การประยุกต์ใช้เครื่องคอมพิวเตอร์ส่วนตัวให้เป็น DCS

COMPUTER APPLICATION TO DCS

โดย

นายณรงศักดิ์ ยอดคีรี 36.013284

นายสมชาย เหรียญเงิน 36.013309

ปริญญานิพนธ์สำหรับวิศวกรรมศาสตรบัณฑิต

สาขาวิศวกรรมการวัดคุมทางอุตสาหกรรม

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้า เจ้าคุณทหาร ลาดกระบัง

ปีการศึกษา 2538

ภาควิชา : เทคโนโลยีการวัดคุมทางอุตสาหกรรม

สาขาวิชา : วิศวกรรมการวัดคุมทางอุตสาหกรรม

คณะ : วิศวกรรมศาสตร์

สถาบัน : เทคโนโลยีพระจอมเกล้า เจ้าคุณทหาร ลาดกระบัง

เรื่อง : การประยุกต์ใช้เครื่องคอมพิวเตอร์ส่วนตัวให้เป็น DCS

COMPUTER APPLICATION TO DCS

จัดทำโดย : 1. นายณรงศักดิ์ ยอดคีรี 36.013284

2. นายสมชาย เหรียญเงิน 36.013309

..... อาจารย์ที่ปรึกษา

(ผู้ช่วยศาสตราจารย์ ธนิตย์ ตรีสุวรรณวัฒน์)

หัวข้อปริญญานิพนธ์ : การประยุกต์ใช้เครื่องคอมพิวเตอร์ส่วนตัวให้เป็น DCS
นักศึกษา : นายณรงศักดิ์ ยอดคีรี 36.013284
นายสมชาย เหรียญเงิน 36.013309
อาจารย์ที่ปรึกษา : ผศ. ธนิตย์ ตรีสุวรรณวัฒน์
ระดับการศึกษา : วิศวกรรมศาตรบัณฑิตทางเทคโนโลยีการวัดคุมทางอุตสาหกรรม
ปีการศึกษา : พ.ศ. 2538

บทคัดย่อ

โครงงานนี้เป็นการนำคอมพิวเตอร์ที่ใช้อยู่ในท้องตลาดในปัจจุบันมาพัฒนาให้สามารถทำงานได้ทัดเทียมกับ DCS โดยการใช้โปรแกรมสำเร็จรูปที่มีใช้อยู่คือ FIX DMACS และ FIX MMI โดยเขียนโปรแกรมให้สามารถทดลองกับระบบควบคุมแบบต่าง ๆ และเขียนโปรแกรมในการ LINK อุปกรณ์ในการควบคุม (Controller) เขียนโปรแกรมสำหรับอุปกรณ์ในการควบคุม (Controller) รวมทั้งรูป PROCESS ที่เราจะทำการทดลอง เพื่อที่จะดูถึงสภาวะของการควบคุมแบบต่าง ๆ ซึ่งโปรแกรมที่ได้ทำการศึกษาสามารถที่จะนำไปใช้งานได้จริงในระบบควบคุมในโรงงานอุตสาหกรรมได้

ABTRACT

THE PROJECT IS DEVELOPMENT MICRO COMPUTER APPROXIMATE TO DCS BY USE PACKAGE SOFTWARE FIX DMACS AND FIX MMI. THIS PROGRAM ARE USE TEST WITH OTHER CONTROL SYSTEM AND LINK CONTROL EQUIPMENT , CONTROLLER AND PROCESS AS TO TESTING FOR LOOKING OTHER STATUS CONTROL. THIS PROGRAM AS STUDY ABLE TO CONTROL SYSTEM IN INDUSTRIAL

สารบัญ

บทนำ

ในปัจจุบันการพัฒนาทางด้านเทคโนโลยีของโรงงานอุตสาหกรรมจะมุ่งเข้าสู่การใช้ระบบควบคุมอัตโนมัติประเภท PLC หรือ DCS ไม่ว่าจะเป็นโรงงานอุตสาหกรรมขนาดใหญ่ หรือ โรงงานอุตสาหกรรมขนาดปานกลาง จำเป็นจะต้องมีระบบคอมพิวเตอร์เข้ามาเกี่ยวข้องเพื่อติดต่อสื่อสารข้อมูลกับอุปกรณ์และเครื่องจักรในโรงงานอุตสาหกรรมโดยในระบบ PLC หรือ DCS มักจะมีอุปกรณ์เสริมที่สามารถสื่อสารข้อมูลที่รับได้ไปยังระบบคอมพิวเตอร์ภายนอก โดยทั่วไปมักจะเป็น Interface Port PLC หรือ DCS กับ Interface Port ประเภทเดียวกันบนระบบคอมพิวเตอร์ภายนอกได้ ทั้งนี้ข้อมูลที่สื่อสารมานั้นต้องอยู่ภายใต้การกำกับของ I/O Driver บนระบบคอมพิวเตอร์ที่ต่ออยู่ด้วย I/o Driver เป็นซอฟท์แวร์พิเศษที่ทำหน้าที่กำกับการอ่านที่รับเข้ามาทาง Interface Port ข้อมูลที่อ่านได้นี้จะถูกต้องหรือไม่ขึ้นอยู่กับ I/o Driver นี้เมื่อได้ข้อมูลเหล่านี้มาใช้ประโยชน์เช่น การแสดงผล สร้างสัญญาณเตือนเก็บบันทึกบนฐานข้อมูล พิมพ์รายงานผลการผลิต และเชื่อมต่อไปยังระบบอื่น ๆ เป็นต้น และตัวที่จะทำที่นี้คือ Application Software บนระบบคอมพิวเตอร์ Application Software นี้สามารถพัฒนาขึ้นมาเองโดยผู้ใช้หรือจัดหามาใช้งานในรูปของซอฟท์แวร์สำเร็จรูป

ในที่นี้เราได้ทำการศึกษาซอฟท์แวร์สำเร็จรูปคือ FIX DMACS ของบริษัท INTELLUTION C0,LTO.ซึ่งเป็นซอฟท์แวร์สำเร็จรูปประเภท SCADA ที่ทำงานบน PC ซึ่งสามารถทำหน้าที่ของ I/O Driver และระบบตามผลข้อมูลอย่างอัตโนมัติ

วัตถุประสงค์

1. เพื่อทำการศึกษาถึงโปรแกรมการทำงานของ FIX DMACS และ FIX MMI
2. เพื่อศึกษาถึงอุปกรณ์ INSTRMENT ที่มีใช้งานจริงในระบบอุตสาหกรรมในที่นี้ได้ทำการศึกษาถึง CONTROLLER และวิธีการเขียนโปรแกรมในการคอนโทรลของ CONTROLLER รุ่น EC-320
3. เพื่อทำการศึกษาถึงวิธีการ LINK ข้อมูลระหว่างโปรแกรม FIX กับ CONTROLLER
4. ทำการศึกษาเกี่ยวกับโปรแกรมทางด้าน I/O DRIVER
5. เพื่อทำการศึกษาถึงรูปแบบการ CONTROL แบบต่าง ๆ

ขอบเขตโครงงาน

1. ทำการเขียนโปรแกรมของ FIX
   คือ -เขียนรูปของ PROCESS ที่จะทำการทดลอง
   -เขียน DATA BASE (วอฟท์แวร์กำหนดฐานข้อมูล)
   -ทำการกำหนดค่าในการทำงานของ-HISTORYCAL TREND
   -ALARM
   -ทำการ LINK SOFTWARE ของ FIX กับ CONTROLLER
2. เขียนโปรแกรมการทำงานของ CONTROLLER
3. ทดลองใช้โปรแกรม FIX กับระบบควบคุมต่าง ๆ เพื่อดูถึงสภาวะการทำงานของการควบคุม

โครงสร้างของ FIX DMACS ประกอบด้วย

-I/O Driver

-ฐานข้อมูล SCADA

-MMI (MAN-MACHINE-INTERFACE)

โดยข้อมูลจากโรงงานถูกอ่านจากอุปกรณ์วัดควบคุม หรือ I/O DRIVER เช่น CONTROLLER ผ่านมาทาง I/O DRIVER ข้อมูลที่อ่านได้ถูกเก็บไว้ในฐานข้อมูล SCADA เพื่อใช้ในการควบคุมการผลิตอย่างอัตโนมัติโดยผู้ควบคุมการผลิตสามารถมองเห็นข้อมูลการผลิตเหล่านี้จากภาพและตัวเลขที่ปรากฏบนจอภาพของ PC โดยภาพและตัวเลขเหล่านี้สามารถเคลื่อนไหวเปลี่ยนแปลงตามค่าที่วัดได้จริงแบบ REAL TIME FIX DMACS โดยมีฟังก์ชั่นทางด้าน MMI ที่ให้ผู้ใช้สามารถกำหนด APPLICATION การทำงานให้กับผู้ควบคุมการผลิตในรูปแบบต่าง ๆ ฟังก์ชั่นที่มีให้เลือกใช้ได้แก่

-การควบคุมแบบ SPC (STATISTICAL PROCESS CONTRLOR)

-การควบคุมการผลิตแบบ BATCH ด้วยสูตรผสม (RECIPE MANAGEMENT)

-การเก็บบันทึกประวัติข้อมูล (HISTORICAL TRENDING)

-การสร้างสัญญาณเตือน (ALARMING)

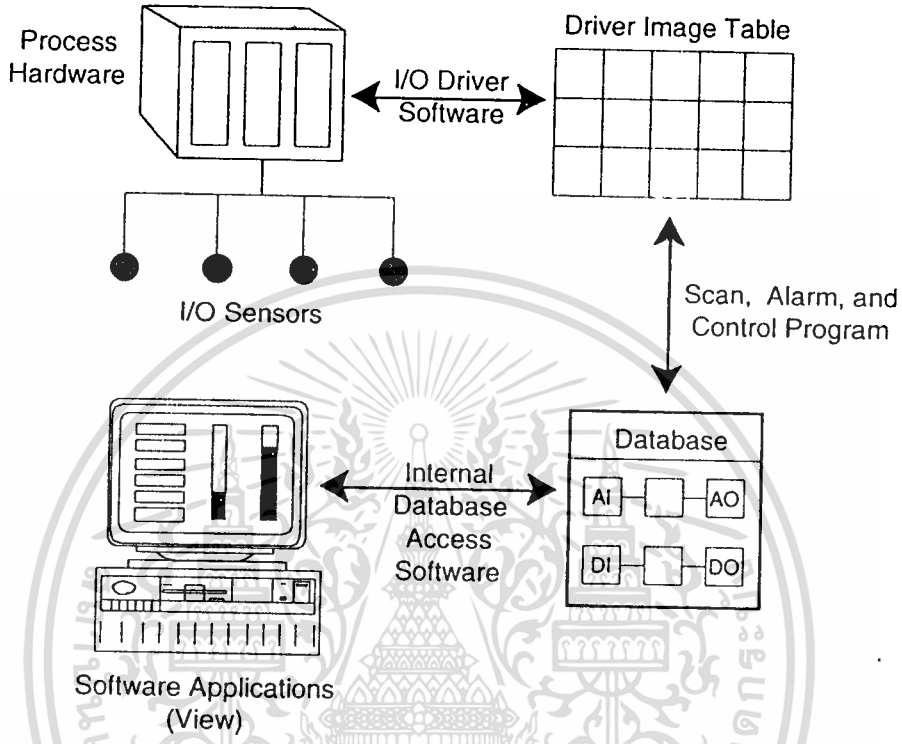
-การออกรายงานผลการผลิต (REPORT GENERATION)

-การสื่อสารข้อมูลระยะไกลผ่านโมเด็ม (REMOTE)

-การรักษาความปลอดภัย (SECURITY)

โดยเราสามารถติดตั้งจากระบบเล็กที่มี NODE เดียว (PC ตัวเดียว) และขยายเป็นหลาย ๆ NODE "ได้ในภายหลัง

หลักการทำงานของ FIX



โดยจะมี PROCESS HARDWARE รับสัญญาณจาก INPUT ภายนอกมาเก็บไว้ และ I/O DRIVER จะดึง PROGRAM จาก PROCESS HARDWARE มาเก็บไว้ที่ DIT (DRIVER IMAGE TABLE) และ SAC ก็จะทำการแลกเปลี่ยนข้อมูลโดยทำหน้าที่ 3 หน้าที่คือ -SCAN

-ALARM

-CONTROL

โดยนำข้อมูลไปเก็บไว้ที่ DATABASE โดย DATABASE จะเป็นตัวกำหนด การทำงานของขบวนการของงาน จากรูปด้านบน I/O DRIVER จะดึงข้อมูลไม่เป็นจังหวะไม่สามารถคอนโทรลได้จึงต้องดึงข้อมูลเก็บไว้ที่ DIT และ SAC ที่สามารถดึงข้อมูลได้เป็นจังหวะก็จะดึงข้อมูลไปเก็บไว้ที่ DATABASE อีกที

โครงสร้างการทำงานของ FIX DMACS จะประกอบด้วยซอฟต์แวร์หลายตัวที่มีหน้าที่ต่างกัน โดยทำงานแยก TASK กันบนวินโดว์ซอฟต์แวร์โดยซอฟต์แวร์ของ FIX DMACS ที่สำคัญประกอบด้วย

ซอฟต์แวร์ I/O DRIVER

ซึ่งทำหน้าที่แลกเปลี่ยนข้อมูลหรือติดต่อกับอุปกรณ์รับส่งสัญญาณผ่านสายสัญญาณ เช่น RS-232 ข้อมูลที่ได้รับจะถูกเก็บไว้อยู่ในหน่วยความจำเพื่อให้ซอฟต์แวร์อื่นนำไปใช้งานต่อไป โดยที่ในขณะเดียวกันซอฟต์แวร์ตัวอื่นก็จะนำเอาข้อมูลมาไว้ที่หน่วยความจำเดียวกันนี้เพื่อให้ I/O DRIVER นำไปส่งให้อุปกรณ์รับส่งสัญญาณในกรณีที่เป็นข้อมูล OUTPUT

ซอฟต์แวร์ I/O DRIVER มีหลายแบบขึ้นอยู่กับชนิดและยี่ห้อของอุปกรณ์รับส่งสัญญาณซึ่งมี PROTOCAL ที่จะใช้แปลงสัญญาณต่างกัน โดยในการศึกษาครั้งนี้เราได้ศึกษา I/O DRIVER ของ EC 300 SERIE CONTROLLER (EC 375, ECBUS DRIVER FER FIX DMACS FOR WINDOWS)

ซอฟต์แวร์ควบคุมการรับส่งข้อมูล

ซึ่งทำหน้าที่ควบคุมอัตราการรับส่งข้อมูลตามที่ผู้ให้กำหนด เนื่องจากซอฟต์แวร์ I/O DRIVER ทำหน้าที่แลกเปลี่ยนข้อมูลกับอุปกรณ์รับส่งสัญญาณอย่างเดียวในอัตราการรับส่งข้อมูลที่ไม่เป็นระเบียบ ดังนั้นจึงจำเป็นต้องมีซอฟต์แวร์ควบคุมการรับส่งข้อมูล ที่จะทำการควบคุมอัตราการรับส่งข้อมูลให้เป็นระเบียบโดยการด่วนค่าที่ได้จาก I/O DRIVER มาเก็บไว้ในฐานข้อมูลที่ผู้ใช้ได้กำหนดไว้และนำค่าจากฐานข้อมูลไป UPDATE ให้กับ I/O DRIVER ในขณะเดียวกันซอฟต์แวร์นี้จะทำหน้าที่ตรวจจับและส่งสัญญาณเตือนเมื่อข้อมูลไม่ได้อยู่ในเกณฑ์ที่กำหนดไว้สำหรับการควบคุม เช่นการแสดงสัญญาณเตือนว่าข้อมูลสูงเกินกว่าค่าที่กำหนดไว้สำหรับการควบคุม (HIGH ALARM) หรือการแสดงสัญญาณเตือนว่าข้อมูลต่ำกว่าค่าที่กำหนดไว้สำหรับการควบคุม (LOW ALARM) รวมถึงการสร้างสัญญาณเตือนในกรณีที่อุปกรณ์รับส่งสัญญาณขาดการติดต่อสื่อสารกับ PC

ซอฟต์แวร์กำหนดฐานข้อมูล (DATABASE CONFIGURATION)

เนื่องจากข้อมูลที่ผู้ใช้ต้องการจากอุปกรณ์รับส่งสัญญาณนั้นจะต้องอ้างอิงถึงตำแหน่งของหน่วยความจำบนอุปกรณ์รับส่งสัญญาณ จึงเป็นการยากต่อผู้ใช้งานในการใช้งานปกติซอฟต์แวร์ตัวนี้จึงให้ผู้ใช้สามารถกำหนดตำแหน่งอ้างอิงให้อยู่ในรูปของชื่อประจำตัว (TAG) โดยสามารถตั้งชื่อเป็นอักษรที่มีความหมายและจดจำได้ง่าย นอกจากนี้ผู้ใช้สามารถกำหนดความต้องการในการใช้งานข้อมูลได้หลายรูปแบบ เช่น กำหนดช่วงเวลาการ SCAN ของข้อมูลได้ผ่านการคำนวณทางคณิตศาสตร์ก่อนไปใช้งานอย่างอื่น กำหนด LOGIC การใช้งานข้อมูลตามเงื่อนไขหรือตามเวลา กำหนดให้มีการตรวจจับสัญญาณที่ระดับต่าง ๆ กำหนดการป้องกันการใช้ข้อมูลโดยไม่ได้รับอนุญาตที่ระดับต่าง ๆ เป็นต้น

	Tag Name	Type	Scan Time	Description
1	AI1	AI	1	Pump Value 1
2	AI13	AI	2	Pump Value 2
3	AI14	AI	19	Pump Value 3
4	AO10	AO	----	Pump Value 4
5	AO11	AO	----	Pump Value 5
6	AO12	AO	----	Pump Value 6
7				

รูปตัวอย่าง ของ DATABASE BUILDER

โดยหน้าที่ของ DATABASE จะทำหน้าที่เก็บ BLOCK แต่ละ BLOCK และ APPLY แต่ละ BLOCK ว่ามีหน้าที่การทำงานอย่างไร เช่น

- CALIBRATE ค่า โดยการทำงานส่วนใหญ่จะทำงานที่ 0-100 % แต่ค่าที่วัดได้คือ 300-1000 (ENGINEERING UNIT) DATABASE ก็จะ CALIBRATE ค่าให้

-ตั้งค่า LIMIT ALARM โดยทำงานร่วมกับ SAC

-ในส่วนของการคำนวณค่า

-ฯลฯ

โดยการนำ BLOCK แต่ละ BLOCK มา CHAIN เข้าด้วยกัน โดย SAC จะเป็นตัวควบคุมทั้งหมด

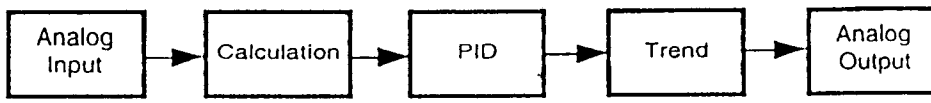
BLOCK จะแบ่งออกเป็น 5 กลุ่ม หลัก

1. STANDARD BLOCK
2. CONTROL BLOCK
3. BATCH BLOCK
4. SPC (STATISTICAL PROCESS CONTROL)
5. SQL BLOCK

ซึ่ง BLOCK ยังแบ่งออกเป็น 2 แบบคือ

-PRIMARY BLOCK จะรับส่งข้อมูลระหว่าง DATABASE กับ DIT หรือ พวก I/O ทั้งหมด และจะมี SCAN TIME ที่สามารถกำหนดได้

-SECONDARY BLOCK ทำหน้าที่ CHAIN กับ PRIMARY BLOCK จะไม่มี SCAN TIME โดย SCAN TIME จะขึ้นอยู่กับ PRIMARY BLOCK โดยการ CHAIN จะ CHAIN ดังรูป



โดย PRIMARY BLOCK จะประกอบด้วย BLOCK ต่าง ๆ ดังนี้

Block:	Function:
Analog Alarm	Retrieves analog data from the Driver Image Table every time the block is scanned and provides the ability to set and acknowledge alarms.
Analog Input	Retrieves analog data from the Driver Image Table every time the block is scanned and provides the ability to set alarm limits.
Analog Output	Sends analog data to the Driver Image Table when the upstream block passes a value or when an operator, Program block, or Easy Database Access (EDA) program sets it.
Analog Register	Provides read/write access to the Driver Image Table only when a link connected to the block is viewed on an operator display.
Boolean	Performs boolean calculations based upon up to eight inputs.
Digital Alarm	Retrieves digital data from the Driver Image Table every time the block is scanned and provides the ability to set and acknowledge alarms.
Digital Input	Retrieves digital data from the Driver Image Table every time the block is scanned and provides the ability to set alarm limits.
Digital Output	Sends digital data to the Driver Image Table when the upstream block passes a value, or when an operator, Program block or EDA program sets it.
Digital Register	Provides read/write access to the Driver Image Table only when a link connected to the block is viewed on an operator display.
Multistate Digital Input	Retrieves digital data for up to 3 inputs from the Driver Image Table every time the block is scanned, combines the inputs into one raw value, and provides the ability to set alarm limits.
Text	Provides the ability to read/write a device's text information.

