์ *IDN? ที่ String Control บน Front Panel

4. Run โปร 

5. บันทึกผลการทดลองพร้อมสรุปผลการทดลอง

6. พิมพ์ข้อความต่าง ๆ ดังต่อไปนี้ที่ *String Control* แล้วสังเกตที่เครื่องกำเนิดสัญญาณ บันทึกผลการทดลองพร้อมสรุปผลการทดลอง (ใช้เครื่องวัดสัญญาณ วัดสัญญาณที่ OUTPUT ของเครื่องกำเนิดสัญญาณเพื่อให้เห็นความเปลี่ยนแปลงได้ชัดเจนยิ่งขึ้น)

6.1 SOUR1: FREQUENCY 1.55E+3 Hz

6.2 SOUR1:VOLTAGE:AMPLITUDE 2.50E+0

6.3 SOUR1:VOLATE:OFFSET 0e+0

6.4 SOUR1:PHASE:ADJUST 0.0E+0DEG

6.5 SOUR1:FUNCTION SIN;

6.6 SOUR1:FUNCTION SQU;

6.7 SOUR1:FUNCTION TRI;

6.8 SOUR1:FUNCTION RAMP;

6.9 SOUR1:FUNCTION PLUS;

6.10 SOUR1:FUNCTION DC;

6.11 SOUR1:FUNCTION PRN;

6.12 SOUR1:FUNCTION USER1;

7. ให้ทดลองเปลี่ยนค่าต่าง ๆ ได้ตามต้องการแล้วสังเกตการเปลี่ยนแปลง

หมายเหตุ จากข้อ 6.1-6.4 สามารถพิมพ์จนครบแล้วสั่ง RUN ครั้งเดียวได้แต่จากข้อ 6.5 –6.12 ต้องพิมพ์ทีละครั้งต่อหนึ่งการ RUN

คำถามท้ายการทดลอง

1. จากการทดลองที่ 2 สามารถเปลี่ยนตัวอักษรเป็นตัวพิมพ์เล็กหรือผสมกันระหว่างตัวอักษรพิมพ์ใหญ่และพิมพ์เล็กได้หรือไม่

2. จงอธิบายลำดับขั้นตอนในการใช้ Function VISA เพื่อควบคุมเครื่องกำเนิดสัญญาณ

3. จงใช้โปรแกรม LabVIW เพื่อเขียนโปรแกรมควบคุมเครื่องกำเนิดสัญญาณรุ่น AFG310 ให้สามารถเปลี่ยนแปลงความถี่, แอมปลิจูด, และ Wave From ได้

ใบงานที่ 3
การใช้โปรแกรม LabVIEW ควบคุมเครื่องวัดสัญญาณ TDS 340A

จุดประสงค์เชิงพฤติกรรม

1. สามารถเขียนโปรแกรมโดยใช้ LabVIEW เพื่อควบคุมเครื่องกำเนิดสัญญาณ TDS 340A ได้

อุปกรณ์ที่ใช้ในการทดลอง

1. เครื่องกำเนิดสัญญาณ AFG 310 1 เครื่อง
2. เครื่องวัดสัญญาณไฟฟ้า TDS 340A 1 เครื่อง
3. คู่มือการใช้เครื่องกำเนิดสัญญาณ(User Manual AFG310) 1 เล่ม
4. คู่มือการใช้เครื่องวัดสัญญาณ(User Manual TDS340A) 1 เล่ม
5. เครื่องคอมพิวเตอร์ 1 เครื่อง
6. เหมือนกับข้อ 5 และ 6 ในใบงานที่ 1

การทดลองที่ 3.1

ลำดับขั้นการทดลอง

1. ทำตามใบงานที่ 2 การทดลองที่ 2.2 ข้อ 1 โดยเปลี่ยน ***String Constant*** ที่ Block Diagram เป็น GPIB0::1::INSTR
2. พิมพ์ *IDN? ที่ ***String Control***
3. Run Program 
4. สังเกตที่เครื่องวัดสัญญาณ บันทึกผลการทดลองพร้อมสรุปผลการทดลอง
5. ทำเช่นเดิมจากข้อ 2 – 4 โดยเปลี่ยนข้อความเป็น *RST

การทดลองที่ 3.2

ลำดับขั้นการทดลอง

1. ต่อเครื่องวัดสัญญาณ CH1 เข้ากับ Output ของเครื่องกำเนิดสัญญาณ
2. ปรับเครื่องกำเนิดสัญญาณได้ตามต้องการ
3. เปิดไฟล์ของการทดลองที่ 3.1
4. พิมพ์ AUTOSET EXECUTE ที่ ***String Control***
5. Run Program 
6. สังเกตที่เครื่องวัดสัญญาณ บันทึกผลการทดลองพร้อมสรุปผลการทดลอง

การทดลองที่ 3.3

ลำดับขั้นการทดลอง

1. ต่อเครื่องวัดสัญญาณ CH1 เข้ากับ Output ของเครื่องกำเนิดสัญญาณ
2. ปรับเครื่องกำเนิดสัญญาณได้ตามต้องการ
3. เปิดไฟล์ของการทดลองที่ 3.1
4. พิมพ์ *RST ที่ ***String Control*** แล้ว Run Program
5. พิมพ์คำสั่งต่าง ๆ ต่อไปนี้ที่ ***String Control*** แล้ว Run Program ตามลำดับ สังเกตการเปลี่ยนแปลงที่เครื่องวัดสัญญาณ ทุก ๆ ครั้งที่มีการ Run Program

    5.1 :MEASU:MEAS1:TYP AMPL;SOU CH1;:MEASU:MEAS1:STA4TE ON;

    5.2 :MEASU:MEAS2:TYP FREQ;SOU CH1;:MEASU:MEAS2:STATE ON;

    5.3 :MEASU:MEAS3:TYP RMS;SOU CH1;:MEASU:MEAS3:STATE ON;

    5.4 AUTOSET EXECUTE

    5.5 MEASU:MEAS1:VAL?;UNI?

    5.6 MEASU:MEAS2:VAL?;UNI?

    5.7 MEASU:MEAS3:VAL?;UNI?

หมายเหตุ

- จากข้อ 5.1 – 5.3 รูปแบบของคำสั่งแบ่งออกได้ 3 ส่วนแต่ละส่วนขั้นด้วยเครื่องหมาย ; ดังนี้

1. MASU:MEAS1:TYP AMPL ; ใช้ระบุ message1 ให้แสดงค่า แอมปลิจูด

2. SOU CH1; บ่งบอกว่าใช้สัญญาณ Input จาก Channel ที่ 1

3. MEASU:MEAS1:STATE ON; ใช้บอกให้เครื่องกำเนิดสัญญาณให้แสดง message1 ที่หน้าจอภาพของเครื่องวัดสัญญาณเอง

จากข้อ 5.4 เป็นคำสั่งเพื่อปรับค่าเครื่องวัดสัญญาณ TDS340A ให้เหมาะสม

- จากข้อ 5.5 – 5.7 เป็น QUERY ใช้อ่านค่าที่ได้จากการวัดสัญญาณ

คำถามท้ายการทดลอง

1. ถ้าต้องการวัดสัญญาณ โดยที่ Input อยู่ที่ Channel 2 จะมีขั้นตอนอย่างไร

2. จงใช้ LabVIEW เพื่อเขียนโปรแกรมควบคุมเครื่องวัดสัญญาณรุ่น TDS 340A โดยให้สามารถอ่านค่า Frequency, Amplitude, RMS ได้

