การควบคุมแบบ ICS โดยผ่านระบบเชื่อมต่อแบบ I^2C

INTEGRAL CONTROL SYSTEM BY INTER-IC COMMUNICATION

โดย

นางสาวกฤษณา เคนหาราช 39014013

นายบัณฑิต มาชูวงศ์สกุล 39014278

นายสิทธิชัย หวังศิริรุ่งเรือง 39014572

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมการวัดคุม

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2542

ปริญญานิพนธ์ ปีการศึกษา 2542

ภาควิชาวิศวกรรมการวัดคุม

คณะ วิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การควบคุมแบบ ICS โดยผ่านระบบเชื่อมต่อแบบ I^2C

INTEGRAL CONTROL SYSTEM BY INTER-IC COMMUNICATION

ผู้จัดทำ

1. นางสาวกฤษณา เคนหาราช รหัสประจำตัว 39014013
2. นายบัณฑิต มาชูวงศ์สกุล รหัสประจำตัว 39014278
3. นายสิทธิชัย หวังศิริรุ่งเรือง รหัสประจำตัว 39014572

.....อาจารย์ที่ปรึกษา

(อาจารย์อาจินต์ น่วมสำราญ)

การควบคุมแบบ ICS โดยผ่านระบบเชื่อมต่อแบบ I^2C

นางสาวกฤษณา เคนหาราช
นายบัณฑิต มาชูวงศ์สกุล
นายสิทธิชัย หวังศิริรุ่งเรือง
อาจารย์อาจินต์ น่วมสำราญ อาจารย์ที่ปรึกษา
ปีการศึกษา 2542

บทคัดย่อ

ในวิทยานิพนธ์ฉบับนี้ เป็นเรื่องของการควบคุมแบบ ICS โดยผ่านระบบเชื่อมต่อแบบ I^2C โดยใช้คอมพิวเตอร์เพียงเครื่องเดียวในการควบคุมกระบวนการที่มากกว่า 1 กระบวนการ โดยมีการแปลงมาตรฐาน RS-232 เป็นระบบบัส I^2C ทำให้เกิดความสะดวกและไม่ยุ่งยาก เนื่องจากใช้สายสัญญาณเพียงสองเส้น คือ สาย SDA และ SCL ในการควบคุม ในการควบคุมแบบนี้สามารถนำไปใช้ในการควบคุมระบบที่ซับซ้อนต่อไป

INTEGRAL CONTROL SYSTEM BY INTER-IC COMMUNICATION

Miss. Kritsana Kenharaj

Mr. Bundit Machoowongsakun

Mr. Sitichai Vangsirirungruang

Mr. Arjin Numsoomran Advisor

Abstract

This thesis is presents an application of Integral Control System by I^2C communication system.This system shows that many processes can be control by only one computer.The standard from of data in RS-232 will be convert into I^2C standard. Therefore, the convenience can be obtained since only two control lines are used, which are SDA and SCL. This kind of controller can be applied to wire complicated systems.

สารบัญ

สารบัญรูปภาพ

สารบัญตาราง

หน้า

บทที่ 1

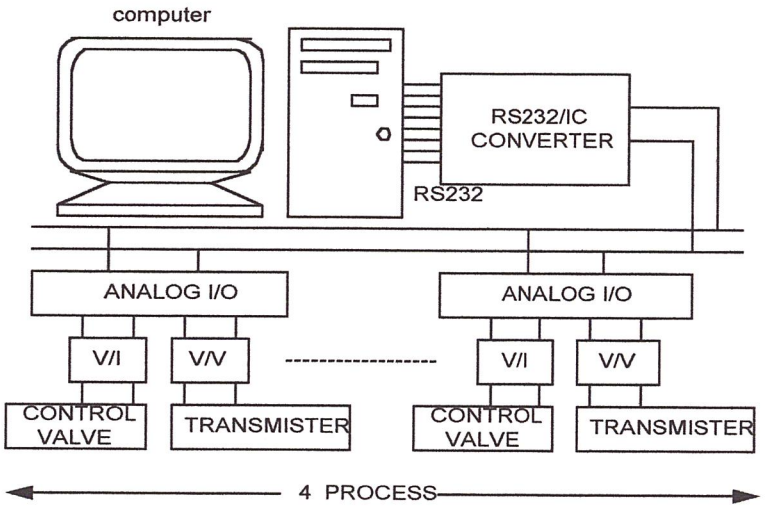
บทนำ

ปัจจุบันมีการนำเอาคอมพิวเตอร์มาใช้ในการควบคุมกระบวนการ แต่ไม่คอยมีใครใช้มาตร-ฐานการเชื่อมต่อแบบ I^2C เราจึงให้ความสนใจ ที่จะศึกษา และนำมาใช้ในการควบคุมกระบวนการ