โดย SECONDARY BLOCK จะประกอบด้วย BLOCK

Block:	Function:
Calculation	Performs mathematical calculations using values from the upstream block and up to seven other constants or block values.
Event Action	Monitors values or alarm conditions of the upstream block and performs actions based on the upstream block's output.
Fanout	Passes the value it receives from its upstream block to up to four additional blocks.
Signal Select	Samples up to six inputs, manipulating the inputs according to a user-selected mode, and outputs a value to the next block.
Timer	Functions as a time counter by incrementing or decrementing its value.
Totalizer	Maintains a floating point total for values passed to it from upstream blocks. This block passes values up to six digits in precision to other blocks. It can display up to fifteen digits of precision in View.
Trend	Allows you to collect up to 80 real-time values from an upstream block. Data can be displayed as a graph in View.

CONTROL BLOCK

จะเป็น BLOCK ที่จะกำหนดการควบคุมว่าจะเป็นแบบต่อเนื่อง , โดยตรงหรือ DIGITAL CONTROL โดยมี BLOCK ดังนี้

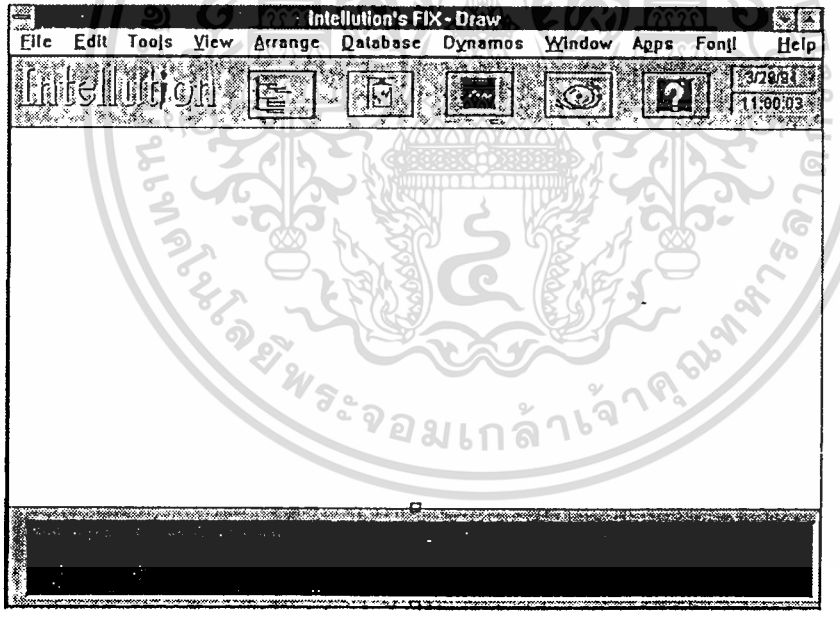
Block:	Function:
Dead Time	Delays up to 255 seconds the transfer of an input value to the next block in the chain. It can store up to 60 values of incoming variables and passes values on a first in/first out basis.
Lead Lag	Provides the ability to simulate process dynamics and includes a digital approximation of the exponential equations for lead lag. This block is useful in feed-forward strategies.
PID	Compares analog inputs to a user-defined setpoint and sends out incremental adjustments to bring the process variable closer to the setpoint.
On-Off Control	Receives analog values and outputs digital values.
Ramp	Increases or decreases values to a target value at a specified rate. Target values can be entered manually or retrieved from other blocks. Three distinct stages can be defined for the ramp process.
Ratio/Bias	Provides the ability to change incoming signals by adding a constant (bias) and/or by multiplying a constant (ratio) after subtracting an offset from the signal. This block uses less memory and executes faster than the Calculation block.

โดย BLOCK ที่สำคัญในการควบคุมที่จะใช้กันบ่อยมากก็คือ-PRIMARY BLOCK, SECONDARY BLOCK,CONTROL BLOCK

ส่วน BLOCK ที่ไม่ได้กล่าวถึงมักจะใช้ในงานควบคุมเฉพาะอย่างที่สามารถนำบล็อคที่ไม่ได้กล่าวถึงไปใช้งานได้

ซอฟต์แวร์สร้างภาพ

ผู้ใช้สามารถกำหนดให้มีการแสดงผลข้อมูลบนจอภาพเป็นรูปหรือตัวอักษรแบบต่าง ๆ ได้โดยไม่ต้องเขียน โปรแกรมซอฟต์แวร์นี้โดยผู้ใช้สามารถกำหนดว่ารูปหรือตัวอักษรที่ต้องการให้แสดงผลนั้นอ้างอิงถึง TAG ไหน ในฐานข้อมูลได้กำหนดไว้แล้วในซอฟต์แวร์กำหนดฐานข้อมูล (DATABASE BUILDER) รูปหรือตัวอักษรที่อ้างอิงจะแสดงผลตามข้อมูลที่ได้รับจริงจากอุปกรณ์รับส่งสัญญาณตามอัตราการ SCAN ที่กำหนดไว้เพื่อให้เห็นภาพของการทำงานได้ โดยหลักการสร้างสิ่งต่าง ๆ เหล่านั้นเป็นไปตามวัตถุประสงค์ตามความต้องการของผู้ใช้ โดยผู้ใช้สามารถนำมาประกอบต่อเนื่องกันเป็นระบบได้และสามารถสร้างหน้าปัดของการควบคุมระบบตามความต้องการได้ นอกจากนี้ผู้ใช้สามารถกำหนดให้ภาพที่สร้างขึ้นมีการรับข้อมูลผ่านจากแป้นพิมพ์จากผู้ปฏิบัติงานตามเงื่อนไขหรือเวลาใด ๆ ได้



รูปตัวอย่างของ ซอฟต์แวร์สร้างภาพ

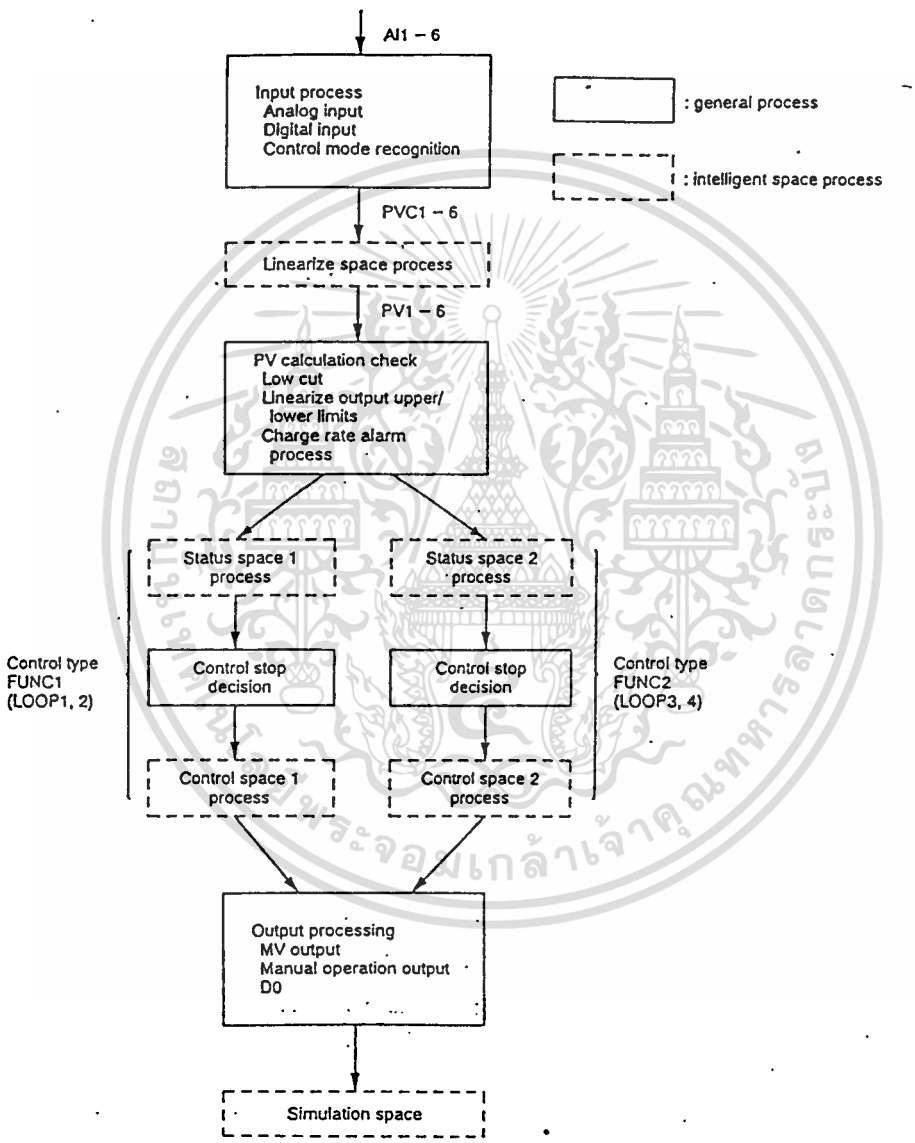
ซอฟต์แวร์แสดงผล

ทำหน้าที่แสดงผลตามรูปภาพ ที่ผู้ใช้กำหนดในซอฟต์แวร์สร้างภายนอกเหนือไปจากซอฟต์แวร์หลัก ๆ แล้วยังมีซอฟต์แวร์อื่น ๆ อีกเช่น ซอฟต์แวร์ที่ทำหน้าที่แสดงผลของ ALARM บนหน้าจอ บนเครื่องพิมพ์ และบนหน่วยความจำ เมื่อมีการตรวจจับสัญญาณเตือนได้ หรือ ซอฟต์แวร์ที่ทำหน้าที่รับส่งข้อมูลที่รับส่งข้องมูลระหว่าง APPLICATION บนวินโดว์แบบ DDE ตามมาตรฐานของ MICROCOFT เป็นต้น

เครื่องควบคุม EC 320

1. ลักษณะการทำงานทั่วไป

การทำงานของโปรแกรมสามารถนำไปประยุกต์ใช้งานควบคุมในขบวนการผลิตต่าง ๆ ได้ โดยมีฟังก์ชั่น ให้เลือกมากมายสำหรับเครื่องควบคุมและสามารถแสดงการทำงานได้ ดังผังงาน (FLOW CHART)



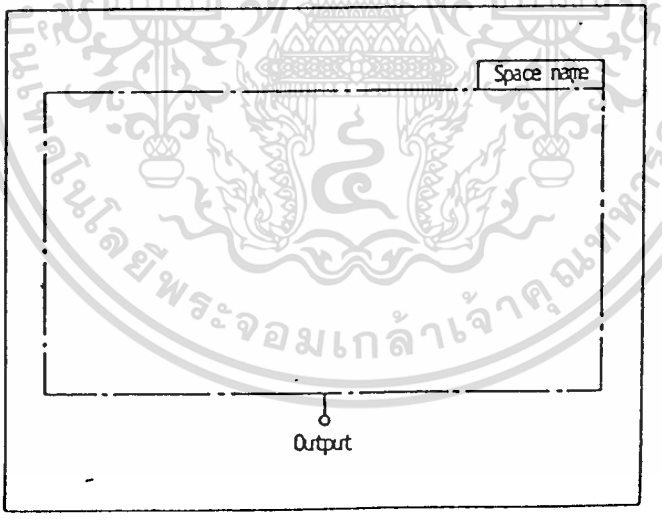
GENERAL PROCESS FLOW CHART

จากผังการทำงานจะสังเกตได้ว่ามีบล็อคอยู่ 2 แบบ คือ แบบเส้นทึบและเส้นปะ (DOT) ซึ่งการทำงานจะแตกต่างกันออกไปคือ ส่านที่เป็นเส้นทึบ การทำงานของบล็อคจะทำงานในส่วนหลักของโปรแกรมและจะทำงานในเวลาที่แน่นอน ส่วนของการทำงานของบล็อคเส้นปะ จะทำงานในส่วนของผู้เขียนโปรแกรต้องการให้เครื่องควบคุมทำงานอย่างไรหรือเรียกว่า INTELLIGENCE SPACE ซึ่งมีฟังก์ชั่นบล็อคให้เลือกได้มากถึง 200 ฟังก์ชั่นโมดุล

เครื่องควบคุมรุ่น EC - 320 จะพิจารณา 2 ลูฟคอนโทรลเท่านั้นซึ่งจะแยกในส่วนของคอนโทรลสเปซ (CONTROL SPACE) เป็น 2 ส่วนโดยอิสระต่อกันดังรูปของผังการทำงานและในส่วนของการคำนวณสามารถคำนวณใช้งานได้ 4 ลูฟ (PID) ซึ่งแต่ละลูฟจะมีเอาท์พุทเป็น MV 1, MV 2, MV 3, MV 4 ตามลำดับและสำหรับการต่อออกไปใช้งานจะระบุเป็น 2 จุดเท่านั้น คือ MV 1กับ MV 3

INTELLINGENT SPACE

ในเครื่องควบคุมรุ่นนี้จะมีสเปซอยู่ 6 สเปซ ซึ่งแต่ละสเปซจะทำงานแตกต่างกันออกไป ซึ่งสเปซมีลักษณะดังรูป



ฟังก์ชั่นโมดุลของ INTELLINGENT SPACE

ฟังก์ชั่นโมดุล ภายในเครื่องควบคุมแบบ 2 ลูฟ มีให้เลือกมากมายเช่น การบวก,ลบวงจรลอจิก, PID เป็นต้น นอจากนี้ยังนำมา LINK กันได้อย่างอิสระ โดยที่ฟังก์ชั่นโมดุลจะเรียกกันแบบอนุกรมเป็นอันดับจากบล็อค 1 (B1) และบล็อค 2 (B2) และต่อไปเรื่อย ๆ ส่วนการเขียนหรือการถ่ายโปรแกรม สามารถทำได้ โดย 2 วิธี

1. เขียนข้อมูลหรือโปรแกรม ไปที่ EEPROM โดยใช้ร่วมกับLOADER รุ่น CE - 394 (TOSHIBA)

2. เขียนข้อมูลหรือโปรแกรม ไปที่ EEPROM โดยการอ่านข้อมูลมาจากคาร์ดหน่วยความจำ (MEMORY CARD)

ส่วนข้อมูลที่อยู่ใน EEPROM จะถ่ายลงใน RAM อีกทีหนึ่งและการเปลี่ยนข้อมูลในพารามิเตอร์ต่าง ๆ จะเปลี่ยนได้โดยใช้ HAND-HELD, TERMINAL หรือที่ FRONT PANEL โดยการเปลี่ยนข้อมูลจะเป็นเฉพาะข้อมูลใน RAM เท่านั้นโดย OUTPUT ของบล็อคจะถูกกำหนด VARIABLE NAME เช่น PV1, CV1 หรือค่าคงที่

ชนิดของ INTELLIGENT SPACE

INTELLIGENT SPACE มีอยู่ 4 ประเภท และแต่ละประเภทยังมีเอทาท์พุท (OUT PUT) ซึ่งขึ้นอยู่กับการใช้งานและออกแบบโดยมีชื่อเรียกแตกต่างกันออกไป

1. LINEARIZATION SPACE อนาลอกอินพุทคือ PVC1 ถึง PVC6

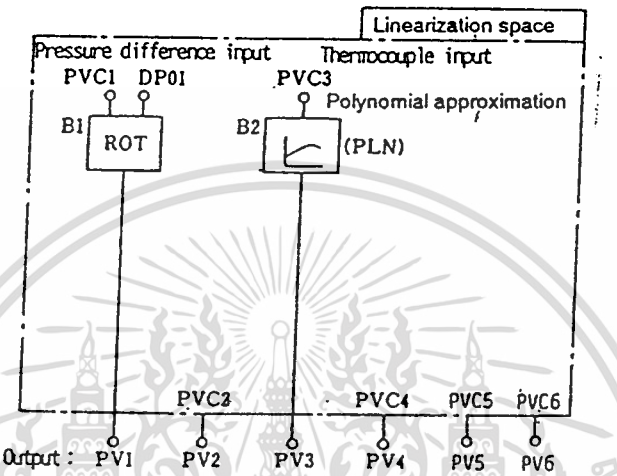
2. STATUS SPACE (1,2) ใช้สำหรับการควบคุม เพื่อเช็คสภาวะการททำงานต่าง ๆ

3. CONTROL SPACE (1,2) ใช้สำหรับการควบคุม เพื่อให้เป็นไปตามเงื่อนไขที่ต้องการ เช่น FEED PORWARD และ VALVE - COMPENSATION

4. SIMULATION SPACE ใช้สำหรับการเช็คโปรแกรม ว่าถูกต้องตามต้องการหรือไม่ถ้าไม่ถูกต้องจะได้ทำการแก้ไขโปรแกรม

ก่อนที่จะทำการเขียนโปรแกรมเราจะต้องศึกษาถึงโครงสร้างของแต่ละสเปซเสียก่อน

Linearization space

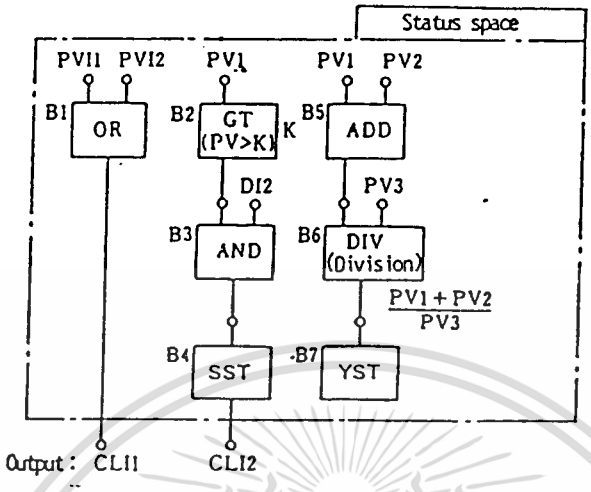


จากรูปนี้ ซึ่งมีสัญญาณอนาลอกอินพุท 2 อินพุท คือ PVC1 และ PVC3 โดยที่สัญญาณอินพุทที่ 1 จะผ่านการถอดรากที่สอง และอินพุทที่ 3 จะผ่านบล้อคการวางโพลีโนเมียล จนกระทั่งถึงที่ เอาร์พุทจะถูกเรียกเป็น PV1 และ PV3