3. จงใช้ LabVIEW เพื่อเขียนโปรแกรมควบคุมเครื่องวัดสัญญาณ TDS 340A และเครื่อง กำเนิดสัญญาณ AFG 310

ใบงานที่ 4
การใช้โปรแกรม LabVIEW ควบคุมเครื่องวัดผ่านเครือข่ายอินเตอร์เน็ต

จุดประสงค์เชิงพฤติกรรม

1. สามารถใช้ Datasocket Open, Close, Read, Write เพื่อส่ง ข้อความ, ตัวเลข, หรือข้อมูลต่าง ๆ ได้
2. สามารถให้ Datasocket ร่วมกับ VISA เพื่อควบคุมเครื่องมือวัดผ่านระบบอินเตอร์เน็ตได้

อุปกรณ์ที่ใช้ในการทดลอง

1. เครื่องกำเนิดสัญญาณ AFG310 1 เครื่อง
2. เครื่องวัดสัญญาณไฟฟ้า TDS340A 1 เครื่อง
3. คู่มือการใช้เครื่องกำเนิดสัญญาณ (User Manual AFG310) 1 เล่ม
4. คู่มือการใช้เครื่องวัดสัญญาณ (User Manual TDS340A) 1 เล่ม
5. เครื่องคอมพิวเตอร์ที่มีโปรแกรม LabVIEW, โปรแกรม Datasocket การ์ด LAN และต่ออยู่กับเครือข่ายอินเตอร์เน็ต 2 เครื่อง
6. เหมือนกับข้อ 5 และ 6 ในใบงานที่ 1

การทดลองที่ 4.1

ลำดับขั้นการทดลอง

1. จงเขียนโปรแกรม Lab4_Write.vi ซึ่งเป็นโปรแกรมที่ทำหน้าเป็นต้นทาง (Source) เพื่อส่งข้อความต่าง ๆ ผ่านเครือข่ายอินเตอร์เน็ตไปยังเครื่องปลายทาง (Target) ดังรูปที่ 1 และ รูปที่ 2 โดยลำดับ ดังนี้

ที่ Front Panel

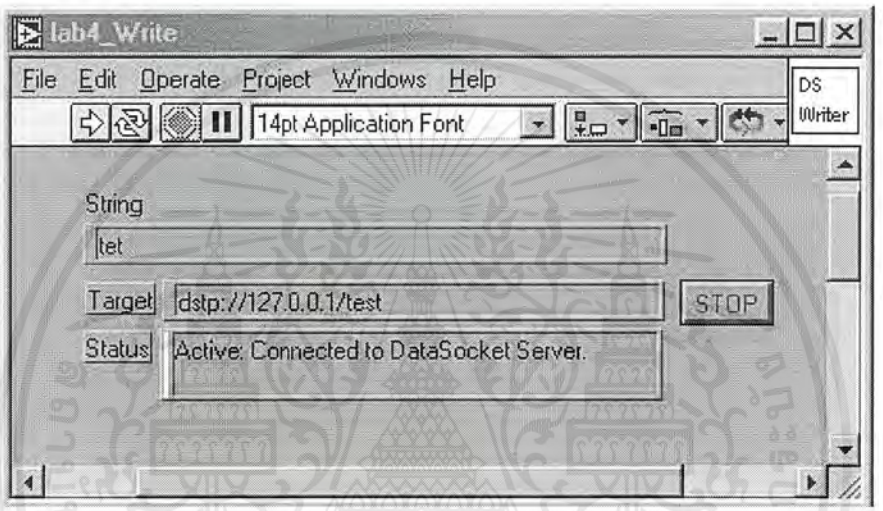
- ***String*** เป็น *String Control* เพื่อพิมพ์ข้อความต่าง ๆ ที่ต้องการส่ง
- ***Target*** เป็น *String Control* เพื่อระบุ URL (Uniform Resource Locators) ของเครื่องปลายทาง URL ของ Datasocket นี้มีลักษณะคล้ายกันกับ URL ของ HTTP (Hypertext transfer protocol) เช่น

URL ของ HTTP จะต้องนำหน้าด้วย HTTP:// แต่ของ Datasocket จะนำหน้าด้วย Dstp:// (Datasocket transfer protocol)

ตัวอย่าง Dstap://ed03.inded.kmitl.ac.th/test

Dstp://161.246.14.106/test

- ***Status*** เป็น *String Indicator* เพื่อแสดงสถานะในการติดต่อสื่อสาร
- ***Stop*** เป็น *Rectangular stop button* เพื่อควบคุม While Loop ใช้หยุดการ RUN โปรแกรม



รูปที่ 1 Lab4_Write.vi Front Panel

ที่ Block Diagram

- ***Datascoket Server Control*** ใช้เปิดโปรแกรม Datasocket Server ซึ่ง Datasocket มีการปฏิบัติการอยู่ 4 อย่างดังนี้

1. Launch เริ่มใช้โปรแกรม Datasocket Server
2. Show แสดงหน้าต่างของโปรแกรม Datasocket Server
3. Hide ซ่อนหน้าต่างโปรแกรม Datasocket Server
4. Close ยกเลิกการใช้งานโปรแกรม Datasocket Server

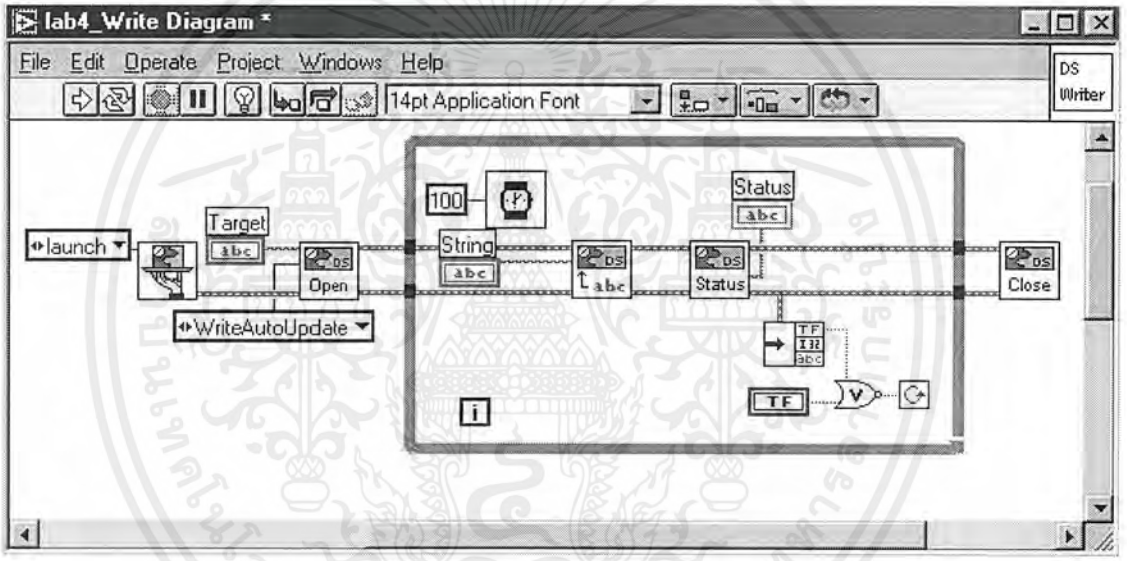
Datasocket Server Control อยู่ใน *Function palette>>Communication>> Datasocket >> Datasocket Advanced>>datasocket Server Contorl.vi*