<u>วัตถุประสงค์</u>

1. เพื่อศึกษามาตรฐานการเชื่อมต่อแบบ I^2C
2. นำมาตรฐานการเชื่อมต่อแบบ I^2C มาควบคุมกระบวนการแบบ ICS

<u>ขอบเขตของโครงงาน</u>



รูปที่ 1.1 แสดงขอบเขตของโครงงาน

- ทางด้านฮาร์ดแวร์

จะทำบอร์ด Input/Output ซึ่งประกอบด้วย

1) ตัวแปลงแรงดันเป็นแรงดัน
2) ตัวแปลงแรงดันเป็นกระแส
3) IC PCF8591
4) แหล่งจ่ายไฟกระแสตรง

- ทางด้านซอฟแวร์

จะเขียนโปรแกรมเพื่อควบคุมกระบวนการ 4 กระบวนการแบบ ICS โดยใช้โปรแกรมภาษา Visual Basic

<u>ประโยชน์ที่คาดว่าจะได้รับ</u>

เพื่อเป็นข้อมูลแก่ผู้ที่สนใจนำไปพัฒนาในการควบคุมกระบวนการอื่นที่ซับซ้อนมากกว่า และในอนาคตอาจจะมีการนำระบบ มาใช้ในการควบคุมอย่างกว้างขวาง

บทที่ 2
การศึกษาทฤษฎีต่างๆที่เกี่ยวข้องกับโครงงาน

ในบทนี้เราจะกล่าวถึงทฤษฎีต่างๆที่จะนำมาใช้ทำโครงงานดังนี้

2.1 รูปแบบการสื่อสารแบบอนุกรม

2.2 มาตรฐานการเชื่อมต่อแบบ RS-232 เนื่องจากเราใช้คอมพิวเตอร์ ในการควบคุมกระบวนการเราต้องมีการติดต่อกับคอมพิวเตอร์ โดยผ่านพอร์ทอนุกรมและใช้มาตรฐานการเชื่อมต่อแบบ RS-232

2.3 ระบบ I^2C และการแปลงจากมาตรฐาน RS-232 เป็น I^2C จุดที่เราสนใจคือการเชื่อมต่อแบบI^2C ดังนั้นจึงต้องศึกษาถึงรายละเอียดของ I^2C และการแปลงมาตรฐาน RS-232 เป็น I^2C

2.4 ถ้าแปลงสัญญาณอะนาลอกเป็นดิจิตอลและแปลงสัญญาณดิจิตอลเป็นอะนาลอก เนื่องจากตัววัดอุปกรณ์ควบคุมตัวสุดท้าย ส่งสัญญาณและรับสัญญาณเป็นอะนาลอก แต่คอมพิวเตอร์ใช้สัญญาณดิจิตอล จึงต้องมีตัวแปลงสัญญาณ เราใช้ IC PCF8591 เป็นตัวแปลงสัญญาณ

2.5 ตัวแปลงแรงดันเป็นแรงดัน และตัวแปลงแรงดันเป็นกระแส เนื่องจากสัญญาณแรงดันมาตรฐานของอุปกรณ์วัดจะเป็น1-5 โวลต์ แต่ IC PCF8591 จะรับสัญญาณได้ 0-5 โวลต์ เพื่อให้ IC PCF8591 ถูกใช้อย่างเต็มที่ จึงมีตัวแปลงแรงดัน 1-5 โวลต์เป็น0-5 โวลต์ และเนื่องจากตัวอุปกรณ์ควบคุมตัวสุดท้ายใช้สัญญาณมาตรฐาน 4-20 มิลลิแอมป์ แต่ IC PCF8591 ให้เอาท์พุทเป็นแรงดัน 0-5โวลต์ ดังนั้นจึงมีตัวแปลงแรงดัน 0-5 โวลต์ให้เป็นกระแส 4-20 มิลลิแอมป์

2.6 โปรแกรม Visual Basic ที่ใช้ติดต่อกับระบบ I^2C และ PCF8591 ส่วนนี้มีความสำคัญเพราะการที่จะควบคุมกระบวนการได้ คอมพิวเตอร์ต้องสามารถรับค่าอินพุทจาก PCF8591 ได้และสามารถส่งสัญญาณควบคุมไปที่ตัว PCF8591 ได้