ส่วนสัญญาณอินพุท 2,4,5 และ 6 จะเป็นสัญญาณ LINEAR ซึ่งอินพุทที่เข้ามาจะถูกเรียกเป็น PV2, PV4, PV5, และ PV6 เมื่อมาถึงที่เอาร์พุทจะถูกเรียกเป็น PV2, PV4, PV5 และ PV6 เป็นต้น

การทำงานของ Space นี้ก็คือ ก่อนที่สัญญาณอินพุทที่จะเข้ามาที่ SPACE นี้จะถูกแปลงเป็นหน่วยเปอร์เซนต์เสียก่อนโดยอัตโนมัติ คือค่า PVC1 ถึง PVC6 จะมีค่าเป็น 0 ถึง 100% หลังจากผ่านวงจร A/D CONVERTER แล้ว

โครงสร้าง STATUS SPACE (1,2)



จากรูปสามารถอธิบายได้

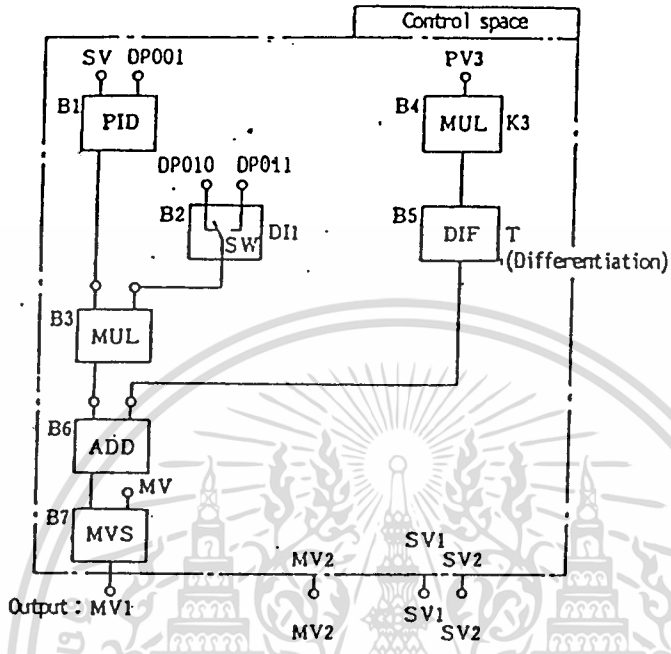
- จากบล็อคแรก (B1) เมื่อค่า PV1 หรือ PV2 มีสัญญาณผิดปกติตัวควบคุมนี้ก็จะหยุดการทำงาน (HALT)

-STATUS VOLUME (S1) แสดงค่าเป็น 0 หรือ 1, S1 จะเป็นผลลัพธ์ทางลอจิก (AND) ของสัญญาณดิจิตอลอินพุท DI2 กับฟังก์ชั่นโมดุล (GT) (SI มีให้เลือกใช้ถึง 8 และยังสามารถนำไปใช้กับสเปซ อื่น ๆ ก็ได้)

-DATA (Y1) เป็นผลลัพธ์ของ PV1+PV2/PV3และเก็บไว้ใช้ในสเปซอื่นได้ (มี Y 1 ถึง Y8 และยังนำไปใช้ใน SPACE อื่นได้

-ในหลักการแล้ว S1-S8, Y1-Y8 จะใช้ในสเปซนี้เพื่อให้เข้าใจได้ง่ายอย่างไรก็ตาม S หรือ Y ยังใช้ในสเปซอื่นเพื่อใช้จำผลลัพธ์

โครงสร้าง CONTROL SPACE

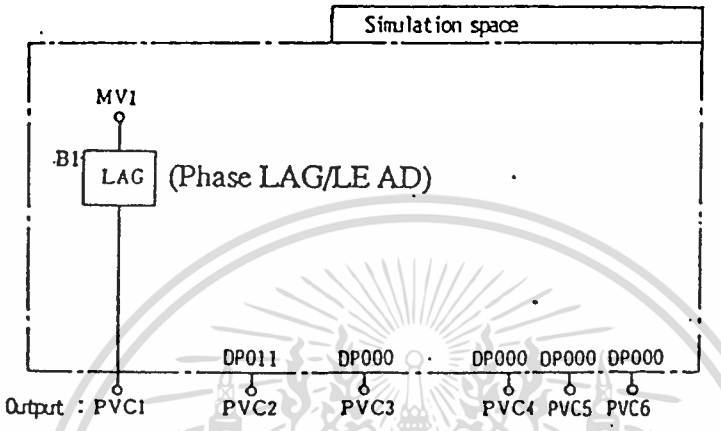


-ค่า GAIN ของ PID ALGORITHM จะเปลี่ยนตามค่าโดยเมื่อสวิทซ์ DI1 ทำงาน (คือค่า DP 010 จะเข้ามาคูณที่ B3 เมื่อ DI 1=0 และค่า DP011 จะเข้ามาคูณที่ B3 เมื่อ DI1=1 ส่วนสัญญาณ FEED FORWARD CONTROL จะใช้ค่า AUXILARY INPUT (PV3)

-MVS โมดูล จะคำนวณโดยสมการ MVn=MVn-1+ MV

-เอาท์พุทของคอนโทรลสเปซ 2 คือ MV3,MV4,SV3 และ SV4

โครงสร้าง SIMULATION SPACE



-สเปซนี้จะทำงานเมื่อได้ SIMULATION REQUSET เท่านั้น โดยผ่านจากตัว HAND HELD TERMINAL เมื่อต้องการ TEST การทำงานของตัวควบคุมโดยระหว่างการ SIMULATE นี้ค่า MV เอาท์พุทจะคงที่ไว้

-บล็อค B1 เป็นตัวหน่วงเวลาแบบ FIRST ORDER LAG เพื่อให้สัญญาณไปที่ PVC 1

-ค่าเอาท์พุท PVC 2 จะไม่ผ่านบล็อคอะไรเลย โดยจะผ่านสัญญาณอินพุทจากภายนอกเข้ามา ส่วน PVC3 และ PVC4 จะเซ็ทเป็น DP พารามิเตอร์ เช่น DP011 เป็นต้น

-ค่า PVC3,PVC4,PVC5,PVC6 ในที่นี้ไม่ได้ต่อบล็อคอะไรไว้เลยโดยจะเซ็ทค่าเป็นค่าเริ่มต้น (INITIAL=0)

-ในระหว่างการทำ SIMULATION, ฟังก์ชั่น ALARM, DISPLAY และ BUZZER ยังทำงานด้วย

-ค่า MV ใน SPACE นี้ต่างกับเอาท์พุทที่ส่งไปยัง FINAL CONTROLELEMENT ซึ่งเป็นค่าคงที่อยู่ ค่า MV ใน SIMULATION SPACE นี้สามารถเซ็ทผ่านคีร์บอร์ด ได้ในระหว่างอยู่ในการ SIMULATION EXECUTION

เอาท์พุทของ INTELLIGENT SPACE

แสดงค่าได้ตามตาราง

No.	Space name	Contents	Space output	Space output initial value	
1	Linearization space (LNZ space)	PV definition	PV1~PV6	PVC1~PVC6	Input is all linear.
2	Status space 1, 2 (STA space)	Control interactive flag definition	CLI1~CLI4	PVI1~PVI4	Inactive flag of relevant control variable
3	Control space 1, 2 (CTL space)	Control algorithm definition	SV1~SV4 MV1~MV4	SV1~SV4 MV1~MV4	* Output is same as input.
4	Simulation space (SIM space)	Simulation PVC definition	PVC1~PVC6	DP000	Simulation PVC not available

SYSTEM TABLE

ในระบบ (SYSTEM) จะประกอบไปด้วยตารางดังต่อไปนี้

-SYSTEM INPORMATION

-ANALOO INPUT TABLE

-FUNCTION MODULE PARAMETER

-POLYNOMIAL LINE

-LOOP CONTROL

-SPACE OUTPUT

ตาราง SYSTEM INFORMATION

สามารถดูได้จากตาราง

No.	Item	Description
1	SYSTEM NUMBER	Code to identify system
2	USER COMMENT	Name to identify system
3	ALARM CONFIRM	Alarm auto-recovery/manual confirmation
4	CYCLE TIME	Controller process period (CTS) (not control period)
5	PASS WORD	Password (integer)
6	SECURITY CODE	Security code (integer)
7	LOCK FLAG	Whether to prohibit changing parameters or not
8	RS232C	RS232C setting
9	RS485	RS485 setting

ตารางอนาลอกอินพุท (ANALOG INPUT)

No.	Item	Description
01	TAG NAME	Tag name
02	INPUT	Input installation specification
03	H-SC	Upper limit sensor check on/off
04	L-SC	Lower limit sensor check on/off
05	PV POINT	PV decimal point specification
06	RH[EU]	Meter range upper limit (engineering unit)
07	RL[EU]	Meter range lower limit (engineering unit)
08	PH[EU]	PV upper limit (engineering unit)
09	PL[EU]	PV lower limit (engineering unit)
10	PVBND[%]	PV non-sensitivity (%)
11	ΔPVLMT[%]	ΔPV limit (%)
12	ΔPVBND[%]	ΔPV non-sensitivity (%)
13	FT	Digital filter coefficient
14	LC[%]	Low cut (%)

ตาราง FUNCTION MODULE พารามิเตอร์

-DP 000 ถึง DP 009 จะเป็นค่าคงที่สำหรับเซ็ทค่าในระบบ (SYSTEM) ซึ่งไม่สามรถเซ็ทค่าได้

-DP 010 ถึง DP 149 สามารถเซ็ทค่าได้

ตาราง POLYNOMIAL LINE

Item	Description
Table No. 1 ~ 6	Polynomial line No.
X01 ~ X11	Polynomial line data X coordinate (X01, X11)
Y01 ~ Y11	Polynomial line data Y coordinate (Y01, Y11)

ตาราง ลูฟคอนโทรล (LOOP CONTROL)

No.	Item	Description
01	CTL, PT	Control variable name PV1–PV6
02	C.MODE	C mode enable/disable
03	SV(A)	SV settig enable/disable in /A mode
04	ΔT	Control period (CLT=CTSXΔT)
05	SVT	SV tracking enable/disable
06	Kp	Proportional gain
07	Ti	Integral time (m)
08	Td	Differential time (m)
09	Gp	Fap (%)
10	Gg	Gain in gap (0–1)
11	RT	Proportion
12	BS	Bias
13	DV	Deviation limit (%)
14	DV BND	Deviation non-sensitivity (band)
15	MH	MV upper limit (%)
16	ML	MV lower limit (%)
17	ΔMV	ΔMV limit (%)
18	ALPHA	2-degree free PID parameter α value
19	BETA	2-degree free PID parameter β value
20	GAMMA	2-degree free PID parameter γ value
21	D/R	Direct/reverse specification for control action
22	V.D	Valve operation specification (direct/reverse)
23	PSCALE	PV bar graph scale pattern 1–20: divided/No. 21–28: user scale
24	SSCALE	SV bar graph scale pattern 1–20: divided/No. 21–28: user scale
25	MSCALE	MV bar graph scale pattern 1–20: divided/No. 21–28: user scale

จากตารางเป็นการแสดงค่าพารามิเตอร์ของแต่ละ ลูฟคอนโทรล ซึ่งเครื่องรุ่นนี้มีถึง 4 ลูฟคอนโทรล ส่วนการใช้งานจะต้องระบุเอาท์พุทและโมดูลให้ตรงกัน

การติดต่อสื่อสารกับ HOST SYSTEM

1. COMPUTER/LOCAL INDICATION

ค่าของ SV และ MV จะถูกเซ็ทค่าผ่านมา (TRANSMITTED) จาก HOST SYSTEM มายังตัวควบคุมทำได้โดยการเซ็ท COMPUTER FLAG จากตัว HOST SYSTEM โดยขณะทำงาน LED จะติดสว่างขึ้น

2. CMP FLAG

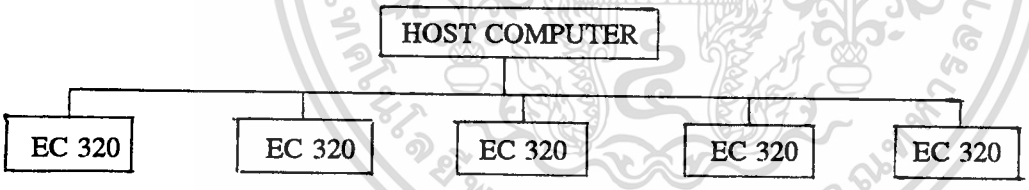
จะมีเฉพาะของแต่ละลูฟโดยการทำการเซ็ทได้เฉพาะที่ HOST SYSTEM โดยอยู่ที่โมดูล PMS โมดูลทำได้โดยการ SET กับ RESET

SET : ต้องการติดต่อสื่อสาร

RESET : ไม่ต้องการติดต่อสื่อสาร

DATA TRANS MITION

การติดต่อจะใช้ บัส เป็นแบบ MODBUS



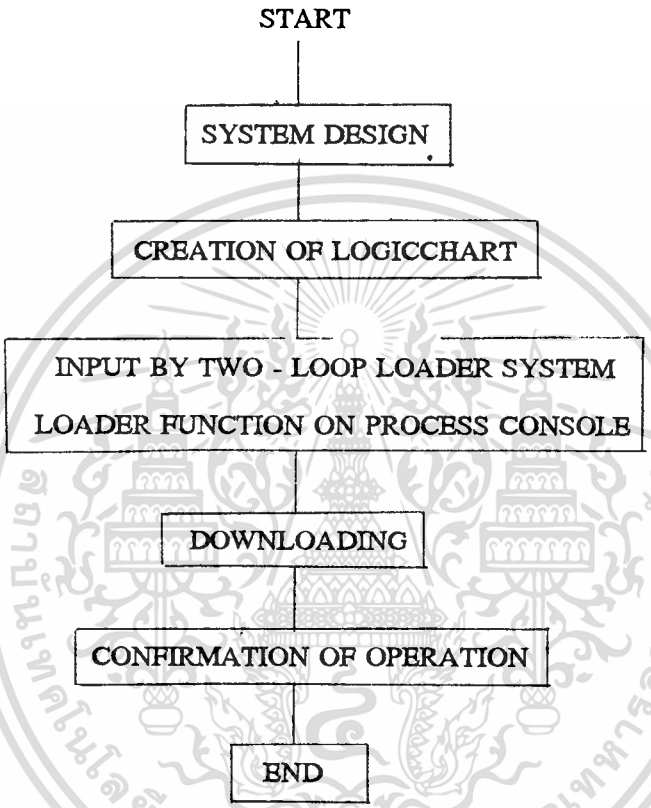
DOWNLOADING

เป็นการ LOAD EC - 394 หรือทำได้จากการถ่ายข้อมูลจาก MEMORY CARD ส่วนตัว SELF - CIAGONOSIS ฟังก์ชั่นของตัวควบคุม ถ้าเช็คเจอค่า ERROR ใน RAM เครื่องจะถ่ายข้อมูล (COAD) จาก EEPROM ไปสู่ RAM ทันที

SYSTEM BUILDUP

การ DOWN LOAD โปรแกรมทำได้โดยการ LOAD จาก LOADER EC - 394 (โดยบัส RS-232C) หรือ MEMORY CARD

ขั้นตอนการ LOADING แสดงได้ดัง FLOW CHART



-วิเคราะห์รูปแบบ และเลือกใช้ฟังก์ชั่นต่าง ๆ ของตัวควบคุม

-ระบุฟังก์ชั่นรวมทั้งลอจิกต่าง ๆลงใน SPACE ต่าง ๆ

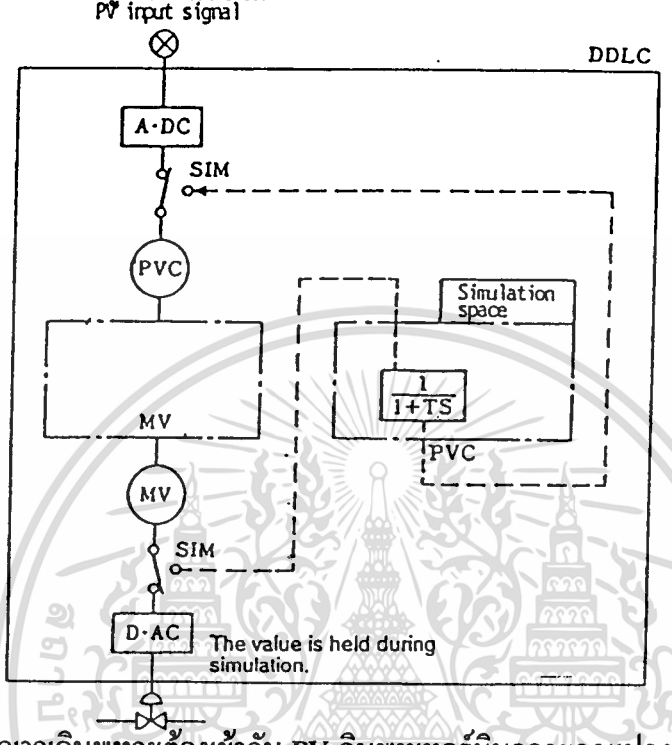
-เมื่อสร้างโปรแกรมเสร็จก็ทำการถ่ายข้อมูล (LOADING) โดยตัว LOADER

-ช่วง LOADING จะเช็ค SYSTEM CODE ที่ด้านหน้าเครื่องรวมทั้งพารามิเตอร์ต่าง ๆ ว่า LOAD ลงมาถูกต้องหรือไม่

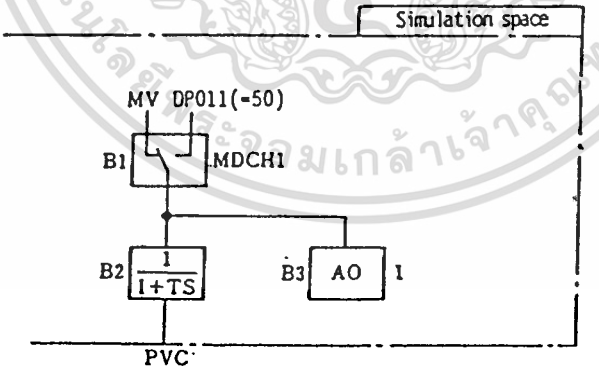
-เช็ค SYSTEM CODE ,ฟังก์ชั่น, อินพุท/เอาท์พุท และอื่น ๆ ก่อนใช้งาน

SIMULATION SPACE

หลังจากการ DOWNLOAD FILE ใส่เข้ากับตัวควบคุมแล้ว SIMULATION SPACE สามารถช่วยได้ในการเช็คโปรแกรมได้ว่าถูกต้องหรือไม่



จากรูป สัญญาณอินพุทจะต้องเข้ากับ PV อินพุทเทอร์มินอลจะถูกแปลงด้วยวงจร A/D และถูกเรียกเป็น PVC



-เมื่อวงจร SIMULATION ทำงานเครื่องจะเปลี่ยนโหมดจากโหมดอัตโนมัติมาเป็นระบบ MANUAL ทันที

-ADP พารามิเตอร์จะเซ็ทค่าเริ่มต้นของ MV ทันที

-จากบล็อค B3 จะสร้างสัญญาณอนาลอกของตำแหน่งวาล์วซึ่งจะส่งไปเป็นสัญญาณ AI

-หลังจากเลิกการ SIMULATION เครื่องควบคุมจะกลับมาตำแหน่ง MANUAL

-ในช่วงการ SIMULATION ที่ตัว "HOLD" จะกระพริบทุก ๆ 0.5 ไซเคิลต่อวินาที

ในการเขียนโปรแกรม เราจำเป็นต้องศึกษาความหมายและการใช้งานของบล็อคต่าง ๆ เสียก่อน เพื่อที่จะเขียนโปรแกรมได้เร็วและถูกต้องตามความต้องการ