- ***Datasocket Open Connection*** ใช้เมื่อเริ่มมีการเชื่อมต่อระบบผ่านเครือข่ายอินเตอร์เน็ต โดยการกำหนด URL และโหมดในการเชื่อมต่อ (Access Mode) ซึ่ง *Datasocket Open Connection* อยู่ใน *Sub palette Communication >> Datasocket*

- ***Datasocket Write String*** เมื่อเริ่มมีการเชื่อมระบบต่อแล้ว ต้องมีการกำหนดว่าจะให้โปรแกรมนี้เป็นโปรแกรมสำหรับส่งหรือรับข้อมูล ฉะนั้น *Datasocket Write String* จึงเป็นการบ่งบอกว่าต้องการส่งข้อมูลที่เป็นกลุ่มตัวอักษรไปให้กับเครื่องปลายทางนั้นเอง ซึ่ง *Datasocket Write String* อยู่ใน *Sub palette Communication* เช่นเดียวกับ *Datasocket Open Connection*

- ***Datasocket Status*** ทำหน้าที่ส่งข้อความแจ้งสถานะต่าง ๆ ของ Datasocket ซึ่ง ***Datasocket Status*** อยู่ใน *Sub palette* เดียวกันกับ *Datasocket Server Control*

- ***Datasocket Close Connection*** ใช้สำหรับยกเลิกการเชื่อมต่อระบบ ซึ่ง *Datasocket Close connection* นี้จะอยู่ใน *Sub palette* เดียวกับ *Datasocket Open Connection*



รูปที่ 2 Block Diagram ของ Lab4_Write

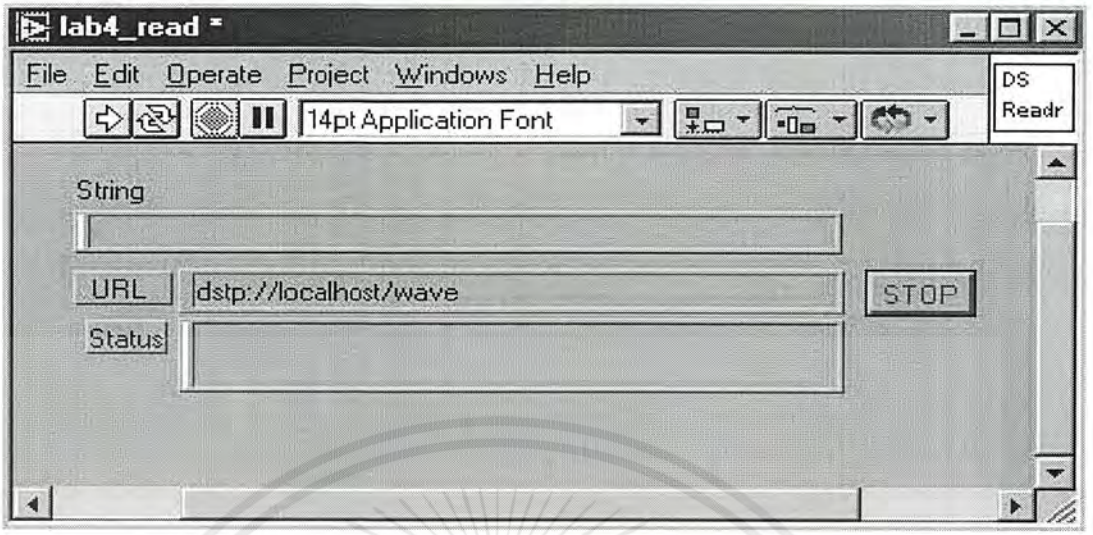
2. จงเขียนโปรแกรม Lab4_read.vi ซึ่งเป็นโปรแกรมที่ทำหน้าเป็นปลายทาง เพื่อรับข้อความต่าง ๆ ที่ส่งมาจากเครื่องต้นทาง ดังรูปที่ 3 และ รูปที่ 4 โดยลำดับ ดังนี้

ที่ Front Panel

ที่ Front Panel ของ Lab4_read.vi จะมีลักษณะคล้ายกับ Front Panel ของ Lab4_Write.vi มาก โดยจะมีข้อแตกต่างกันเล็กน้อยดังนี้

- ***String*** เป็น *String Indicator* เพื่อรับข้อความต่าง ๆ ที่ส่งมาจากเครื่องต้นทาง

- ***URL*** เพื่อระบุ URL ของเครื่อง

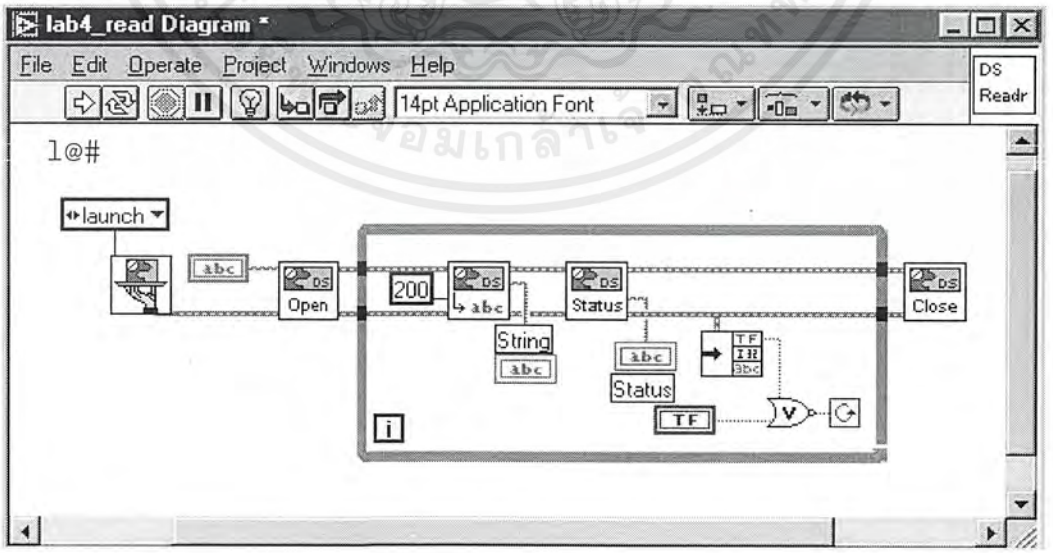


รูปที่ 3 Front Panel ของ Lab4_read.vi

ที่ Block Diagram

ที่ Block Diagram ของ Lab4_read.vi ก็จะมีการเปลี่ยนแปลงจาก Block Diagram ของ Lab4_Write.vi เพียงเล็กน้อยเช่นกัน โดยมีสิ่งที่เปลี่ยนแปลงมีดังนี้