2.7 ทฤษฎีการควบคุมกระบวนการ จะแบ่งเป็น

- ทฤษฎีการควบคุมแบบป้อนกลับ
- การปรับค่า PID

หลักการและทฤษฎี

2.1 การสื่อสารแบบอนุกรม

การสื่อสารแบบอนุกรมแบ่งได้เป็น 2 แบบคือแบบซิงโครนัสและแบบอะซิงโครนัสแบบซิงโครนัสจะมีสัญญาณนาฬิการ่วมอยู่กับการรับและส่งสัญญาณด้วย ดังนั้นการติดต่อแบบซิงโครนัสนี้ จะต้องใช้สายในการเชื่อมต่อ อย่างน้อยที่สุด 3 เส้น คือ สัญญาณนาฬิกา,ข้อมูลและกราวด์

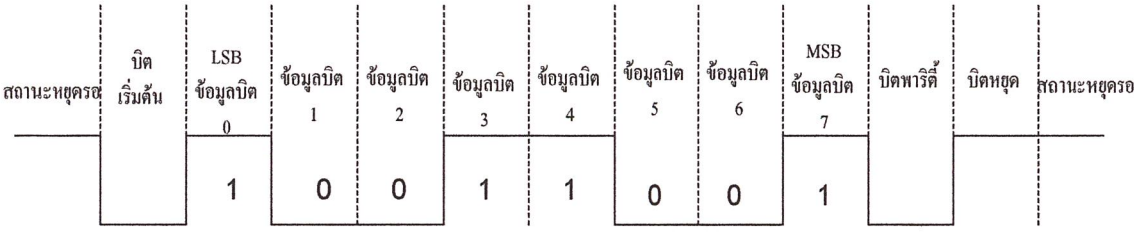
การสื่อสารข้อมูลแบบอะซิงโครนัสการรับและส่งสัญญาณไม่จำเป็นต้องมีสายสัญญาณนาฬิกาแต่จะใช้การกำหนดค่าอัตราเร็วในการรับและส่งข้อมูลให้มีค่าเท่ากัน ซึ่งเรียกอัตราเร็วนี้ว่าอัตราบอด หรือบอดเรต(baudrate) มีหน่วยเป็นบิตต่อวินาที (bit per second:bps)

รูปแบบของข้อมูลที่ใช้ในการรับส่งแบบอะซิงโครนัสประกอบด้วย 4 ส่วนด้วยกันคือ

1. บิตเริ่มต้น(start bit)
2. บิตข้อมูลแบบอนุกรม
3. บิตตรวจสอบพาริตี้(parity bit)
4. บิตปิดท้ายหรือบิตหยุด(stop bit)

เมื่อไม่มีการส่งข้อมูลขา data จะมีสถานะลอจิก "1" เรียกสถานะนี้ว่าสถานะหยุดรอ(waiting stage) การเริ่มต้นส่งข้อมูลจะเริ่มจากการให้ขา data มีลอจิก "0" ด้วยช่วงเวลา 1 บิต เรียกบิตนี้ว่า บิตเริ่มต้น จากนั้นบิตข้อมูลจะถูกส่งออกไปโดยเริ่มจากบิตที่มีนัยสำคัญต่ำสุด หรือ บิต LSB ก่อนจากนั้นตามด้วยบิตพาริตี้ บิตสุดท้ายที่จะส่งคือ บิตปิดท้ายหรือบิตหยุดโดยจะทำให้ขา data มีสถานะลอจิก "1" อีกครั้งด้วยระยะเวลา 1 บิต ,1.5 บิต หรือ 2 บิต เพื่อเป็นการแสดงว่าสิ้นสุดข้อมูลแล้ว

อัตราเร็วในการรับและส่งข้อมูลแบบอะซิงโครนัสหรืออัตราบอด ที่ใช้สำหรับพอร์ต RS-232 มีด้วยกันหลายค่า ได้แก่ 110,150,300,600,1,200,2,400,4800,9,600และ 19,200 บิตต่อวินาที