Arithmetic Operation

○:Data-type data
●:Flag-type data
◎:Hybrid-type data
▼:Integer-type data

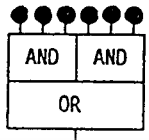
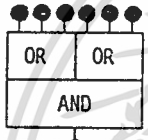
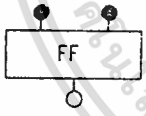
Y(-1):Last caluculated value for Y
. :Additional data storage areas are required, see Table 9.1

No.	Function	Function Block	Data type	Description	Initial value
00	No operation	U1 NOP Y=U1	U1:data	No calculation is executed	
01	Addition/ subtraction	U1 U2 ADD H1 H2 H3 Y Y=H1U1+H2U2+H3	U1:data U2:data H1:data H2:data H3:data	H1,H2,H3:constant	H1=DP001 H2=DP001 H3=DP000
02	Addition of multiple points	U1U2U3U4U5U6 ADM Y Y=U1+U2+U3+U4+U5+U6	U1:data U2:data U3:data U4:data U5:data U6:data	Two to six inputs are acceptable. Each unassigned input is ignored at the calculation.	
03	Multiplica-tion	U1 U2 MUL H1 Y=H1U1U2	U1:data U2:data H1:data	H1:constant	H1=DP001
04	Division	U1 U2 DIV H1 H2 H3 H4 Y Y=(H1U1+H2)/(H3U2+H4)	U1:data U2:data H1:data H2:data H3:data H4:data	If H3U2+H4=0 then Y=last value calculated for Y before the denominator is 0 H1,H2,H3,H4:constant	H1=DP001 H2=DP000 H3=DP001 H4=DP000
05	Absolute value	U1 ABS Y=|U1|	U1:data	If U1 >= 0 then Y=U1 If U1 < 0 then Y=-U1	

Logical Operation (1/2)

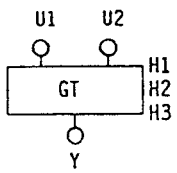
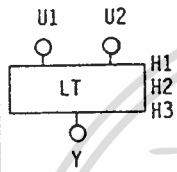
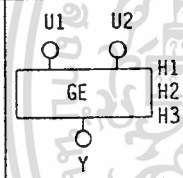
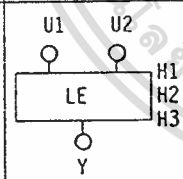
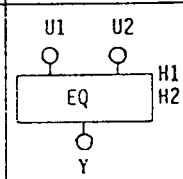
No.	Function	Function Block	Data type	Description	Initial value
10	Two-input selection (SW)	U1 U2 SW H1 Y If H1=1, Y=U2 If H1=0, Y=U1	U1:hybrid U2:hybrid H1:flag	According upon the status (0 or 1) of the condition switch H1, data U1 or U2 is selected as block output. The flag type parameter can be specified by H1.	H1=DP000
11	Logical product	U1U2U3U4U5U6 AND Y Y=U1∧U2∧U3∧U4∧U5∧U6	U1:flag U2:flag U3:flag U4:flag U5:flag U6:flag	Two to six inputs are acceptable. (If all inputs=1, Y=1) Each unassigned input is ignored at the calculation.	
12	Logical sum	U1U2U3U4U5U6 OR Y Y=U1∨U2∨U3∨U4∨U5∨U6	U1:flag U2:flag U3:flag U4:flag U5:flag U6:flag	Two to six inputs are acceptable. (If any inputs=1, Y=1) Each unsigned input is ignored at the calculation.	
13	Exclusive OR	U1 U2 EOR Y=U1 ⊕ U2	U1:flag U2:flag	The exclusive logical sum of U1 and U2 is output. U1 \ U2: 0 → 0, 1 → 1 0 → 0, 1 1 → 1, 0	
14	Signal inversion (negation)	U1 NOT $Y=\overline{U1}$	U1:flag	The flag status is changed from U1 to Y as follows: If U1=1, then Y=0 If U1=0, then Y=1	

Logical Operation (2/2)

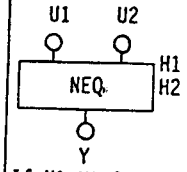
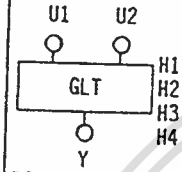
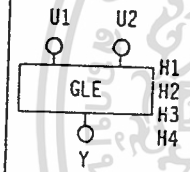
No.	Function	Function Block	Data type	Description	Initial value
15	AND/OR (A/O)	U1U2U3U4U5U6  Y Y=(U1∧U2∧U3) ∨(U4∧U5∧U6)	U1:flag U2:flag U3:flag U4:flag U5:flag U6:flag	If any input is unassigned then its input is set equal to 1. If all input is unassigned then Y=0.	
16	OR/AND (O/A)	U1U2U3U4U5U6  Y Y=(U1∨U2∨U3) ∧(U4∨U5∨U6)	U1:flag U2:flag U3:flag U4:flag U5:flag U6:flag	If all input is unassigned then Y=1. If any input is unassigned then its input is set equal to 0.	
17	Flip-Flop (FF)	U1 U2  Y U1=1 ;Y=1 U1=0 U2=0;Y=Y(-1) U1=0 U2=1;Y=0	U1:flag U2:flag	U1:Set flag U2:Reset flag	

Comparison (1/2)

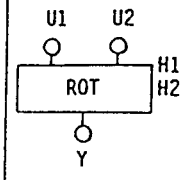
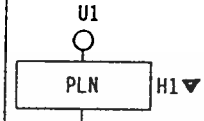
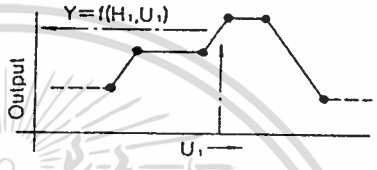
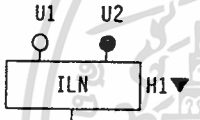
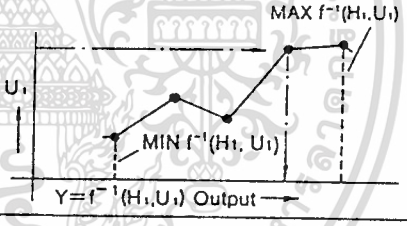
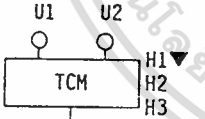
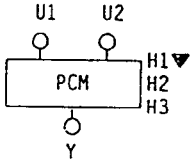
All comparison function blocks make data comparison only.
The result of the comparison is output as flag type data (true=1, false=0)

No.	Function	Function Block	Data type	Description	Initial value
20	GT (Greater than)	 If U1<H1U2+H2 then Y=1 else Y=0	U1:data U2:data H1:data H2:data H3:data	The output returns from '1' to '0' when the condition of U1<=H1U2+H2-H3 is met. H3 determines the hysteresis band.	H1=DP001 H2=DP000 H3=DP000
21	LT (Less than)	 If U1<H1U2+H2 then Y=1 else Y=0	U1:data U2:data H1:data H2:data H3:data	The output returns from '1' to '0' when the condition of U1>= H1U2+H2+H3 is met. H3 determines the hysteresis band.	H1=DP001 H2=DP000 H3=DP000
22	GE (Greater than or equal to)	 If U1>=H1U2+H2 then Y=1 else Y=0	U1:data U2:data H1:data H2:data H3:data	The output returns from '1' to '0' when the condition of U1<H1U2+H2-H3 is met. H3 determines the hysteresis band.	H1=DP001 H2=DP000 H3=DP000
23	LE (Less than or equal to)	 If U1<=H1U2+H2 then Y=1 else Y=0	U1:data U2:data H1:data H2:data H3:data	The output returns from '1' to '0' when the condition of U1>H1U2+H2+H3 is met. H3 determines the hysteresis band.	H1=DP001 H2=DP000 H3=DP000
24	EQ (Equal to)	 If U1=H1U2+H2 then Y=1 else Y=0	U1:data U2:data H1:data H2:data		H1=DP001 H2=DP000

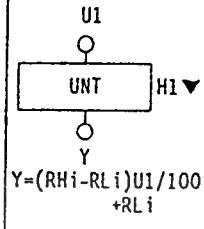
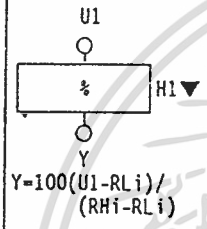
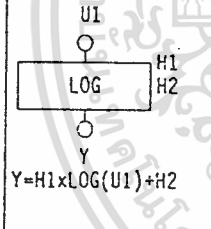
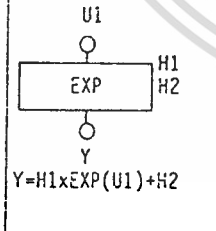
Comparison (2/2)

No.	Function	Function Block	Data type	Description	Initial value
25	NEQ (Not equal)	 If U1≠H1U2+H2 then Y=1 else Y=0	U1:data U2:data H1:data H2:data		H1=DP001 H2=DP000
26	<<	 If H1U2+H2<U1 <H3U2+H4 then Y=1 else Y=0	U1:data U2:data H1:data H2:data H3:data H4:data		H1=DP001 H2=DP000 H3=DP001 H4=DP000
27	<=<=	 If H1U2+H2<=U1 <=H3U2+H4 then Y=1 else Y=0	U1:data U2:data H1:data H2:data H3:data H4:data		H1=DP001 H2=DP000 H3=DP001 H4=DP000

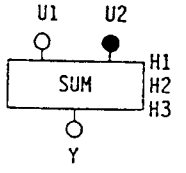
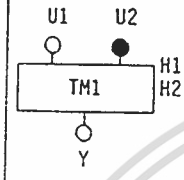
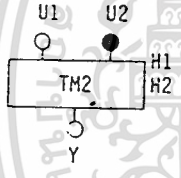
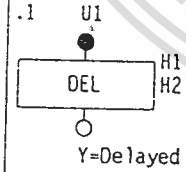
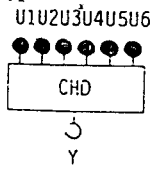
Linearize (1/2)

No.	Function	Function Block	Data type	Description	Initial value
30	Square root extraction	 Y=H1 U1xU2	U1:data U2:data H1:data H2:data	Y=H1 U1xU2 (When U1xU2<=0, Y=0) H2:Setting of low cut When U1<H2, the calculation is made on the condition of U1=0. The low cut is not performed when H2=0.	H1=DP008 (10) H2=DP000
31	Polynomial line conversation	 H1:Table number (1 - 6) U1:Variable	U1:data H1:int.	If U1 is outside the area defined by the polynomial line tables, the values output will be on the dashed lines indicated in the diagram below. 	H1=1
32	Inverse polynomial line conversion	 H1:Table number (1 - 6) U1:Variable	U1:data U2:flag H1:int.	Polynomial line inverse function If U2=0 then Y=MIN[f^{-1}(H1,U1)] else Y=MAX[f^{-1}(H1,U1)] 	H1=1
33	Temperature compensation	 Y=U1x(H2+H3)/ {(Eng.value of H1) +H3}	U1:data U2:data H1:int. H2:data H3:data	This is used for temperature compensation. H1:Input code H2:Design temperature(Td) H3:Constant The constant of H3 is 273.15(deg C) or 459.4(deg F). H1=0;use engineering unit U2 data H1=1 to 6;PV1 to PV6	H2=DP009 (100) H3=DP004 (273.15)
34	Pressure compensation	 Y=U1x{(Eng.value of H1)+H3}/(H2+H3)	U1:data U2:data H1:int. H2:data H3:data	This is used for pressure compensation. H1:Input code H2:Design pressure(Pd) H3:Constant The constant of H3 is 1.0332(kg/cm2), 101.3(kPa) or 14.72(PSIG). H1=0;Engineering unit data of U2 H1=1 to 6;PV1 to PV6	H2=DP008 (10.0) H3=DP006 (1.0332)

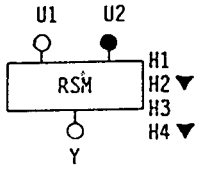
Linearize (2/2)

No.	Function	Function Block	Data type	Description	Initial value
35	Engineering unit conversion		U1:data H1:int.	This is used for the engineering unit conversion. RLi:Lower limit of instrument range for H1 RHi:Higher limit of instrument range for H1 H1 :Input code 1 to 6 : PV1 to PV6	H1=1
36	Percent conversion		U1:data H1:int.	This is used for the percent unit conversion. RLi:Lower limit of instrument range for H1 RHi:Higher limit of instrument range for H1 H1 :Input code 1 to 6 : PV1 to PV6	H1=1
37	Natural Logarithm		U1:data H1:data H2:data		H1=DP001 H2=DP000
38	Exponential		U1:data H1:data H2:data		H1=DP001 H2=DP000

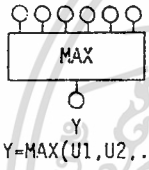
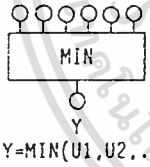
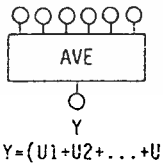
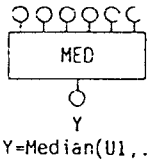
Counter Timer

No.	Function	Function Block	Data type	Description	Initial value
40	Totalizer (Summer)	 U1:Data U2:Initialization flag	U1:data U2:flag H1:data H2:data H3:data	This is used for flow totalization, program pattern timing generation, etc. If U2=1, then: Y=H1+(H2xU1+H3) [Initialized state] If U2=0, then: Y=Y(-1)+(H2xU1+H3) [Totalization] Y(-1):Last calculated value for Y	H1=DP000 H2=DP001 H3=DP000
41	Timer 1 (Retrigger type)	 U1:Set time (sec) U2:Reset flag H1,H2;Set time coefficient	U1:data U2:flag H1:data H2:data	If U2=1, then Y=H1U1+H2 Initialization value If U2=0, then Y=Y(-1)-CTS If Y<0, then Y=0	H1=DP001 H2=DP000
42	Timer 2 (Non-retrigger type)	 U1:Set time (sec) U2:Reset flag H1,H2:Set time coefficient	U1:data U2:flag H1:data H2:data	If Y(-1)>0, then Y=Y(-1)-CTS but if Y<0, Y=0 for U2=0, else if U2=1, then Y=H1U1+H2 Initialization value	H1=DP001 H2=DP000
43	On/off delay timer	 Y=Delayed U1	U1:falg H1:data H2:data	H1:Delay on (sec) H2:Delay off (sec)	H1=DP000 H2=DP000
46	Status change detection		U1:data U2:data U3:data U4:data U5:data U6:data	At the rising point of input from U1 to U6, the output is turned ON for one processing cycle(Y). Usually only rising point is detected. When falling point should be detected, invert input.	

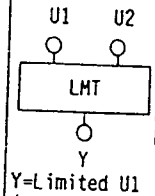
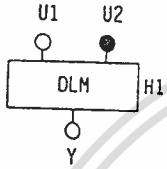
Selector Limiter (1/2)

No.	Function	Function Block	Data type	Description	Initial value
49	Double precision totalizer (Summer)		U1:data U2:flag H1:data H2:int. H3:data H4:int.	If U2=1 then:Y=H1+U1(CTS/K) [Initialized state] If U2=0 and Y<H3 then:Y=Y(-1)+U1(CTS/K) [Totalization] Y(-1):Last calculated value for Y If U2=0 and Y>=H3 then:Y=Y(-1)+U1(CTS/K)-H3 H2:Time unit (0/1/2:sec/min/hour) H3:Auto reset point H4:Output precision (0/1/2/3/4:1/0.1/0.01/0.001/0.0001) K=1/60/3600, when H2=0/1/2	H1=DP000 H2=0 H3=DP000 H4=0

Selection is made excluding inactive signals such as sensor errors.
If all inputs are inactive, Y=Y(-1) (last calculated value for Y)

No.	Function	Function Block	Data type	Description	Initial value
50	Maximum value	 Y=MAX(U1,U2,...,U6)	U1:data U2:data U3:data U4:data U5:data U6:data	Two to six inputs are acceptable. Maximum value of U1 - U6 is output as Y.	
51	Minimum value	 Y=MIN(U1,U2,...,U6)	U1:data U2:data U3:data U4:data U5:data U6:data	Two to six inputs are acceptable. Minimum value of U1 - U6 is output as Y.	
52	Mean value	 Y=(U1+U2+...+Ui)/i	U1:data U2:data U3:data U4:data U5:data U6:data	(U1+U2+U3+U4+U5+U6)/i where i=2 to 6 inputs	
53	Median value	 Y=Median(U1,...,U6)	U1:data U2:data U3:data U4:data U5:data U6:data	Median value of U1,U2,U3,U4,U5,U6. When the number of inputs is even, the smaller of the two medians is output as Y.	

Selector Limiter (2/2)

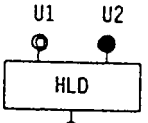
No.	Function	Function Block	Data type	Description	Initial value
54	Limiter	 Y=Limited U1 (U1 limited at high and low points)	U1:data U2:data H1:data H2:data H3:data H4:data	Condition expression 1)If H1U2+H2<=U1<=H3U2+H4, then Y=U1 2)If H1U2+H2>U1 then Y=H1U2+H2 3)If H3U2+H4<U1 then Y=H3U2+H2	H1=DP001 H2=DP000 H3=DP001 H4=DP000
55	Changing rate limiter	 Y=Limitted U1 (△U1 is limited)	U1:data U2:flag H1:data	H1:Changing rate limit value (%/CTS) U2:Inverted initialization flag When U2=1, 1)If U1-Y(-1)<=H1, then Y=U1 2)If U1-Y(-1)>H1, then Y=Y(-1)+H1 3)If Y(-1)-U1>H1, then Y=Y(-1)-H1 When U2=0, Y=U1 If INZ=0 (at initialization time), Y=U1. Y(-1):Last calculated value for Y.	H1=DP001

NOTE:Function blocks 50 - 53 accept from 2 - 6 inputs.