- ***Datasocket Open Connection*** ไม่ต้องใส่ Access Mode
- ***Datasocket Read String*** เพื่อรับค่าตัวอักษรจากเครื่องต้นทางโดยมีพารามิเตอร์ Time out เพื่อกำหนดระยะเวลาสูงสุดต่อการอ่าน 1 ครั้ง



รูปที่ 4 Block Diagram ของ Lab4_read.vi

3. เปิดโปรแกรม Lab4_read.vi และ Lab4_Write.vi พร้อมกัน 2 โปรแกรม

4. ที่ Lab4_read.iv กำหนด URL เป็น dstp://127.0.0.1/xxx

- 127.0.0.1 หมายความว่า URL นี้เป็น URL ที่สามารถใช้รับ – ส่งข้อมูลภายในเครื่องคอมพิวเตอร์เครื่องเดียวกันได้

- /xxx เป็นชื่อ ส่วนขยายที่เราสามารถกำหนดเองได้

ตัวอย่าง

Dstp://127.0.0.1/test

5. ที่ Lab4_Write.vi ทำเช่นเดียวกันกับข้อที่ 4 เพื่อระบุ URL ของเครื่องปลายทาง (Target)

6. ที่ Lab4_Write.vi พิมพ์ ข้อความต่าง ๆ ที่ *String Control* ได้ตามต้องการ

7. RUN โปรแกรม Lab4_Write.vi 

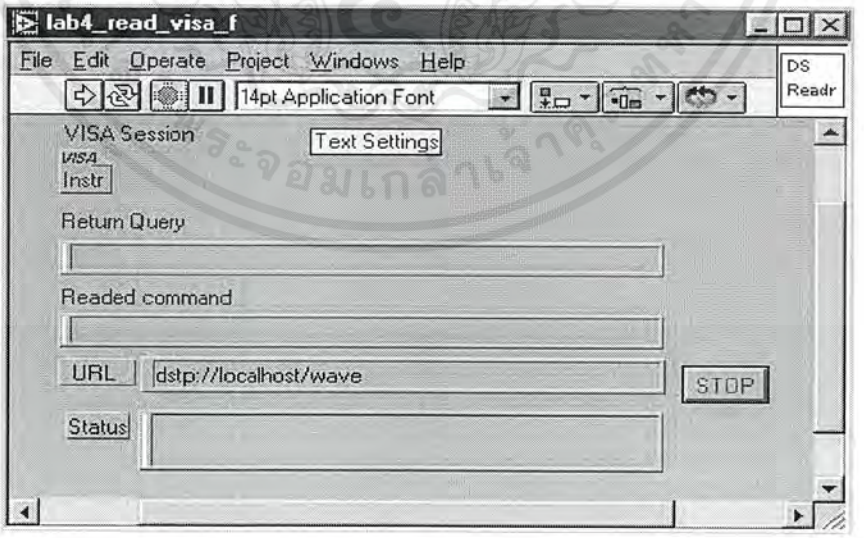
8. RUN โปรแกรม Lab4_read.vi 

9. สังเกตการเปลี่ยนแปลงของทั้ง 2 โปรแกรมแล้วบันทึกผลการทดลอง, สรุปผลการทดลอง

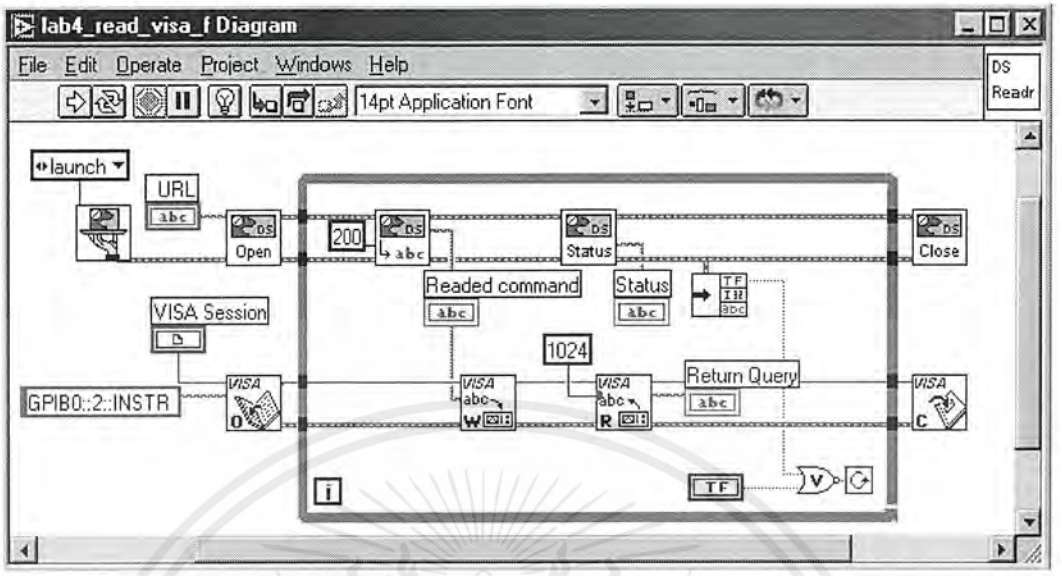
การทดลองที่ 4.2

ลำดับขั้นการทดลอง

1. ดัดแปลงโปรแกรม Lab4_read.vi เพื่อให้สามารถควบคุมเครื่องกำเนิดสัญญาณผ่านเครือข่ายอินเตอร์เน็ตได้ ดังรูปที่ 5 และ รูปที่ 6



รูปที่ 5 Front Panel ของ Lab4_read.vi ที่ดัดแปลงแล้ว



รูปที่ 6 Block Diagram ของ Lab4_read.vi ที่ดัดแปลงแล้ว

2. กำหนด URL เป็นชื่อ DNS หรือ IP Address ของเครื่องคอมพิวเตอร์แต่ละตัว ในที่นี้คือเครื่องคอมพิวเตอร์ที่ต่ออยู่กับเครื่องมือวัด

3. RUN โปรแกรม Lab4_read.vi 

4. เปิดโปรแกรม Lab4_Write.vi ที่เครื่องคอมพิวเตอร์เครื่องใดก็ได้ที่ต่ออยู่กับเครือข่ายอินเตอร์เน็ต

5. ที่ lan4_Write.vi Front Panel >> Target ให้พิมพ์ URL ของเครื่องปลายทาง (URL เดียวกับข้อ 2)

6. เปลี่ยนค่าต่าง ๆ ของเครื่องกำเนิดสัญญาณ

7. ที่ Lab4_Write.vi >> Front Panel >>  ให้พิมพ์ *RST

8. RUN โปรแกรม Lab4_Write.vi

9. สังเกตการเปลี่ยนแปลงที่เครื่องคอมพิวเตอร์ที่ RUN โปรแกรม Lab4_read.vi และ เครื่องกำเนิดสัญญาณ บันทึกผลการทดลองปละสรุปผลการทดลอง

10. ที่โปรแกรม Lab4_Write.vi >> Front panel >> String ให้พิมพ์ข้อความต่อไปนี้ แล้วRUN โปรแกรมจากนั้นทำซ้ำข้อ 9 –10 ตามลำดับ