รูปที่ 2.1 รูปแบบของข้อมูลอนุกรมแบบอะซิงโครนัส

2.2 มาตรฐานพอร์ตอนุกรมแบบ RS-232

มาตรฐาน RS-232 ถูกใช้ในการกำหนดรูปแบบการสื่อสารข้อมูลกันระหว่างอุปกรณ์เชื่อมต่อข้อมูล (Data Terminal Equipment : DTE) กับวงจรข้อมูลปลายทาง (Data Circuit Terminating :DCE) อุปกรณ์ DTE จะต้องเป็นอุปกรณ์ประมวลผลในตัว เช่น ไมโครคอนโทรลเลอร์หรือไมโครคอมพิวเตอร์ ซึ่งมีความสามารถในการสร้างบิตข้อมูลอนุกรมได้ ส่วนอุปกรณ์ DCE ทำหน้าที่เป็นเพียงตัวรับข้อมูลที่ส่งมาจาก DTE เท่านั้น

สำหรับการใช้ในคอมพิวเตอร์ พอร์ตอนุกรม RS-232 ถูกใช้เพื่อเชื่อมต่อกับโมเด็ม,เมาส์ และ เครื่องพิมพ์ที่สามารถติดต่อทางพอร์ตอนุกรมได้

มาตรฐานการเชื่อมต่อแบบ RS-232 จะใช้คอนเน็กเตอร์แบบ DB-25 ตัวผู้หรือ DB-9 ตัวผู้ซึ่งคอนเน็กเตอร์แบบ DB-25 จะมีขาต่อใช้งานเพียง 9 เส้นเช่นเดียวกับคอนเน็กเตอร์ แบบ DB-9 เนื่องจากขาอื่นๆที่เคยมีการใช้งานมาในอดีตไม่ค่อยมีความสำคัญมากนักจึงถูกยกเลิกไป

●ขา Data Carrier Detect : DCD หรืออาจเรียกว่า Carrier Detect : CD ขานี้จะแอกตีฟเมื่อมีการส่งสัญญาณพาหะจากอุปกรณ์สื่อสารข้อมูล เช่น โมเด็ม สำหรับการใช้งานปกติขานี้จะไม่ถูกใช้งานมากนัก

●ขา Receive Data : RD หรือ R×D ขานี้ใช้เพื่อรับสัญญาณอนุกรมเข้ามายังคอมพิวเตอร์ โดยนำข้อมูลที่อ่านได้ไปเก็บไว้ในรีจิสเตอร์บัฟเฟอร์

●ขา Transmitter Data : TD หรือ T×D ขานี้ใช้เพื่อส่งข้อมูลอนุกรมออกมาจากคอมพิวเตอร์ โดยการนำข้อมูลที่เก็บอยู่ในบัฟเฟอร์ส่งข้อมูลออกไป

●ขา Data Terminal Ready : DTR เป็นขาเอาท์พุทที่ใช้ในการส่งสัญญาณข้อมูลออกจากคอมพิวเตอร์ เพื่อให้อุปกรณ์ปลายทางรับรู้ว่าต้องการติดต่อกับอุปกรณ์ปลายทาง โดยขา DTR นี้จะต้องเชื่อมต่อกับขา DSR ของอุปกรณ์ปลายทาง และขา DTR ของอุปกรณ์ปลายทางจะต้องเชื่อมต่อกับขา DSR ของคอมพิวเตอร์ และถ้าใช้การเชื่อมต่อแบบ 3 สายต้องเชื่อมต่อขา DTR และ DSR ของพอร์ตอนุกรมเข้าด้วยกัน และจะต้องเชื่อมเข้ากับขา DCD ด้วยในกรณีที่โปรแกรมสื่อสารที่ใช้ในการตรวจสอบสัญญาณพาหะ

●ขา Signal Ground : GND เป็นขากราวด์ของสัญญาณ

●ขา Data Set Ready : DSR ขานี้จะใช้ควบคู่กับขา DTR เพื่อตรวจสอบการเชื่อมต่อระหว่างคอมพิวเตอร์กับอุปกรณ์ปลายทาง ซึ่งขา DSR นี้จะเป็นขาสำหรับรับรู้ข้อมูลจากภายนอก

●ขา Request To Send : RTS เป็นขาเอาท์พุทสำหรับส่งสัญญาณร้องขอให้อุปกรณ์ปลายทางส่งข้อมูลมาทางคอมพิวเตอร์โดยขาที่รับสัญญาณ RTS ก็คือขา CTS ซึ่งในกรณีที่มีการเชื่อมต่อแบบ 3 สาย จะต้องเชื่อมต่อขา RTS และ CTS เข้าด้วยกัน เพื่อให้การรับและส่งข้อมูลเกิดขึ้นได้ตลอดเวลา