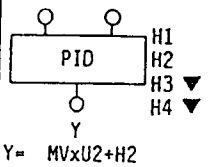
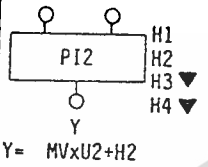
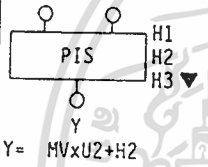
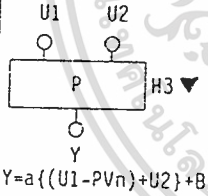
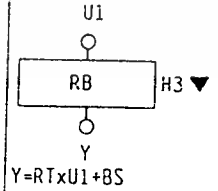
Dynamic Characteristics Compensation (1/2)

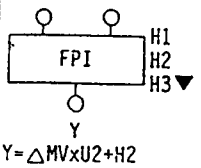
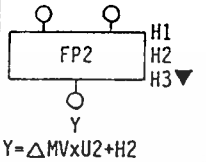
No.	Function	Function Block	Data type	Description	Initial value
60	Lead/Lag (LAG)	.1 U1 U2 LAG H1 H2 Y Y={1+(H1S)}/{1+(H2S)}	U1:data U2:flag H1:data H2:data	H1:Gain parameter, change gain = (H1-H2)/H1 H2:Lead/lag time constant (sec) U2:Initialize flag =0 Not requesting initialization =1 Requesting initialization S :Internal LaPlace Transform If INZ=0 or U2=1, then Y=U1 else Y=calculated value.	H1=DP001 H2=DP001
61	Differential (DIF)	.1 U1 U2 DIF H1 H2 Y Y=(H1S)/{1+(H2S)}	U1:data U2:flag H1:data H2:data	H1:Gain parameter, gain = H1/H2 H2:Differential time (sec) U2:Initialization flag =0 Not requesting initialization =1 Requesting initialization If INZ=0 or U2=1, then Y=0 else Y=calculated value S :Internal LaPlace Transform	H1=DP001 H2=DP001
62	Dead time	.4 U1 U2 DT H1 H2 H3 H4 Y Y=Previous U1	U1:data U2:flag H1:flag H2:int. H3:data H4:data	If H1=1 or INZ=0 then initialization, Y=U1 and all previous U1=present U1 else Y=Delayed and interpolated signal Delay time=H2xH3xCTS U1:Input data U2:Sampling cycle set flag H1:Initialization flag H2:No. of samples (Max. 16 samples) H3:Sampling period (in CTS's) H4:No. of output samples	H1=DP000 H2=1 H3=DP001 H4=DP001
63	Moving average	.2 U1 U2 MAV H1 H2 H3 Y Y=moving average	U1:data U2:flag H1:flag H2:int. H3:data	If H1=1 or INZ=0 then Y=U1 and all previous U1=present U1 else Y=moving average U1:Input data U2:Sampling cycle set flag H1:Initialization flag H2:No. of samples (Max. 8 samples) H3:Sampling period (In CTS's)	H1=DP000 H2=1 H3=DP001
64	Sampling average	.2 U1 U2 SAV H1 H2 Y Y=averaged value of n samples	U1:data U2:flag H1:flag H2:int.	If H1=1 or INZ=0 then Y=U1 and all previous U1=present U1 else Y=averaged value of n samples U1:Input data U2:Sampling timing signal H1:Initialization flag H2:No. of samples (Max. 8 samples)	H1=DP000 H2=1

Dynamic Characteristics Compensation (2/2)

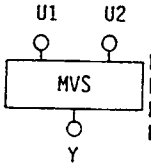
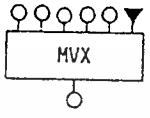
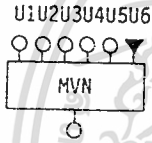
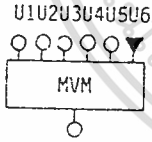
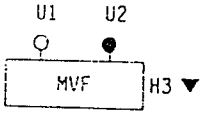
No.	Function	Function Block	Data type	Description	Initial value
65	Hold	U1 U2  Y If U2=1, Y=U1 If U2=0, Y=Y(-1)	U1:hybrid U2:flag	If U2=0 then Y=Y(-1),the last value at U1 If U2=1 then Y=present value of U1	

Control Operation (2/1)

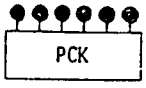
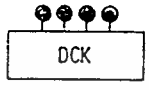
No.	Function	Function Block	Data type	Description	Initial value
70	PID	.1 U1 U2  Y= MVxU2+H2	U1:data U2:data H1:data H2:data H3:int. H4:int.	U1:SV U2:Multiplier to output H1:Derivative constant H2:Bias to output H3:Loop number (1 - 4) H4:Differential type 0=PI-D 1=PID 2=IPD	U2=DP001 H1=Dp001 H2=DP000 H3=1 H4=1
71	Error squared PID	.1 U1 U2  Y= MVxU2+H2	U1:data U2:data H1:data H2:data H3:int. H4:int.	Same as block 70	U2=DP001 H1=DP001 H2=DP000 H3=1 H4=1
72	Sample PI	.1 U1 U2  Y= MVxU2+H2	U1:data U2:data H1:data H2:data H3:int.	Same as block 70 except there is no differential type option (H4)	U2=DP001 H1=DP001 H2=DP000 H3=1
73	Proportional control	U1 U2  Y=a{(U1-PVn)+U2}+B	U1:data U2:data H3:int.	U1:SV H2:Compensation input H3:Loop number (1 - 4) A=Kp B=Ti Y=MV If the P function block is used, the MVS function block is not used for the controller output.	U2=DP000 H3=1
74	Ratio control	U1  Y=RTxU1+BS	U1:data H3:int.	U1:Input H3:Loop number (1 - 4) RT=Ratio BS=Bias	H3=1

No.	Function	Function Block	Data type	Description	Initial value
76	Two-degree-of-freedom PID	.1 U1 U2  Y=△MVxU2+H2	U1:data U2:data H1:data H2:data H3:int.	U1:SV U2:Multiplier to output H1:Derivative constant H2:Bias to output H3:Loop number Y= MV (compensation value)	U2=DP001 H1=DP001 H2=DP000 H3=1
77	Error squared two-degree-of-freedom PID	.1 U1 U2  Y=△MVxU2+H2	U1:data U2:data H1:data H2:data H3:int.	Same with FPI	U2=DP001 H1=DP001 H2=DP000 H3=1
78	Loop number designation	U1  Z=Bar graph status	U1:flag H1:flag H3:flag H4:flag	This is used to make switching online of the SV and PV values on the bar graph and digital display. The switching involves different loops H1:Display mode specification H3:Display loop number H4:Loop position [Diaplay Loop Specification] See page 136	H1=DP000 H3=DP000 H4=DP000
79	Mode inhibit	U1 U2 U3  Z=Control-mode inhibition	U1:flag U2:flag U3:flag H3:int.	U1=1:M mode is inhibited for H3 loop U2=1:A mode is inhibited for H3 loop U3=1:C mode is inhibited for H3 loop H3 :Loop number (1 - 4)	H3=1

MV Processing

No.	Function	Function Block	Data type	Description	Initial value
80	MV caluculation	 Y=MV(n)+U2 MV(n)=MV(n-1)+U1	U1:data U2:data H1:data H2:flag H3:int. H4:flag	U1: MV U2:MV initialization value H1:Feed forward input H2:MV initialize flag H3:Loop number (1 - 4) H4:Specification to ignore MV at initialization If H2=1, initialization of Y if H2=0, at control time.	H1=DP000 H2=DP000 H3=1 H4=DP000
82	Max. MV selector	U1U2U3U4U5U6  Y=MAX(U1,U2,U3,U4) (at U5=0) Z=SELmn (m=U6)	U1:data U2:data U3:data U4:data U5:data U6:int.	If U5=0 then Y=MAX(U1,U2,U3,U4) else Y=input specified by U5 SELmn=1 (m=U6,n=selected Un) SELmk=0 (m=U6,k=not selected Uk) (U6:1 - 2)	U5=DP000 U6=1
83	Min. MV selector	U1U2U3U4U5U6  Y=MIN(U1,U2,U3,U4) (at U5=0) Z=SELmn (m=U6)	U1:data U2:data U3:data U4:data U5:data U6:int.	If U5=0 then Y=MIN(U1,U2,U3,U4) else Y=input specified by U5 SELmn=1 (m=U6,n=selected Un) SELmk=0 (m=U6,k=not selected Uk) (U6:1 - 2)	U5=DP000 U6=1
84	Median MV selector	U1U2U3U4U5U6  Y=MED(U1,U2,U3,U4) (at U5=0) Z=SELmn (m=U6)	U1:data U2:data U3:data U4:data U5:data U6:int.	If U5=0 then Y=MED(U1,U2,U3,U4) else Y=input specified by U5 SELmn=1 (m=U6,n=selected Un) SELmk=0 (m=U6,k=not selected Uk)	U5=DP000 U6=1
85	Fixed MV output	 Z={MVn=U1}	U1:data U2:flag H3:int.	U1:MV request value U2:MV request flag H3:Loop number (1 - 4)	H3=1

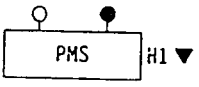
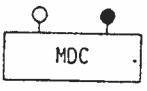
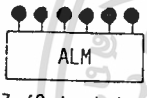
Alarm Check

No.	Function	Function Block	Data type	Description	Initial value
86	PV check	U1U2U3U4U5U6  Zi={PViCK=Ui}	U1:flag U2:flag U3:flag U4:flag U5:flag U6:flag	If U1=1, then:the upper limit, lower limit, and rate of change alarm checks on PVi are made. (PViCK is set to 1) else:the upper limit, lower limit, and rate of change alarm checks on PVi are not made. (PViCK is set to 0)	
87	Deviation check	U1U2U3U4  Zi={DVi=Ui}	U1:flag U2:flag U3:flag U4:flag	If Ui=0 then:the i loop deviation alarm check is not made.(DVi=0) else:the i loop deviation alarm check is made.(DVi=1) (deviation=rate of change)	

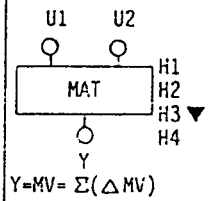
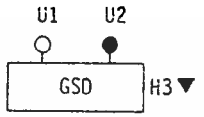
Output, Definition (1/2)

No.	Function	Function Block	Data type	Description	Initial value
90	Analog output	U1 AO H1▼ Y=U1 Z=[AOn=U1] H1:Output point 1 for AO1 2 for AO2	U1:data H1:int.	The value of U1 ranging from 0 to 100% is converted analog output in the range of 1-5 VDC. An output value between 0V and 1V can be generated by inputting a negative value.	H1=1
91	Digital output	U1 DO H1▼ Y=U1 Z=[DOn=U1] H1:Output point 1 for DO1 2 for DO2 3 for DO3 4 for DO4	U1:flag H1:int.	The status (0 or 1) of the U1 flag is digitally output to the point specified by H1.	H1=1
92	Status registration	U1 SST H1▼ Y=U1 Z=[Si=U1] H1:Flag number i=1 to 16	U1:flag H1:int.	The U1 data is set into the status flag Si (i=H1:1 to 16). When U1=0, Y=0 When U1=1, Y=1	H1=1
93	Data registration	U1 YST H1▼ Y=U1 Z=[Yi=U1] H1:Data number i=1 to 16	U1:data H1:int.	U1 data is registered as variable Yi (i=H1:1 to 16). Y=U1	H1=1

Output, Definition (2/2)

No.	Function	Function Block	Data type	Description	Initial value
94	Parameter set	U1 U2  Z={Specified parameter=U1}	U1:data U2:flag H1:var.	U1:Data to be set U2:Request flag H1:Parameter specification (see Table 9.4)	H1=DP010
95	Mode change	U1 U2  Z=Control mode U1:Mode assign U2:Request flag H3:Loop No.(1 - 4)	U1:data U2:flag H3:int.	Control mode is switched as apecified by the U1 input. U2=0 : Not requesting mode change =1 : Requesting mode change U1 : Mode specification =1 : M =2 : A =3 : C U1 values other than 1 to 3 are ignored.	H3=1
99	Alarm output	U1U2U3U4U5U6  Z={Output to the alarm LED =U1 U2 U3 U4 U5 U6}	U1:flag U2:flag U3:flag U4:flag U5:flag U6:flag	If U1 to U6 is on, alarm LED on the front panel lights. When the ALM function block is used, only those variables registered for each input are targetted for ALM LED.	

Self-tuning

No.	Function	Function Block	Data type	Description	Initial value
103	Self-tuning	 Y=MV= Σ(△MV)	U1:data U2:data H1:flag H2:flag H3:int. H4:data	U1:SV U2:PV H1:ON(1)/OFF(0) self tuning H2:MV initialization flag H3:Loop number (1 - 4) H4:MV initial value Y :Manipulated volume MV	H1=DP000 H2=DP000 H3=1
102	Gain scheduling		U1:data U2:flag H3:int.	U1:Scheduling input U2:ON(1)/OFF(0) scheduling H3:Loop number (1 to 4)	H3=1

Bar graph display

No.	Function	Function Block	Data type	Description	Initial value
110	Bar graph display	U1 BDP H1 H2 H3 ▼	U1:flag H1:flag H2:data H3:int.	U1:Execution flag H1:Bar graph display(0)/point display(1) H2:Loop number H3:BAR number (1 - 4) If U1=1 then BARn is registered.(n=H3)	H1=DP000 H2=DP001 H3=1
111	Bar graph display mode	U1 BDM H1 H2 H3 H4	U1:flag H1:data H2:flag H3:data H4:data	U1:Execution flag H1:Display MV and mode H2:One loop display(0)/two loop display(1) H3:Display BAR number H4:Display BAR number (See page 138)	H1=DP000 H2=DP000 H3=DP000 H4=DP000

คุณลักษณะของ ECBUS DRIVER

ECBUS DRIVER สำหรับ FIX DMACS บน WINDOWS ใน COMM และ AECI เป็นเครื่องมือสำหรับจัดเตรียมการติดต่อและรับส่งข้อมูลระหว่าง SAC (SCAN , ALARM AND CONTROL) บนโปรแกรม FIX DMACS บน WINDOWS กับอุปกรณ์ทางด้าน HARDWARE ของตัวควบคุม แบบขบวนการลูฟควบคุมในตระกูล EC 300 และ TL, EX100 PROGRAMABLE CONTROLLER

ECBUS DRIVER จะให้การสนับสนุนตามหน้าที่ดังนี้

-N:N TRANSMISSION CONTROL

-DATA TRAMSMISSION FUNCTION (RAW DATA, REFERENCE และ GROUP DATA)

-DETECTION OF A TROMSMISSION ERROR

-บอกถึงรายละเอียดกสนทำงานของโปรแกรมในกรณีการติต่อเกิดข้อผิดพลาด

-OBTAINING และ SURRENDERING THE MASTER

ชนิดของ ECBUS DRIVER

สำหรับการติดต่อของตัวควบคุมกับระบบของ FIX DMACS จะมีการติดต่อกันที่ PORTS (COM) มาตรฐานเพียงอย่างเดียวและแบบ 50- PIN ในการาติต่อแบบ AECI CARD โดยสามารถสรุปถึงการติดต่อระหว่าง FIX CMACS กับ ตัวควบคุมได้ 2 ข้อคือ

1. STANDARD CONNECTION PORTS (COMM)
2. 50- PIN CONNECTION AECI CARD (AECI)

โดย PORTS มาตรฐานจะใช้ COM 1 และ COM 2 สำหรับการติดต่อใน COMM และ 50-PIN CONNECTOR ก็ติดอยู่กับ AECI CARD จะใช้สำหรับติดต่อในแบบ AECI CARD โดยที่ AECI 1 BOARD สามารถที่จะติดอยู่ใน คอมพิวเตอร์ได้

ในระบบ COMM และระบบ AECI ไม่สามารถที่จะใช้บนคอมพิวเตอร์ตัวเดียวกันในเวลาเดียวกันได้

EC 375,ECBUS DRIVER FOR FIX DMACS FOR WINDOWS

อุปกรณ์ที่จะใช้ในการทำงาน

โดยที่ ECBUS DRIVER จะทำงานในข้อกำหนดดังนี้

PERSONAL COMPUTER : IBM-PS/V OR A COMPATIBLE TYPE

PROCESSOR : I 80386SX, I 80486SX/DX/DX2 32-BIT COMPUTER (A NUMERICAL OPERATION PROCESSOR IS REQUIRED FOR SX)

MAIN MEMORY : 8-MB RAM OR LARGER

HARD DISK : 80-MB FREE SPACE OR LARGER

OS : MICROSOFT WINDOWS VER 3.1

CRT : VGA OR XGA (DISPLAY)

COMMUNICATION PORT: ONE OR MORE RS-232C SERIAL COMMUNICATION PORTS RS-232C/RS-485 CONVERTER (COMM)

สายไฟ (CABLE) :

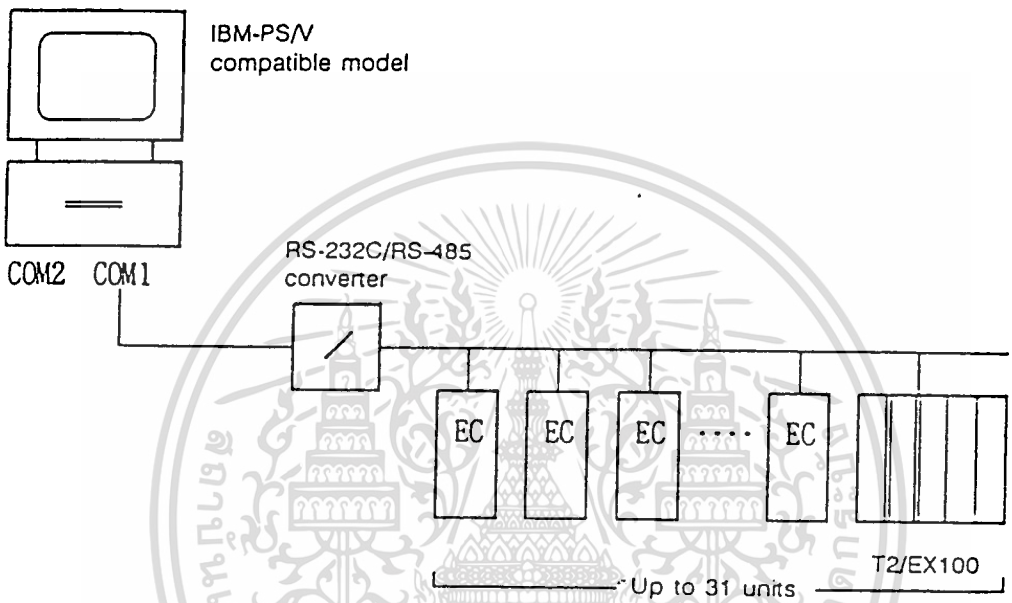
-ระหว่าง คอมพิวเตอร์ และ RS-485 CONVERTER (COMM)

-ระหว่าง คอมพิวเตอร์ และ ตัวควบคุม (AECI)

ข้อกำหนดของระบบ

ในที่นี้จะกล่าวถึงข้อกำหนดของระบบ ในรูปแบบของ COMM กับตำแหน่งของคอมพิวเตอร์ตัวหลัก โดยที่ PORT สามารถจะต่อกับตัวควบคุม (controller)ได้มากถึง 31 ตัว ในการติดต่อแบบ N:N

โดยข้อมูลจะถูกส่งจาก COM PORT ของคอมพิวเตอร์โดยมี RS-232 C/RS-485 เป็นตัวกลับสัญญาณตามลักษณะการติดต่อดังรูป



ซึ่งรายละเอียดการต่อมีดังนี้

CONNECTABLE CONTROLLER

EC 310,EC311,EC320,EC321,EC311 EX100

(RS-485 INTERFACE MODULE สำหรับการติดต่อสำหรับตัวควบคุมในตระกูล EC 300, และ MODULE การพิเศษสำหรับการติดต่อ (ML11) สำหรับ EX 100)

จำนวนของตัวควบคุมที่จะต่อกับ PORT

EC 300 SERIES : 2 < < 31

T 2 , EX 100 : 0< < 3 (OF 31 UNI

TRANSMISSION SPEED

19200 , 9600 (โดย 19200 BPS ใช้สำหรับคอมพิวเตอร์ 80486/50 MHz หรือมากกว่า

โหมดการทำงาน (OPERATION MODE)

ใน ECBUS และ ECBUS/H PROTOCOLS โดย FIX DMACS สำหรับ WINDOWS จะทำงานในโหมดสำรอง เมื่อบันเริ่มทำงาน และจะเปลี่ยนเป็นโหมดหลักโดยการเปลี่ยนแปลงตำแหน่งหลัก (MASTER STATION) จากตัวควบคุม (CONTROLLER)

คุณลักษณะการทำงาน

ในการตั้ง ECBUS DRIVER จะมีภาพแสดงในเมนูโดยในเมนู DRIVER จะมีรูปอธิบายโปรแกรม ใน EC BUS DRIVER เป็นการค้นหาจาก SCU(SYSTEM CONFIGURATION UTILITY) ในระบบ FIX DMACS