11.1. SOUR1: FREQUENCY 1.55E+3 Hz

11.2. SOUR1:VOLTAGE:AMPLITUDE 2.50E+0

11.3. SOUR1:VOLATE:OFFSET 0e+0

11.4. SOUR1:PHASE:ADJUST 0.0E+0DEG

11.5. SOUR1:FUNCTION SIN;

11.6. SOUR1:FUNCTION SQU;

11.7. SOUR1:FUNCTION TRI;

11.8. SOUR1:FUNCTION RAMP;

11.9. SOUR1:FUNCTION PLUS;

11.10. SOUR1:FUNCTION DC;

11.11. SOUR1:FUNCTION PRN;

11.12. SOUR1:FUNCTION USER1;

การทดลองที่ 4.3

ลำดับขั้นการทดลอง

1. ที่ Block Diagram ของโปรแกรม Lab4_read.vi ที่ใช้ในการทดลองที่ 4.2 ให้เปลี่ยนชื่อของเครื่องมือวัดจากเดิมเป็น GPIB0::2::INSTR ให้เป็น GPIB0::1::INSTR

2. ทำการทดลองซ้ำการทดลองที่ 4.3 ข้อ 2 – 10

3. ที่โปรแกรม Lab4_Write.vi >> Front panel >> String ให้พิมพ์ข้อความต่อไปนี้ แล้วRUN โปรแกรมจากนั้นทำซ้ำการทดลองที่ 4.2 ข้อ 9 –10 ตามลำดับ

3.1. :MEASU:MEAS1:TYP AMPL;SOU CH1;:MEASU:MEAS1:STA4TE ON;

3.2. :MEASU:MEAS2:TYP FREQ;SOU CH1;:MEASU:MEAS2:STATE ON;

3.3. :MEASU:MEAS3:TYP RMS;SOU CH1;:MEASU:MEAS3:STATE ON;

3.4. AUTOSET EXECUTE

3.5. MEASU:MEAS1:VAL?;UNI?

3.6. MEASU:MEAS2:VAL?;UNI?

3.7. MEASU:MEAS3:VAL?;UNI?

หมายเหตุ โปรแกรมตัวอย่างการใช้งาน Datasocket สามารถดูได้ขณะเริ่มใช้งาน LabVIEW ที่ Search Example >> Datasocket

คำถามท้ายการทดลอง

1. URL หมายถึงอะไร และโปรแกรมส่วนรับหรือส่งข้อมูลเป็นตัวกำหนดชื่อของ URL

2. จงเขียนโปรแกรมเพื่อส่งข้อมูล โดยให้สามารถส่งข้อมูลที่เป็น Numeric, Boolean และ String ได้ในโปรแกรมเดียว

3. จากข้อที่ 1 จงเขียนโปรแกรมรับข้อมูลดังกล่าว

4. ในโปรแกรม Lab4_Write.vi นี้ไม่สามารถแสดงค่าที่อ่านได้จากเครื่องมือวัด จงดัดแปลงโปรแกรม Lab4_Write.vi ให้สามารถแสดงค่าที่อ่านจากเครื่องมือวัดได้ เช่น คำสั่ง *IDN?

ภาคผนวก ค.

Programmer Manual

รูปแบบคำสั่ง

เราสามารถควบคุม Oscilloscope ได้โดยผ่านพอร์ต GPIB หรือ RS-232 ซึ่งเราจะต้องใช้ชุดคำสั่งและการตรวจสอบ และในที่นี้จะเป็นการบอกรูปแบบของคำสั่งและการตรวจสอบ รวมทั้งสัญลักษณ์ของ Oscilloscope ที่ใช้ในกระบวนการนี้

เราจะส่งคำสั่งไปควบคุม Oscilloscope โดยใช้รหัส ASCII ซึ่งจะแสดงรูปแบบของรหัสอยู่ในภาคผนวก คู่มือที่ใช้ในการ BNF (Backus-Naur Form) แสดงดังในตารางที่ ค.1

ตารางที่ ค.1 สัญลักษณ์ของ BNF และความหมาย

สัญลักษณ์	ความหมาย
< >	ขอบเขต
::=	กำหนดค่า
|	Exclusive OR
{ }	กลุ่ม,หนึ่งประเภทที่ต้องการ
[ ]	ทางเลือก,สามารถละเว้นได้
( )	หมายเหตุ

โครงสร้างของคำสั่งและการตรวจสอบ

ชุดคำสั่งนั้นจะมีทั้งคำสั่งในการตั้งค่าและคำสั่งที่ใช้ในการตรวจสอบ ซึ่งคำสั่งทั้งสองแบบนี้จะมีรูปแบบของคำสั่งเหมือนกัน แต่คำสั่งในการตรวจสอบค่าจะต้องมีเครื่องหมาย "?" ต่อท้ายคำสั่งด้วย ตัวอย่างเช่น

คำสั่งในการตั้งค่าคือ

ACQuire : MODe

คำสั่งในการตรวจสอบก็คือ

ACQuire : MODeM?

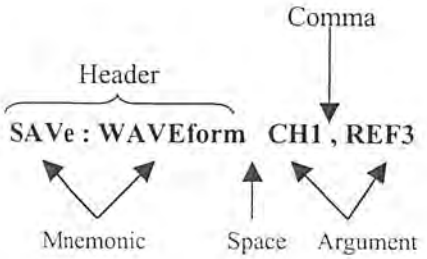
แต่ชุดคำสั่งทุกชุดนั้นไม่จำเป็นนี่จะต้องมีทั้งคำสั่งในการตั้งค่าคำสั่งในการตรวจสอบเสมอไปบางคำสั่งก็มีแต่คำสังในการตั้งค่าเพียงอย่างเดียวและบางคำสั่งก็มีคำสั่งในการตรวจสอบอย่างเดียวเช่นกัน

ประโยคคำสั่ง คือ ชื่อคำสั่งหรือการตรวจสอบ ตามด้วยข้อมูลที่ Oscilloscope ต้องการที่จะนำมาเพื่อปฏิบัติงานตามคำสั่งนั้นๆ

ประโยคคำสั่ง ประกอบด้วยส่วนสำคัญที่แตกต่างกัน 5 ส่วน ดังที่แสดงในตารางที่ ค.2 จะเป็นคำจำกัดความของแต่ละส่วน และในรูปที่ 1 เป็นตัวอย่างคำสั่งที่ใช้งาน