โดยโปรแกรม DID (DRIVER IMAGE DEFINETION) จะกำหนดคุณสมบัติอ้างอิงถึงข้อมูลบนเครื่องควบคุมและข้อมูล (DATA - RETRIEVAL) จะถูกสร้างโดยไฟล์ DID

โดยไฟล์ DID จะเป็นโหลดหน่วยความจำจาก DIT (DRIVER IMAGE TABLE) ที่ทำงานในระบบ FIX DMACS เมื่อ SAC เริ่มทำงาน POLLTABLE ใน DIT จะอ้างโดย DBB (DATABASE BUILDER) ECBUS DRIVER จะอ่านข้อมูลที่อ้างอิงบนตัวควบคุมอื่น ๆ จากตาราง POLL TABLE นี้ หรือเขียนข้อมูลบนตาราง POLL TABLE ตามที่เราต้องการจะเขียน

ในการเขียนหรือกำหนดตัวอักษรบน "DRV" สำหรับ DRIVER

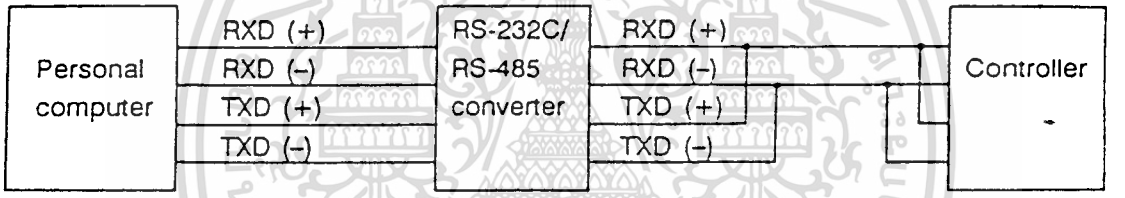
TEC : (COMM)

TEA : (AECI)

การต่อสาย (COMM)

สำหรับการต่อสายเคเบิล จะอ้างอิงถึงคู่มือการใช้งานทางด้านฮาร์ดแวร์ของตัวควบคุมแบบลูฟขบวบการควบคุมในรุ่น EC 320 และ PORT RS-232C จะได้รับการสนับสนุนจาก IBM SERIAL PORT

เมื่อต่อสายกับ RS-232C /RS-485 , RXD, TXD และ GND บนคอมพิวเตอร์ด้านข้างก็จะต่อด้วย RS-232C



รูปแบบการต่อสายระหว่าง PC และตัวควบคุม

ในการต่อคอมพิวเตอร์กับตัวควบคุมนั้นเราต้องมี RS-485 TRAMSMISSION INTER FACE MODULE ในตัวควบคุมเสียก่อนเพราะในส่วนของบอร์ดนี้เป็น OPTION พิเศษสำหรับการติดต่อสื่อสาร ซึ่งไม่เกี่ยวกับการควบคุมแบบ SINGLE LOOP หรือใช้ตัวควบคุมแบบ ลูฟขบวนการควบคุมธรรดาซึ่งไม่มีบอร์ดนี้ก็ยังสามารถทำงานได้ตามปกติ

โดยที่ RS-485 จะต่อกับ EC 320 ซึ่งเป็นตัวควบคุมแบบลูฟขบวนการควบคุมโดยจะต่ออยู่ภายในตัวควบคุม

คุณสมบัติของ RS-485

1. สัญญาณ : COMPLIANT WITH RS-485
2. TRANSMISSION PROTOCOL : REFER TO "MODEL NUMBER TABLE"
3. TRAMSMISSION SPEED : 300, 1,200,2,400,9600 และ 19200 BAUD
4. DISTANCE : 1 KM (MAX)
5. NO. OF UNITS TO BE CONNECTED : 32
6. TRAMSMISSION MODE : 1:N OR N:N
7. INPUT/OUTPUT INSULATION
   จะได้รับสัญญาณ INPUT จากภายนอกสำหรับตัวควบคุม
8. POWER SUPPLY
   จะได้รับ POWER SUPPLY จากภายในของตัวควบคุม
9. OPERATION CONDITIONS
   ทำงานที่อุณภูมิ : 0-50 C
   ทำงานที่ความชื้นสัมพัทธ์ : 10-90 %
10. คุณสมบัติอื่น ๆ
    น้ำหนัก : 100 กรัม
    ขนาด : 57x117x25 mm.

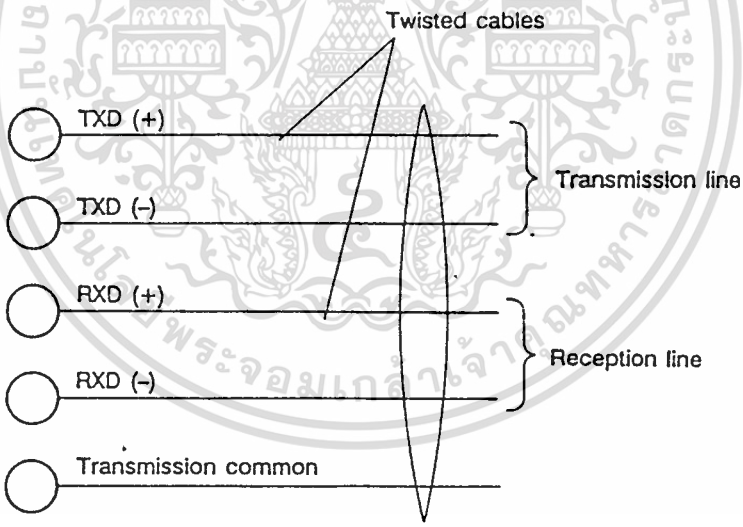
MODEL NUMBER TABLE

MODEL NUMBER CODE	DESCRIPRTION
3Y8A1120G001	COMPLIANT WITH THE MODBUS PROTOCOL (1:N)
3Y8A1120G003	ECBUS PROTOCOL (N:N OR 1:N)

สายไฟให้ใช้สาย SHIELDED, TWISTED CABLE (AWG24) สำหรับการรับส่งข้อมูล

การต่อสายภายนอก

ให้อ้างถึงคู่มือการใช้งานของตัวควบคุม สำหรับการต่อสายภายนอก



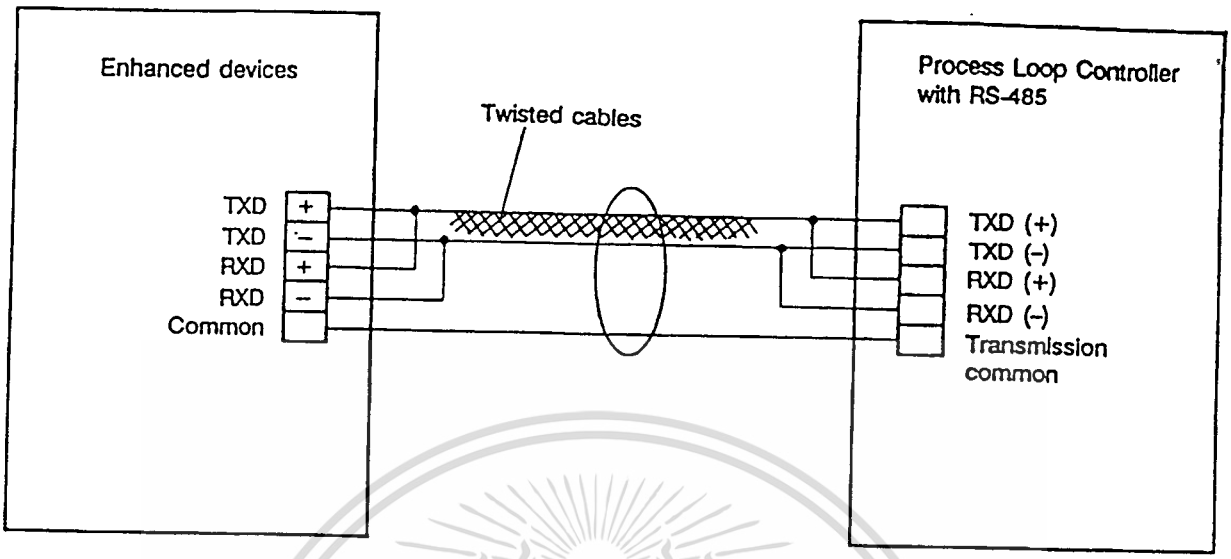
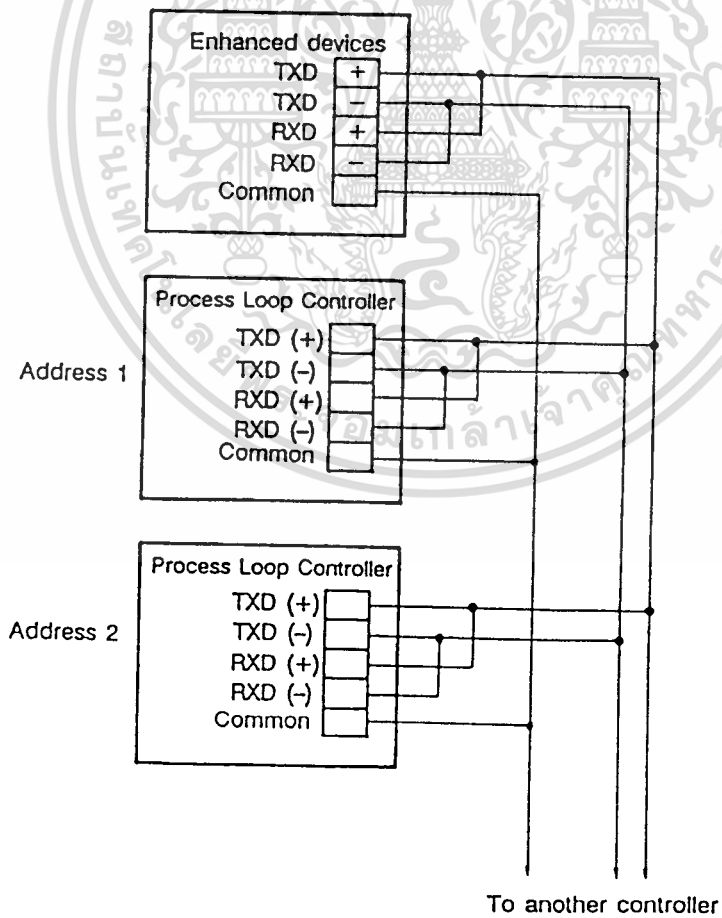


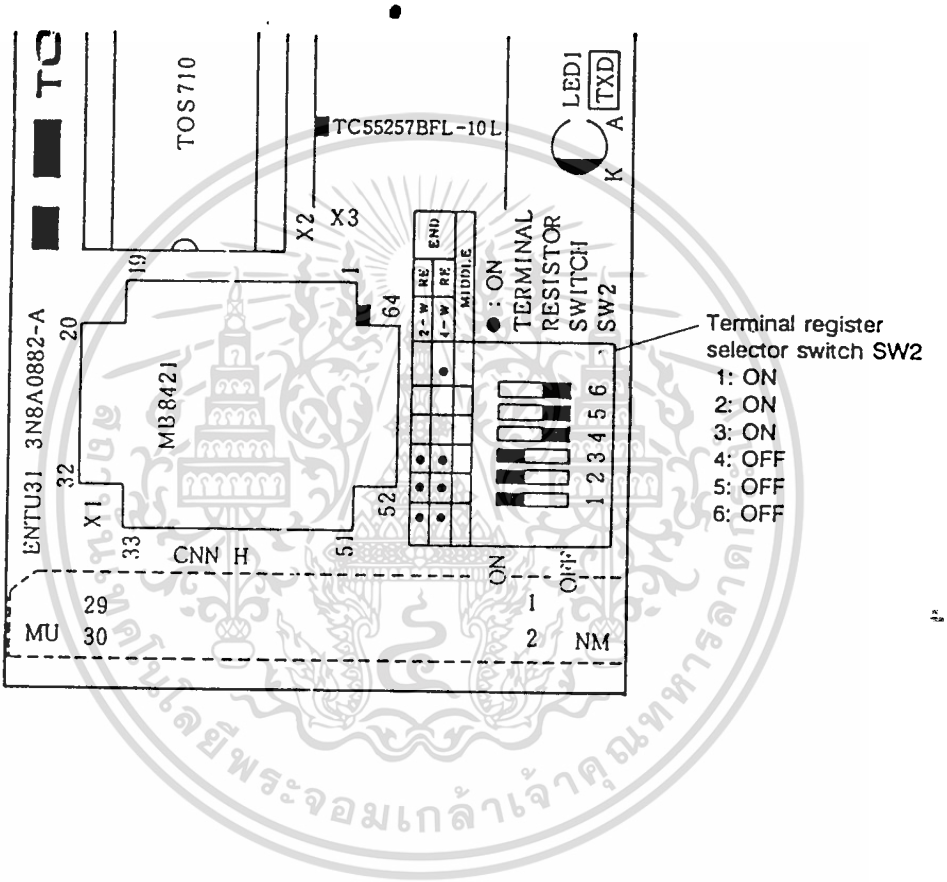
Figure 5.2 1:1 Connection



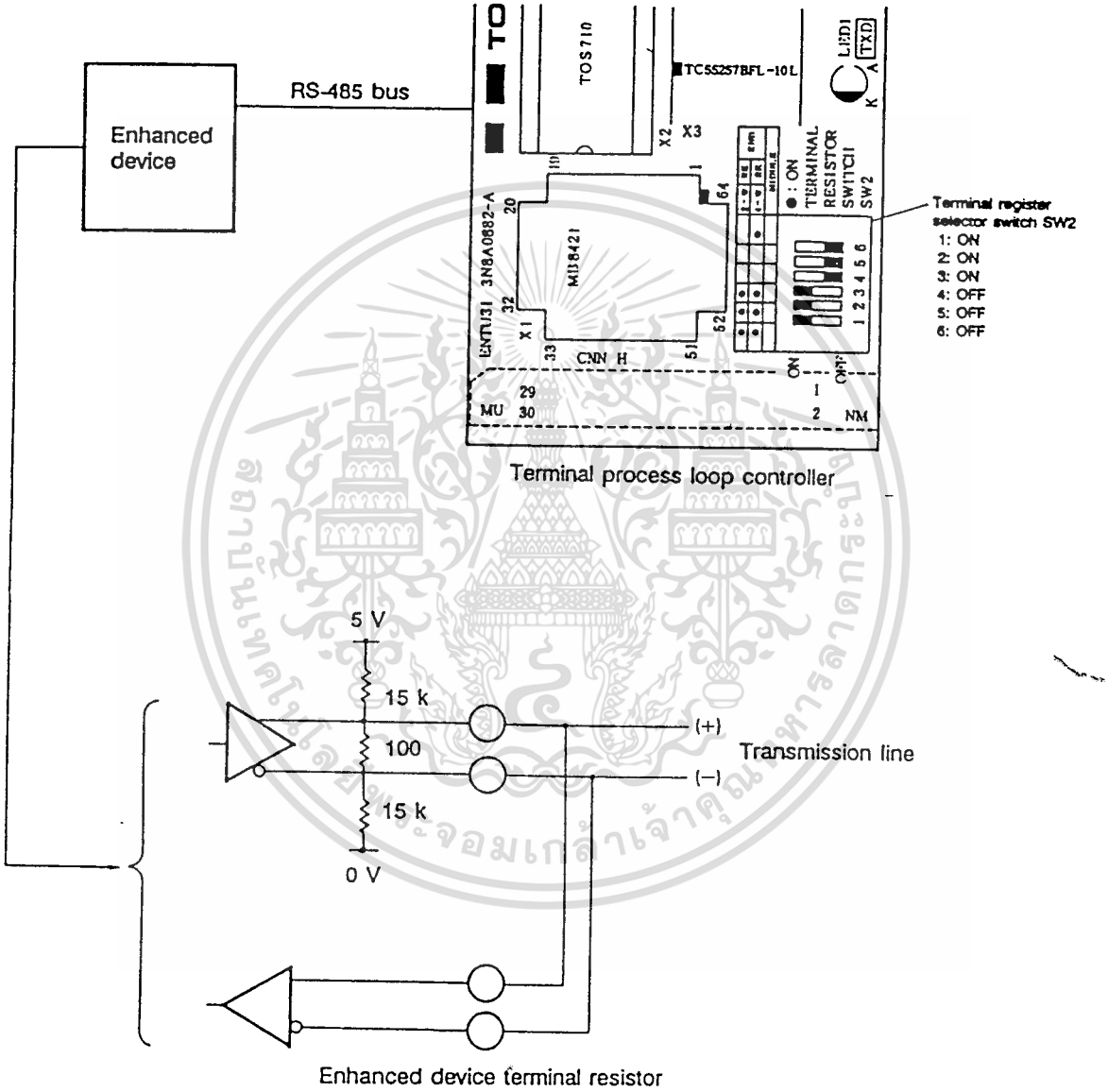
Bus Type Connection

วิธีการตั้งสวิทช์ (DIP SWITC) สำหรับ RS485 ในการควบคุมแบบ SINGLE LOOP

เมื่อเราต้องการที่จะควบคุมแบบลูฟเดียว เราต้องตั้งสวิทช์ (DIP SWITCH) บนบอร์ด RS-485 ดังรูป



ระบบสำหรับการต่อโดยตรงกับอุปกรณ์เสริม

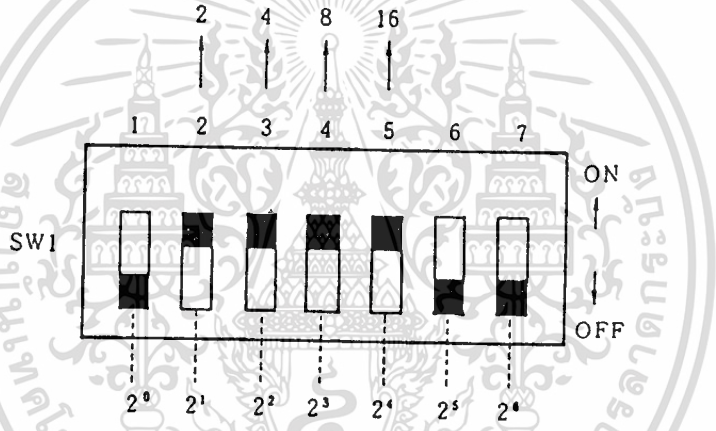


การตั้งสวิทช์ (SWITCH SETTING)

โดยการตั้งตำแหน่ง (ADDRESS) ของสวิทช์ 1 ที่จะใช้กับอุปกรณ์ที่จะต่อ กับ BUS RS-485 โดยมันสามารถตั้งได้ตั้งแต่ 1 ถึง 127 แต่ในการต่อตัวควบคุมมากที่สุดคือ 31 ตัวควบคุม โดยในการตั้งค่าสามารถตั้งได้ดังนี้ ถ้าเราใช้สวิทช์ตัวไหนก็ให้ตั้งเป็น "1" หรือ ON ถ้าเราไม่ใช้ก็ให้ตั้งเป็น "0" หรือ OFF โดยรายละเอียดของสวิทช์ 1 ดูได้จากรูป

Switch No.	Address No.
SW1-1	1
SW1-2	2
SW1-3	4
SW1-4	8
SW1-5	16
SW1-6	32

Example: Selection of address "30" with Process Loop Controller



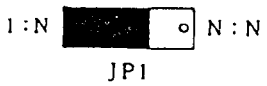
ส่วนการตั้งพินสวิทช์ (JP1)

ใช้สำหรับการติดตั้ง TRANSMISSION PROTOCOL ซึ่งมีอยู่ 2 ลักษณะ

คือ - MODBUS (FOR 1:N TRANSMISSION)

-ECBUS (FOR N:N TRANSMISSION)

โดยการตั้งสวิทช์ดังรูป



การใช้ FIX DMACS และอุปกรณ์ตัวควบคุมทดลองใช้กับการควบคุมต่าง ๆ

ซึ่งในที่นี้เราได้ทกลองการควบคุมกับชุดทดลองโดยการสร้างระบบควบคุมแบบง่ายโดยเราได้ศึกษาการควบคุมแบบวัดระดับ (LEVEL CONTROL) การควบคุมการไหล (FLOW CONTROL). โดยเราได้ศึกษาถึงระบบควบคุมแบบ CASCADE CONTROL และการควบคุมแบบ FEEDFORWARD

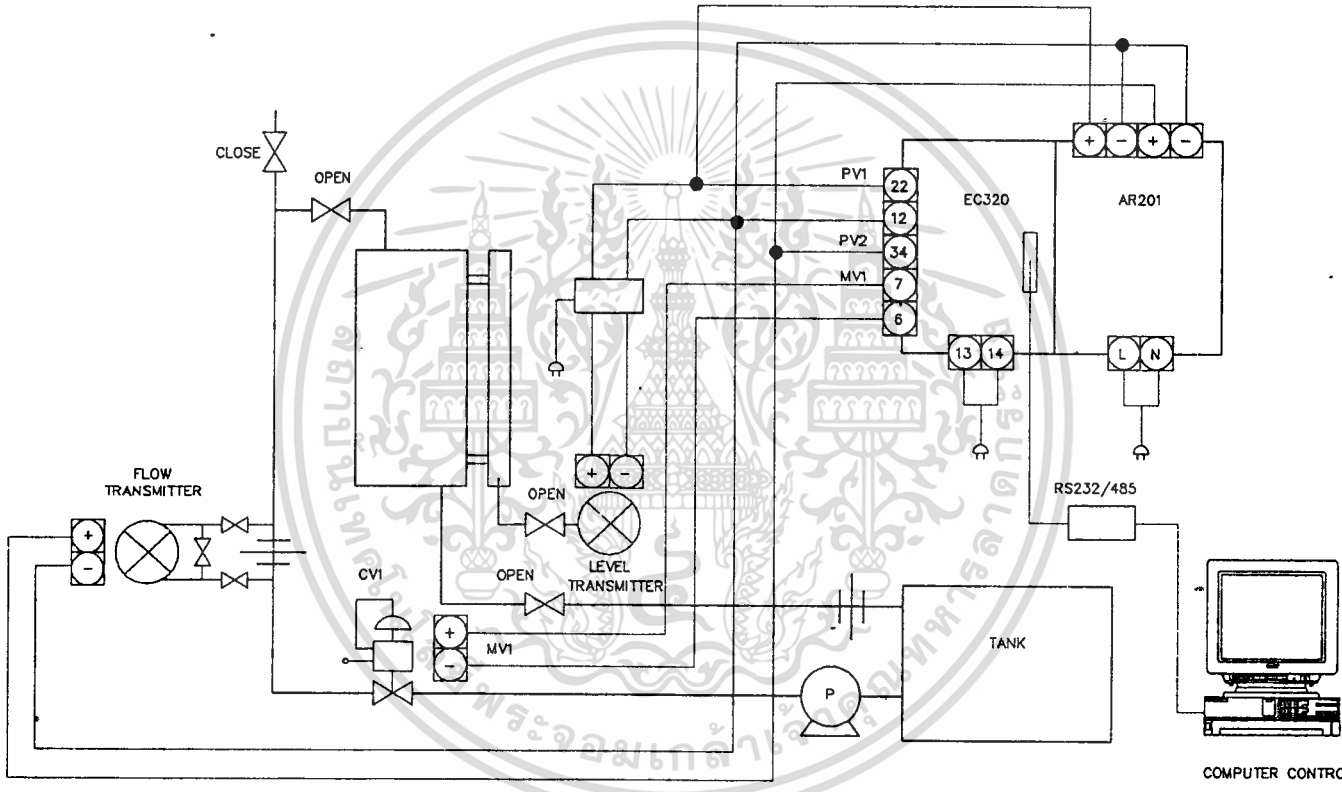
CASCADE CONTROL

ระบบควบคุมแบบ CASCADE ประกอบไปด้วยตัวควบคุมหลัก (MASTER CONTROL) และตัวควบคุมรอง (SLAVE CONTROLLER) โดยตัวควบคุมหลักจะทำหน้าที่รักษาค่าของตัวแปรที่ถูกควบคุมให้เป็นไปตามต้องการส่วนตัวควบคุมรองจะทำหน้าที่คตวบคุมตัวแปรอื่นที่สามารถทำให้ตัวแปรที่ถูกควบคุมเปลี่ยนแปลงค่าได้ ดังนั้นสัญญาณควบคุมของตัวควบคุมหลักจะเป็นค่าที่กำหนดไว้ (SET POINT) ของตัวควบคุมรอง และนั่นก็คือไปควบคุมการทำงานของวาล์วควบคุมในทางอ้อมด้วย

โดยจุดมุ่งหมายของระบบควบคุมแบบ CASCADE ก็เหมือนกับตัวควบคุมแบบ ลูฟเดียวทั่ว ๆ ไป คือ ทำให้เกิดความสมดุลย์ระหว่างพลังงานไหลเข้ากับพลังงานไหลออกและทำให้ตัวแปรที่ถูกควบคุมมีค่าคงที่ตามต้องการ อย่างไรก็ตามลูฟที่ 2 ของระบบควบคุมแบบ CASCADE จะทำให้ LAG ในกระบวนการมีค่าลดลง ดังนั้น การไหลเข้าที่มีเสถียรภาพที่ดีจะทำให้การทำงานทั้งหมดมีความเที่ยงตรงมากขึ้น ตัวควบคุมารองอาจถือได้ว่าเป็นอุปกรณ์ควบคุมสุดท้ายที่มีความประณีตละเอียดมากและถูกควบคุมโดยตัวควบคุมหลัก (ทำนองเดียวกับระบบควบคุมที่มีตัวควบคุมซึ่งเป็นตัวกำหนดการทำงานของวาล์วควบคุมเพียงตัวเดียว) และตัวแปรที่สองนี้จะเป็นเสมือนตัวกลางของการควบคุม ตัวอย่างเช่น ถ้าตัวควบคุมรองนี้คือตัวควบคุมการไหลนั่นก็คือ ตัวควบคุมหลักจะไม่ได้เป็นตัวบังคับหรือกำหนดตำแหน่งของวาล์วแต่จะไปควบคุมการไหลแทน

ในการเซ็ทค่า SP,MV หรือ ค่าพารามิเตอร์ในการควบคุมต่างๆเช่น KP,TI,TD,RH,RL เป็นต้น สามารถเซ็ทค่าโดยเซ็ทที่ตัวควบคุม (CONTROLLER) และที่คอมพิวเตอร์ได้ แต่ในการแสดงผล (DISPLAY) แล้ว ในส่วนของคอมพิวเตอร์สามรถที่จะแสดงผลหรือสภาวะการทำงานได้มากกว่าตัวควบคุมการทำงาน (CONTROLLER)ซึ่งการแสดงผลนี้ขึ้นอยู่กับ SOFTWAREในการสร้างภาพหรือSOFTWARE ในการ LINK ระหว่าง PAGE แต่ละ PAGE ที่เราได้ทำการสร้างขึ้นมา เช่น การแสดงผลของ ALARM SUMMARY ทางคอมพิวเตอร์จะมีส่วนดีตรงทีเราสามารถที่จะทราบถึงสภาวะที่ผ่านมาหรือบันทึกเป็น ERCORD แล้ว REPORT ให้ทราบ ส่วนการแสดงผลของตัวควบคุมจะเกิดในช่วงขณะทีมันเกิดการผิดพลาดในการควบคุมเมื่อสภาวะนั้นหายไปเราก็จะไม่ทราบถึงเหตุผลของการเกิดข้อผิดพลาดในการควบคุมเป็นต้น

โดยการทดลองเราจะให้การควบคุมแบบ CASCADE โดยประกอบด้วยตัวควบหลัก (MASTER CONTROLLER) เป็นตัวควบคุมระดับ (LIC) และตัวควบคุมรอง (SLAVE CONTROLLER) คือตัวควบคุมอัตราการไหล (FIC) ดังรูปข้างล่างนี้

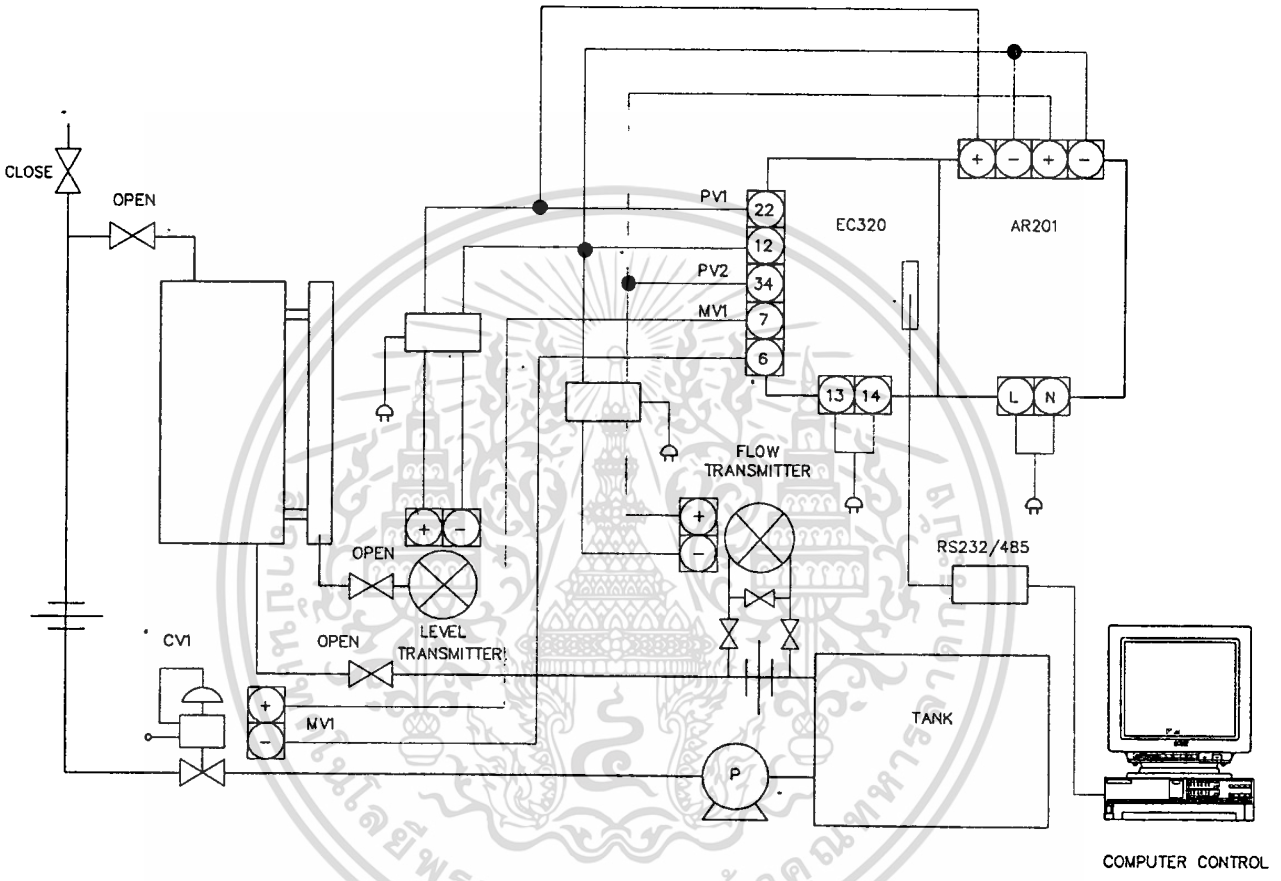


FEEDFORWARD CONTROL

ในการควบคุมระดับนั้นการควบคุมแบบ FEEDFORWARD สามารถประยุกต์ใช้ในระบบควบคุมเพื่อนำให้เกิดความสมดุลย์ระหว่างพลังงานไหลเข้ากับพลังงานไหลออกแต่ก่อนที่จะประยุกต์ใช้การควบคุม แบบ FEEDFORWARD กับกระบวนการใด ๆ นั้น ขั้นแรกจะต้องเข้าใจถึงกระบานการให้สมบูรณ์เสียก่อนซึ่งสามารถทำได้โดยการสร้างสมการอธิบายกระบวนขึ้นมา จากนั้นจัดสมการให้อยู่ในรูปแบบที่สามรถจะหาคำตอบได้ด้วยอุปกรณ์การวัดและควบคุมที่มีอยู่ นอกจากนี้ยังต้องมีการจัดเตรียมอุปกรณ์เพื่อชดเชยการ LEAD และ LAG ที่จะเกิดขึ้นจากอุปกรณ์ในกระบวนการซึ่งเป็นสิ่งที่หลีกเลี่ยงไม่ได้ ถ้าหากสามารถจัดเตรียมอุปกรณ์ต่าง ๆ ได้สมบูรณ์ ก็จะพบว่าระบบควบคุมแบบ FEEDFORWARD จะเป็นระบบควบคุมกระบวนการที่ดี แต่ในทางปฏิบัติแล้วอาจจะเกิดมีค่าความคลาดเคลื่อนเกิดขึ้นในช่วงเวลาหนึ่ง หรือ อาจจะมีคุณสมบัติของความไม่เป็นเชิงเส้นเพิ่มขึ้นมาซึ่งจะทำให้การควบคุมยากยิ่งขึ้น อย่างไรก็ตามปัญหาเหล่านี้สามารถแก้ไขได้โดยเพิ่มการป้อนกลับให้กับระบบควบคุมแบบ FEEDFORWARD ผลก็คือทำให้ระบบควบคุมแบบ FEEDFORWARD เป็นระบบที่ดีที่สุด

อย่างไรก็ตามระบบควบคุมแบบ FEEDFORWARD นี้ค่อนข้างจะยุ่งยาก จึงทำให้มีราคาแพงกว่าระบบควบคุมแบบป้อนกลับมาก นอกจากนี้การปรับค่าต่าง ๆ ในระบบควบคุมป้อนกลับนั้นจะทำได้ง่ายกว่าระบบควบคุมแบบ FEEDFORWARD ในทางปฏิบัติแล้ว โดยทั่วไปจึงใช้การควบคุมแบบป้อนกลับ และจะใช้การควบคุมแบบ FEEDFORWARD เมื่อกระบวนการที่จะควบนั้นต้องการความเที่ยงตรงสูงเท่านั้นอนึ่งระบบแบบนี้มีผลตอบสนองที่ช้าและ DEAD TIMES ที่นานตลอดจนเป็นระบบดีอยู่ยากและซับซ้อนด้วย