ตารางที่ ค.2 ส่วนประกอบของประโยคคำสั่ง

สัญลักษณ์	ความหมาย
<Header>	ชื่อคำสั่งพื้นฐาน ถ้าตอนท้ายของ Headerมีเครื่องหมาย "?"จะเป็นคำสั่งที่ใช้ในการตรวจสอบ Header จะเริ่มต้นด้วยเครื่องหมาย ":" ถ้าคำสั่งที่เชื่อมโยงอยู่กับคำสั่งอื่นจะต้องมีเครื่องหมาย ":" คั่นก่อนที่จะเริ่มต้นคำสั่งต่อไปในการเริ่มต้นคำสั่งถ้า Header ของคำสั่งขึ้นต้นด้วยเครื่องหมาย "*" แล้วก็ไม่จำเป็นต้องใช้เครื่องหมาย ":" ก็ได้
<Mnemonic>	หัวฟังก์ชันย่อย ปกติ Header ของคำสั่งจะมี Mnemonic เพียงตัวเดียว ถ้า Header ของคำสั่งมีหลายๆ Mnemonic จะต้องแยกออกทีละทางจากแต่ละที่อื่นๆ โดยเครื่องหมาย ":"
<Argument>	ปริมาร,คุณภาพ และข้อจำกัด หรือ ขอบเขตจำกัดของ Header ซึ่งไม่จำเป็นที่ทุกคำสั่งจะต้องมี Argument และในขณะเดียวกัน 1 คำสั่งก็สามารถมีหลายๆ Argument ได้ โดยที่ Argument จะแยกจาก Header ด้วยการเคาะ Space Bar และจะแยกจากกันโดยเครื่องหมาย ","
<Common>	Comma (,) 1 ตัวจะอยู่ระหว่าง 1 Argument ของหลายๆ Argument ซึ่งจะต้อทำการกด Spaceทั้งก่อนและหลัง Comma
<Space>	ช่องว่างระหว่าง Header และ Argument ซึ่งอาจจะมีหลายๆช่องว่างก็ได้



รูปที่ ค.1 ส่วนประกอบสำคัญของประโยคคำสั่ง

คำสั่ง คำสั่งที่ใช้ในกรณีของ Oscilloscope จะแสดงข้อกำหนดของ Function หรือ เปลี่ยนแปลงค่าใดค่าหนึ่งของตัวมัน ซึ่งมีโครงสร้างของคำสั่ง ดังนี้

- [:] <Header>[<Space><Argument>[<comma><Argument>]...]

หัวคำสั่งเป็นการรวมกันของ Mnemonic 1ตัว หรือมากกว่า ซึ่งจะจัดระเบียบเป็นแบบลำดับชั้น หรือ โครงสร้างแบบต้นไม้ Mnemonic ตัวแรกคือพื้นฐานหรือรากของต้นไม้ และแต่ละลำดับถัดไปของ Mnemonic คือระดับหรือกิ่งปีดของระดับชั้นก่อนหน้านั้น คำสั่งที่ระดับสูงขึ้นไปในโครงสร้างแบบต้นไม้นี้ จะส่งผลกระทบต่อคำสั่งระดับต่ำ เครื่องหมาย ":" ที่นำหน้าจะส่งค่ากลับไปยังฐานหรือรากของต้นไม้เสมอ ซึ่งนั่นก็คือ Mnemonic ตัวแรกนั้นเอง

การตรวจสอบ การตรวจสอบในกรณีของ **Oscilloscope** จะต้องมีการส่งกลับข้อมูลเกี่ยวกับและการตั้งค่า

- [:] <Header]?
- [:]<Header>?[<Space><Argument>[<Comma><Argument>]...]

เราสามารถกำหนดรายละเอียดของคำสั่งตรวจสอบได้ที่ทุกระดับชั้น ภายในโครงสร้างแบบต้นไม้เว้นเสียแต่ว่าจะเป็นโครงสร้างสัญลักษณ์แบบอื่น ที่กิ่งของโครงสร้างจะทำการตรวจสอบการส่งกลับข้อมูลเกี่ยวกับ Mnemonic ทั้งหมดที่ระดับเดียวกันหรือต่ำกว่า ตัวอย่างเช่น

DISplay : INTENSITY : CONtast?

เป็นการส่งกลับค่าความแรงจากจุดที่แรงที่สุดของ Waveform ขณะที่ DISplay : INTENSITY? จะเป็นการส่งกลับค่าความแรงที่เราตั้งค่าไว้จากทุกส่วนของตัวแสดงผล

เราสามารถควบคุม Oscilloscope ให้ส่งค่ากลับไปยัง Header ที่ส่วนของผลตอบสนองในการตรวจสอบ ได้สองกรณี โดยใช้ โดยใช้ คำสั่ง HEADer ไปควบคุม ดังนี้ ถ้า Header ทำงานอยู่ Oscilloscope จะทำการส่งส่งคำสั่งกลับไปยัง Header ที่ส่วนของการตรวจสอบและรูปแบบของผลตอบสนองที่ใช้บังคับในคาร Set คำสั่ง และเมื่อ Header ไม่ทำงานหรือหยุดทำงาน Oscilloscope จะส่งค่าทุกค่าในการตอบสอนงกลับไปยัง Header ซึ่งมันจะทำให้ง่ายต่อการกระจายและรวบรวมข้อมูลจากการตอบสอง ตารางที่ ค.3 แสดงความแตกต่างในการตอบสนอง

ตารางที่ ค.3 การเปรียบเทียบระหว่าง Header ที่มีผลตอบสนองกับไม่มี

Query	Header off Response	Header on Response
CURSor: VBArs: DELTa?	1.064E-3	: CURSor: VBArs:DELTa 1.064E-3
ACQuire: NUMAVg?	16	: ACQuire:NUMAVG 16

Clearing the Oscilloscope

ทำการ Clear เอาท์พุท และ Reset Oscilloscope เพื่อรับคำสั่งใหม่ โดยใช้คำสั่ง Device Clear (DCL) GPIB หรือ ใช้สัญญาณ RS-232 BREAK

Command Entry

กฎทั่วไปในการเข้าถึงคำสั่ง

- เราสามารถเข้าถึงคำสั่งที่มีระดับสูงกว่าและในระดับที่ต่ำกว่าได้
- เราสามารถขึ้นต้นคำสั่งด้วย White Space ได้ โดยที่ White Space เป็นการรวมกันของ ASCII Control Characters 00,09 หรือ 20 Decimal
- Oscilloscope จะไม่ทำงานตามคำสั่งที่ประกอบด้วย การรวมกัน White Space และ Line Feeds

คำสั่งย่อ

เราสามารถทำการย่อคำสั่งต่างๆได้ ตัวอย่างเช่น

คำสั่ง ACQuire:NUMAvg

เราสามารถทำการย่อคำสั่งได้ดังนี้

ACQ:NUMA หรือ acq:numa

การเชื่อมต่อคำสั่ง

เราสามารถเชื่อมต่อและรวมคำสั่งในการตั้งค่าและการตรวจสอบโดยใช้เครื่องหมาย (;) และ Oscilloscope จะทำงานตามลำดับที่เชื่อมต่อ

เมื่อเราต้องการเชื่อมต่อคำสั่งเราต้องปฏิบัติตามกฎ ดังนี้

1. Header ที่แตกต่างกันจะถูกจะถูกแยกออกจากกันโดยเครื่องหมาย (;) และจะเริ่มต้นด้วยเครื่องหมาย (:) ทุกคำสั่ง ยกเว้น คำสั่งแรก ตัวอย่างเช่น

คำสั่ง TRIGger:MODe NORMal และ ACQuire : NUMAVg 8

สามารถเชื่อมต่อให้เป็นคำสั่งเดียวกันได้ดังนี้

TRIGger:MODe NORMal ;:Quire : NUMAVg 8

2. ถ้าแต่ละคำสั่งที่ต้องการเชื่อมต่อกันมี Headers ที่ Mnemonic ตัวสุดท้ายแตกต่างกัน เราสามารถย่อคำสั่งที่ 2 และตัดเครื่องหมาย (:) ที่เริ่มต้นทิ้งไปได้ ตัวอย่างเช่น