รูปแบบการต่อทดลองแบบ FEEDFORWARD CONTROL



ภาคผนวก

FIX Key Macro Report
File: DRAW.KMR

<CTRL-N>

1: MENU NEW

<CTRL-O>

1: MENU OPEN

<CTRL-S>

1: MENU SAVE

<CTRL-P>

1: MENU PRINT

<CTRL-Z>

1: MENU UNDO

<CTRL-X>

1: MENU CUT

<CTRL-C>

1: MENU COPY

<CTRL-V>

1: MENU PASTE

<CTRL-N>

1: MENU NEW

<CTRL-O>

1: MENU OPEN

<CTRL-S>

1: MENU SAVE

<CTRL-P>

1: MENU PRINT

<CTRL-Z>

1: MENU UNDO

<CTRL-X>

1: MENU CUT

<CTRL-C>

1: MENU COPY

<CTRL-V>

1: MENU PASTE

<DELETE>

1: MENU DELETE

<CTRL-D>

1: MENU DUPLICATE

<CTRL-Y>

1: MENU ZOOM_TO

<CTRL-B>

1: MENU SEND_TO_BACK

<CTRL-F>

1: MENU BRING_TO_FRONT

FIX Key Macro Report
File: VIEW.KMR

<CTRL-O> Open a new picture

1: MENU OPEN

<CTRL-C> Close the current picture

1: MENU CLOSE

<CTRL-N> Move on to NEXT picture

1: MENU NEXT

<CTRL-P> Move on to PREVIOUS picture

1: MENU PREVIOUS

<CTRL-L> Return to the LAST picture

1: MENU LAST

<CTRL-Y> Zoom to designated region

1: MENU ZOOM_TO

<CTRL-D> Return the current picture to default

1: MENU DEFAULT_VIEW

<CTRL-K> Acknowledge a single alarm

1: MENU ACKNOWLEDGE_ONE

<CTRL-E> Enable/Disable Alarm Checking

1: MENU ENABLE_DISABLE_ALARMS

<CTRL-H>

1: MENU SILENCE_HORN

<CTRL-R>

1: MENU TOGGLE_HORN

<CTRL-T> Display BLOCK info for selected link

1: MENU TAG_STATUS

<CTRL-M> Place current link's BLOCK in MANUAL

1: MENU MANUAL_MODE

<CTRL-A> Place current link's BLOCK in AUTO

1: MENU AUTO_MODE

<CTRL-F> Place current link's BLOCK OFF SCAN

1: MENU OFF_SCAN

<CTRL-S> Place current link's BLOCK ON SCAN

1: MENU ON_SCAN

<PGDN> Move on to NEXT picture

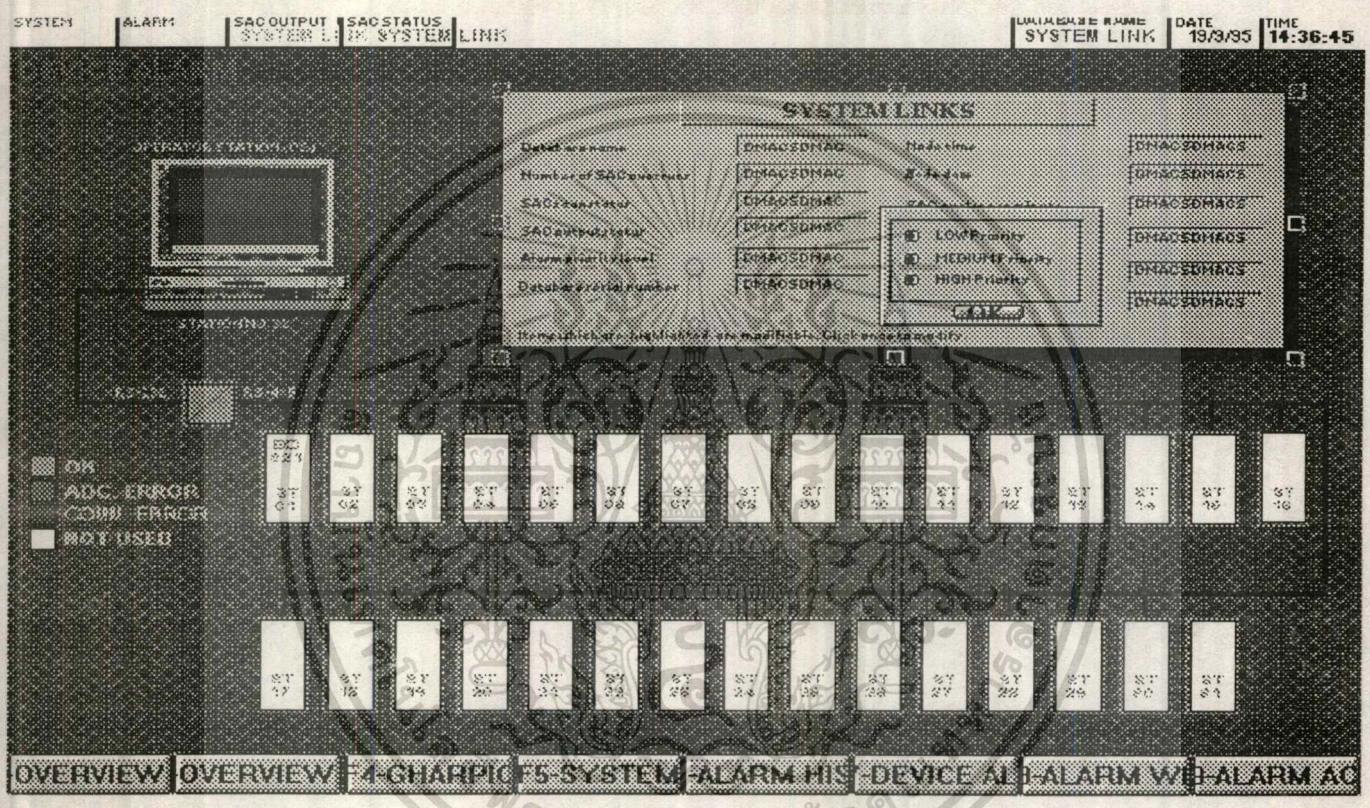
1: MENU NEXT

<PGUP> Move on to PREVIOUS picture

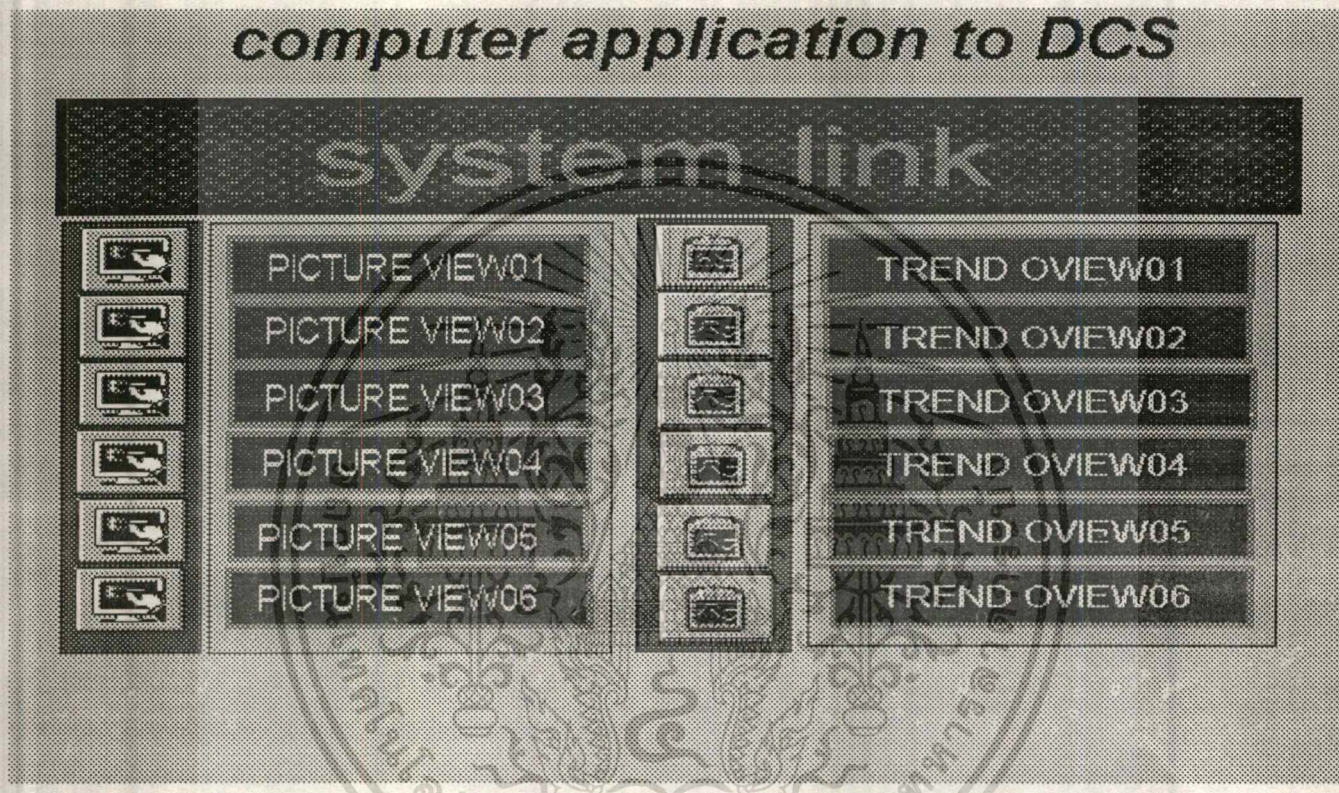
1: MENU PREVIOUS

<CTRL-F1> user defined hypertext help / link

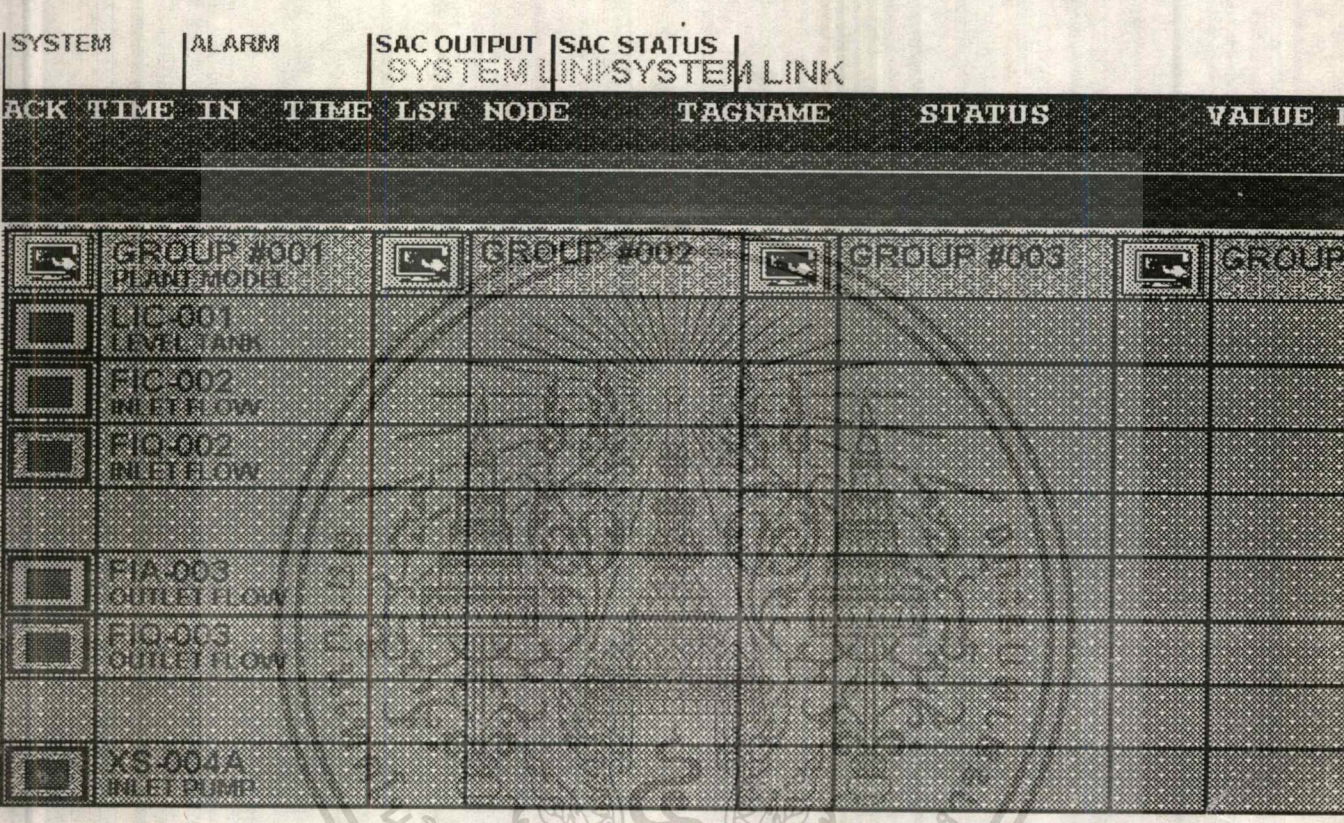
รูปภาพของการติดต่อของระบบ FIX WDMACS



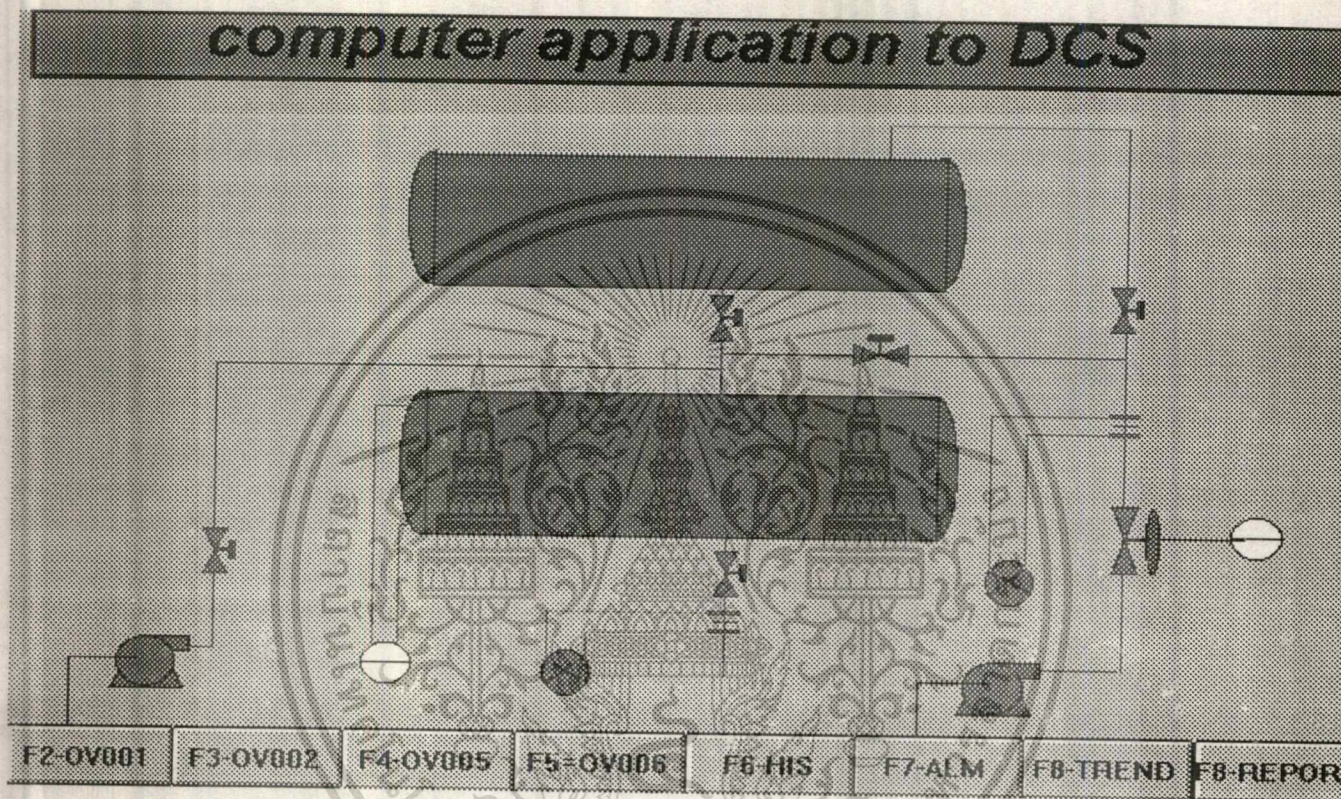
รูปภาพการติดต่อระหว่างรูปแต่ละรูป



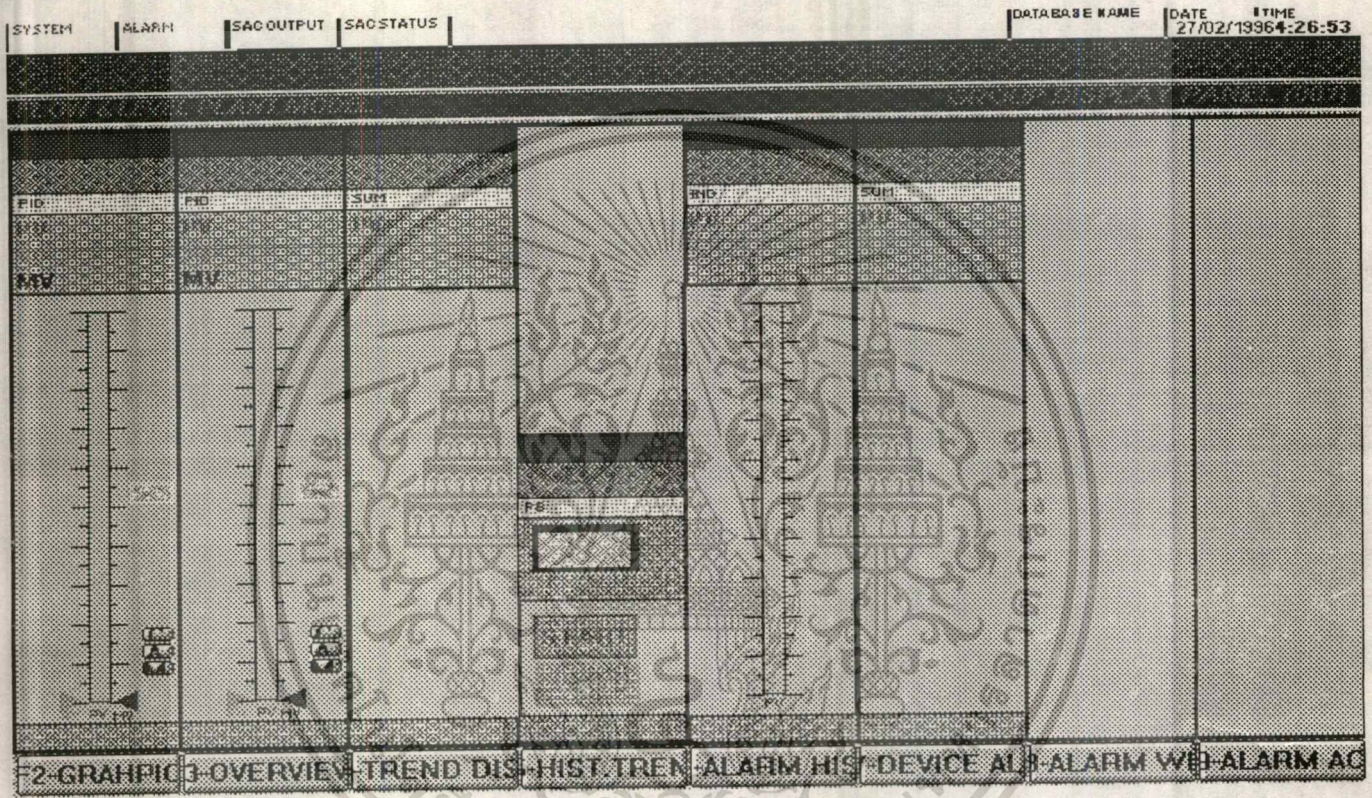
รูปภาพของการแสดงสภาวะการทำงานของลูฟคอนโทรล



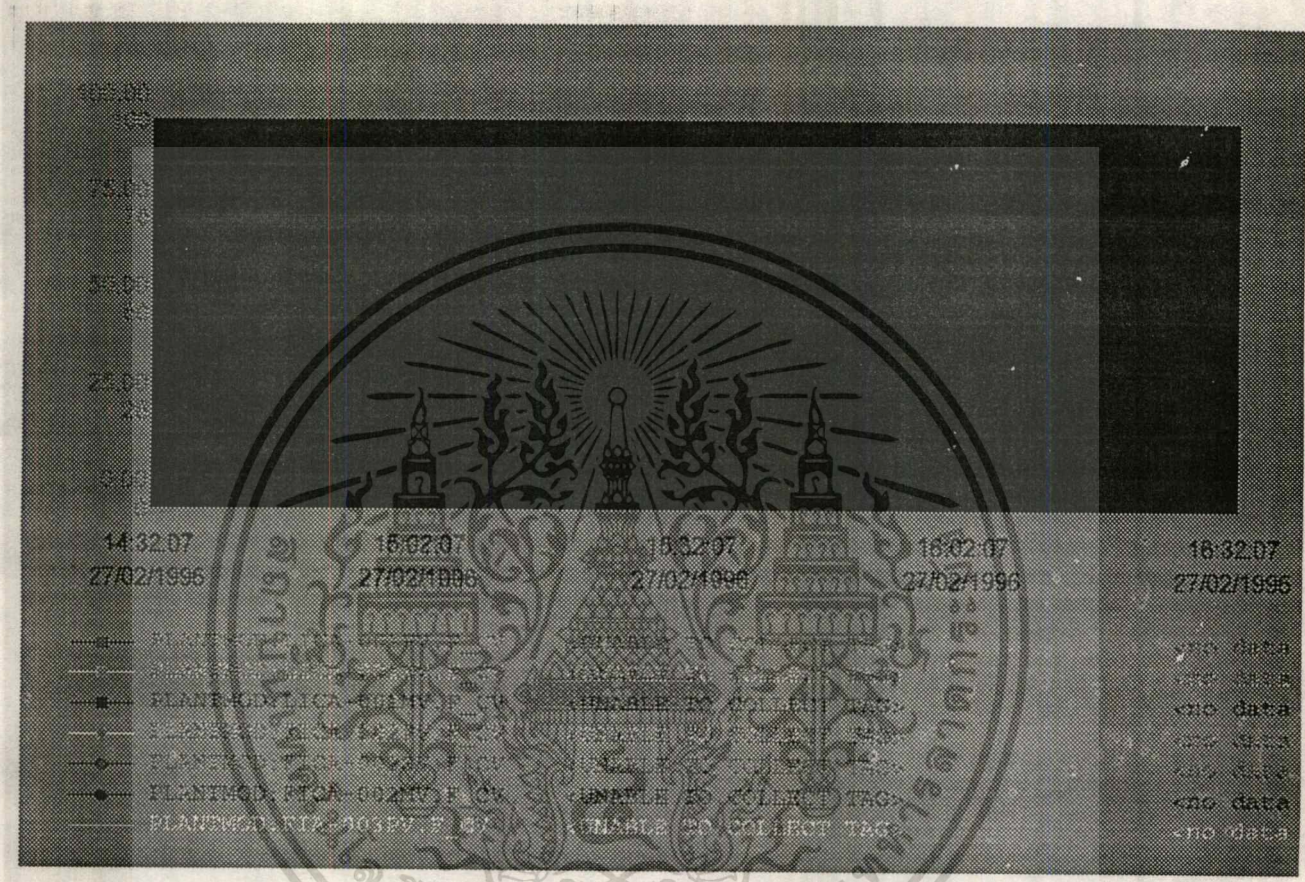
รูปภาพของขบวนการการควบคุมการทำงาน



รูปภาพของการควบคุมโดยการใช้ภาพของคอนโทรลเลอร์



รูปภาพของการแสดงสภาวะการทำงาน



บรรณานุกรม

INTELLUTION : THE FIX DMACS MANUAL : INTELLIGENT SOLUTION

TOSHIBA : EC375, ECBUS DRIVER FOR FIX DMACS FOR WINDOWS : TOSHIBA CORPORATION

TOSHIBA : เทคนิคและการใช้งานตัวควบคุม EC320 : บริษัท คอนโทรลอจิค จำกัด

คู่มือทดลองแพลนท์โมเดล : บริษัท คอนโทรลอจิค จำกัด