คำสั่ง ACQuire:MODe ENVelope และ ACQuire:NUMAVg 4

สามารถเชื่อมต่อให้เป็นคำสั่งเดียวกันได้ดังนี้

ACQuire:MODe ENVelope;NUMAVg 4

ซึ่งสามารถทำงานได้ดีเท่ากับการเขียนแบบยาวๆ

ACQuire:MODe ENVelope;: ACQuire:NUMAVg 4

3. เครื่องหมาย (*) จะถูกให้ความสำคัญมากที่สุด

ACQuire:MODe ENVelope ;*TRG

Oscilloscope จะทำงานตามกระบวนการของคำสั่งไปเรื่อยๆ ถ้าคำสั่งนั้นไม่มีเครื่องหมาย (*) ซึ่งจะเป็นไปตามตัวอย่างที่ใช้บังคับดังนี้

ACQuire:MODe ENVelope ;*TRG;NUMAVg 2

4. เมื่อใดที่เราทำการเชื่อมต่อการตรวจสอบ Oscilloscope จะทำการเชื่อมต่อผลตอบสนองของทุกการตรวจสอบเป็นผลตอบสนองเดียวกัน ตัวอย่างเช่น ถ้าความเข้มของการแสดงผลของ text คือ "brigth" และของ waveform คือ "dim" การเชื่อมต่อการตรวจสอบเป็นดังนี้

DISplay:INTENsity:TEXt?;WAVEform?

จะส่งกลับแต่ละส่วน:DISPLAY:INTENSITY:TEXTBRI;:DISPLAY:INTENSITY:WAVEFOM DIM ถ้า Header on หรือ BRI;DIM ถ้า Header off

5. เราสามารถเชื่อมต่อ ตั้งค่าคำสั่งและการตรวจสอบไว้ในประโยคเดียวกัน ดังเช่นตัวอย่างนี้

ACQuire:MODe NORMal;NUMAVg?STATE?

ตัวอย่างการเชื่อมต่อที่ไม่ถูกต้อง

- DISPlay:INTENsity:TEXt BRI;ACQuire:NUMAVg 16
  (ไม่มี (:)หน้า ACQuire)
- DISPlay:INTENsity:TEXt DIM;:WAVEform BRI
  (เครื่องหมาย Extra (:) ก่อน WAVEform สามารถใช้ ISPlay:INTENsity:WAVEform แทนที่ได้)
- DISPlay:INTENsity:TEXt DIM;:*TRG
  (เครื่องหมาย Extra (:) ก่อน เครื่องหมาย *)

การสิ้นสุดคำสั่ง ในคู่มือนี้ใช้ <EOM> (End of Massage) เพื่อแทนการสิ้นสุดคำสั่ง

GPIB End of Massage Terminal GPIB EOM Terminal สามารถเป็นตัวจบคำสั่งได้ (EOI จะเป็นตัวยืนยันพร้อมด้วยข้อมูลไบท์สุดท้าย) รหัส ASCII ของ Line Feed (LF) จะดูตามข้อมูลไบท์สุดท้าย หรือดูทั้งสองอย่าง Oscilloscope จะต้องจบคำสั่งด้วย LF และ EOI เสมอ และจะต้องเว้นช่องว่างก่อนการสิ้นสุดคำสั่งด้วย

การสร้าง Mnemonic บาง Header Mnemonics จะระบุเป็น 1 ในขอบเขตของ Mnemonics ตัวอย่างเช่น Channel Mnemonic สามารถที่จะเป็น CH1 หรือ CH2 ก็ได้เหมือนกัน เราจะใช้ Mnemonic อันใดอันหนึ่งตามคำสั่ง ตัวอย่างเช่น นี่คือคำสั่ง CH1:VOLts และนี่ก็คือ คำสั่ง CH2:VOLts เช่นเดียวกัน ในลักษณะคำสั่งนี้คือการแสดงตัวเลือกเขียนย่อได้ดังนี้ CH<x>

ตำแหน่ง Cursor ของMnemonic เมื่อ Oscilloscope แสดง Cursor คำสั่งจะต้องระบุที่ Corsor ด้วยข้อมูลทั้งคู่ที่นำไปใช้งาน

สัญลักษณ์	ความหมาย
ตำแหน่ง<x>	Cursor ให้เลือกที่ตำแหน่ง <x> เป็น 1หรือ 2

การระบุเครื่องวัดให้ Mnemonic คำสั่งสามารถระบุ เครื่องมือวัดที่เราต้องการตั้งค่าหรือตรวจสอบที่ Mnemonic ใน Header Oscilloscope สามารถแสดงเครื่องมือวัดอัตโนมัติได้ 4 ตัว ด้วยการแสดล Waveform ของแต่ละตัว การแสดงเครื่องมือวัดสามารถระบุได้ในวิธีนี้

สัญลักษณ์	ความหมาย
MEAS<x>	ระบุเครื่องมือวัด;<x>เป็น 1,2,3หรือ4

Channel Mnemonic คำสั่งระบุ Channel ที่ใช้งานที่ Mnemonic ใน Header

สัญลักษณ์	ความหมาย
CH<x>	ระบุ Channel;<x>เป็น 1 หรือ 2

Math Waveform Mnemonic คำสั่งสามารถระบุ Waveform ในทางคณิตศาสตร์ที่ใช้งานได้ที่ Mnemonic ใน Header

สัญลักษณ์	ความหมาย
MATH<x>	ระบุ Math Waveform;<x>เป็น 1

Reference Waveform Mnemonic คำสั่งสามารถระบุ Reference Waveform ที่ใช้งานได้ที่ Mnemonic ใน Header

สัญลักษณ์	ความหมาย
REF<x>	ระบุ Reference Waveform;<x>เป็น 1 หรือ 2

Waveform Mnemonic ในบางคำสั่งเราสามารถระบุ Waveform คือ Channel Mnemonic, Math Waveform หรือ Reference Waveform สามารถระบุได้ตามนี้

สัญลักษณ์	ความหมาย
<wfm>	สามารถระบุได้เป็นCH<x>,MATH<x>,MATH12<X> หรือ REF<x>

ชนิดของ Argument คำสั่ง Argument สามารถเป็น 1ในหลายๆ รูปแบบ แต่ละลักษณะของคำสั่ง แต่ละลักษณะของ Argument แต่ละชนิดสามารถใช้งานได้กับคำสั่งนั้นๆ

Numeric Argument ส่วนมากแล้วคำสั่งของ Oscilloscope ต้องการ Numeric Argument ในคู่มือนี้จะเสนอ Argument ตามตารางข้างล่างนี้

สัญลักษณ์	ความหมาย
<NR1>	สัญลักษณ์แสดงค่าของเลขจำนวนเต็ม
<NR2>	ค่าของเลขทศนิยมที่ไม่มีเลขยกกำลัง
<NR3>	ค่าของเลขทศนิยมที่มีเลขยกกำลัง

Quoted String Argument บางคำสั่งจะรับหรือส่งกลับข้อมูลในรูปแบบของการอ้างข้อความ ซึ่งมันจะทำให้ง่ายขึ้น กลุ่มของ อักขระ ASCII ส่งโดยเครื่องหมาย (') (Single Quote) หรือ เครื่องหมาย (") (Double quote) ตัวอย่างเช่น "This is a quote string"

สัญลักษณ์	ความหมาย
<Qstring>	การอ้างถึงข้อความของ ข้อความ ASCII

เมื่อจะใช้ Quote String เราต้องปฏิบัติตากฎดังต่อไปนี้

- Quote String สามารถเรียกใช้อักขระได้ตามที่กำหนดไว้ในอักขระ ASCII 7 bit
- ต้องใช้เครื่องหมาย Quote Character ("-") หรือ ('-') ชนิดเดียวกันในการเปิดและ ปิดข้อความ เช่น "This is a valid string"

- เราสามารถใช้เครื่องหมายทั้งสองชนิดร่วมกันได้ในข้อความที่ยาวมากได้ เราสามารถทำได้ดังนี้

  "This is an 'acceptable' string"
- เราสามารถเรียกใช้ Quote character เพื่อทำให้ข้อความง่ายขึ้น โดยการใช้ Quote character ซ้ำๆ กัน ตัวอย่างเช่น "Here is a" "mark"
- ข้อความสามารถมี upper หรือ lower Case Character
- เราไม่สามารถที่จะจบข้อความด้วย END Massage ได้ก่อนปิดเขตข้อความ
- ความยาวสูงสุดของข้อความที่ส่งกลับจากการตรวจสอบคือ 1000 อักขระ

ตัวอย่างข้อความที่ไม่ถูกต้อง

"Invalid strings Argument'

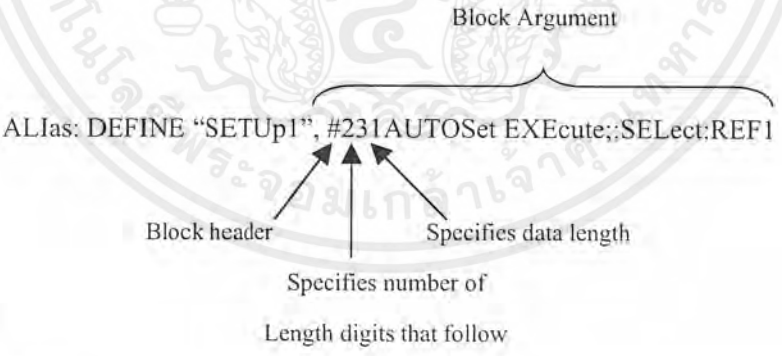
(เครื่องหมาย Quote ต่างชนิดกัน)

"Test<EOI>"

(EOI ไม่สามารถใช้ในการสิ้นสุดข้อความได้)

Block Argument คำสั่งหลายคำสั่งใช้ block Argument

ด้วย พอร์ต GPIB, เส้นสัญญาณ EOI ในไบท์สุดท้าย รูปที่ ค. 2 แสดงตัวอย่างของ Block Argument



Block header = #

Specifies data length = **31**

Specifies number of Length digits that follow = **2**

Block Argument = **#231AUTOSet EXEcute;:SELect:REF1**

บรรณานุกรม

คมกฤษณ์ ชูเรือง,แวฮาซัน แวหะมะ และ อนุชา หาญสมบัติเจริญ. 2541. "การควบคุมอุปกรณ์ GPIB ผ่านเครือข่ายอินเตอร์เน็ต." ปริญญานิพนธ์วิศวกรรมศาสตร์บัณฑิต สาขาวิศวกรรมอิเล็กทรอนิกส์, สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง.

อำพล ทองระอา. 2541. "เอกสารประกอบคำสอนเรื่อง การใช้งานโปรแกรม LabVIEW." สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง.

Leonard Sokoloff. **Basic Concept of LabVIEW™ 4.** New Jersy: Prentice-Hall, Inc. 1998

National Instrument. **LabVIEW User Manual**. Texas: 1998

Tektronix Product. **Programmer Manual TDS 340A, TDS 360 & TDS 380 Digital Real Time Oscilloscopes.**

National Instrument. "DataSocket."[Online]. Avaible: http://www.ni.com. 1999.

ประวัติผู้แต่ง

ชื่อผู้ทำปริญญานิพนธ์	นายจุมพล เยรบุตร
วันเดือนปีเกิด	7 กันยายน 2521
สถานที่เกิด	โรงพยาบาลยโสธร
ภูมิลำเนาเดิม	652/654 ถ.แจ้งสนิท ต.ในเมือง อ.เมือง จ. ยโสธร 35000
โทรศัพท์	045-712085
ที่อยู่ปัจจุบัน	13/10 ซอย เกกีงาม หมู่ 3 ถ.ฉลองกรุง แขวงลำปาทิว เขตลาดกระบัง กรุงเทพฯ 10520
โทรศัพท์	02-3266128 ห้อง 14K

ประวัติการศึกษา

ประถมศึกษา	โรงเรียนเทศบาล 2 สามัคคีวิทยา
มัธยมศึกษาตอนต้น	โรงเรียนยโสธรพิทยาลัย
ประกาศนียบัตรวิชาชีพ (ปวช.)	วิทยาลัยเทคนิคยโสธร
ประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.)	วิทยาลัยเทคนิคยโสธร
ปริญญาตรี	สาขาอิเล็กทรอนิกส์และคอมพิวเตอร์ ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง
ผลงานที่ได้รับรางวัล	-
คติพจน์	-

ประวัติผู้แต่ง

ชื่อผู้ทำปริญญานิพนธ์	นางสาวอรนุช สัมฤทธิ์
วันเดือนปีเกิด	1 กรกฎาคม 2520
สถานที่เกิด	โรงพยาบาลเจ้าพระยาอภัยภูเบศร จ. ปราจีนบุรี
ภูมิลำเนาเดิม	13/1 หมู่ 2 ต.บางบริบูรณ์ อ.เมือง จ.ปราจีนบุรี 2500
โทรศัพท์	01-6571319
ที่อยู่ปัจจุบัน	13/1 หมู่ 2 ต.บางบริบูรณ์ อ.เมือง จ.ปราจีนบุรี 2500
โทรศัพท์	01-6571319

ประวัติการศึกษา

ประถมศึกษา	โรงเรียนวัดสง่างาม สุทธิเกียรติวิทยา
มัธยมศึกษาตอนต้น	โรงเรียนปราจีนกัลยาณี
ประกาศนียบัตรวิชาชีพ (ปวช.)	วิทยาลัยเทคนิคปราจีนบุรี
ประกาศนียบัตรวิชาชีพชั้นสูง (ปวส.)	วิทยาลัยเทคนิคปราจีนบุรี
ปริญญาตรี	สาขาอิเล็กทรอนิกส์และคอมพิวเตอร์ ภาควิชาครุศาสตร์วิศวกรรม คณะครุศาสตร์อุตสาหกรรม สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหาร ลาดกระบัง
ผลงานที่ได้รับรางวัล	-
คติพจน์	ไม่เสียใจกับผลที่ได้รับถ้าเราพยายามทำดีที่สุดแล้ว