●ขา Clear To Send : CTS เป็นขาอินท์พุททำหน้าที่รอรับสัญญาณที่ส่งเข้ามา เมื่อมีการส่งสัญญาณเข้ามาที่ขา T×D จะถูกส่งออกไป ขานี้ใช้เพื่อตรวจสอบอุปกรณ์ต่อพ่วงว่าพร้อมที่จะรับข้อมูลแล้วหรือยัง

●ขา Ring Indicator : RI ใช้แสดงสถานะสัญญาณเรียกจากสายโทรศัพท์ ปกติในการสื่อสารโดยทั่วไปสายนี้จะไม่ถูกใช้งาน จะใช้งานก็ต่อเมื่อมีการเชื่อมต่อกับโมเด็มแล้ว ยังมีความต้องการการตรวจสอบสัญญาณเรียกจากสายโทรศัพท์

2.3 มาตรฐานการเชื่อมต่อแบบ I^2C

I^2C (Inter-IC Communication) คือ การติดต่อสื่อสารระหว่างไอซี โดยบัส I^2C โดยมีจุดมุ่งหมายให้ไอซีหรือโมดูลสามารถติดต่อสั่งงาน และควบคุมภายใต้สายสัญญาณ 2 เส้นคือ

1. สายข้อมูลอนุกรมหรือ SDA (Serial Data Line)
2. สายสัญญาณนาฬิกาอนุกรมหรือ SCL(Serial Clock Line)

คุณสมบัติของ I^2C

-สาย SDA และสาย SCL เป็นสายสัญญาณสองทิศทาง

-ต่อตัวตานทานพูลอัป (R_p) กับแรงดัน 5 Volt ไว้ตลอดเวลา เพื่อให้สายมีสถานะลอจิกสูงในขณะที่ไม่มีการติดต่อใช้งาน และจะช่วยป้องกันสัญญาณรบกวน

-วงจรเอาท์พุทของอุปกรณ์ที่ต่ออยู่บนบัส I^2C ต้องมีลักษณะเป็นวงจรเดรนเปิด (Open-Drain) คอลเล็กเตอร์เปิด (Open-Collector)

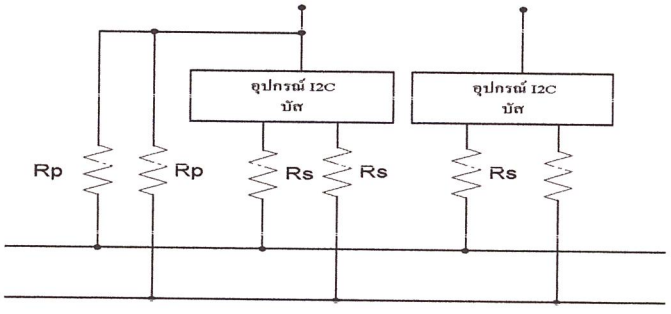
-อัตราการถ่ายทอดข้อมูล 100 กิโลบิตต่อวินาทีในโหมดปกติ (Standard Mode) และสูงถึง400 กิโลบิต ต่อวินาทีในโหมดความเร็วสูง (Fast Mode)

-อุปกรณ์ที่ต่อร่วมอยู่บนบัส I^2C ค่าความจุไฟฟ้ารวมที่เกิดขึ้นระหว่างสาย SDA กับ SCL จะต้องไม่เกิน 400 pF

-การเข้าถึงอุปกรณ์บนบัส I^2C ใช้ข้อมูลสำหรับการเข้าถึง 2 ค่าคือ 7 บิต (7-Bit Addressing) หรือ 10 บิต(10 –Bit Adressing)

-สามารถเชื่อมต่ออุปกรณ์ที่ใช้ไฟเลี้ยงไม่เท่ากันตลอดการสื่อสารได้ โดยการต่อสาย SDA และ SCL ของอุปกรณ์ต่อพ่วงเข้าด้วยกัน และต้องต่อตัวต้านทานพูลอัป (R_p) เข้ากับแรงดัน +5 Volt ไว้ด้วยเสมอ

อุปกรณ์และตัวต่อตัวต้านทานอนุกรม (R_s) เข้ากับขา SDA และ SCL ก่อนเข้าสู่บัส I^2C เพื่อป้องกันไฟกระชากขนาดใหญ่



รูปที่ 2.2 การต่อตัวต้านทาน Rs เพื่อลดสัญญาณรบกวนขนาดใหญ่ที่อาจเข้ามาในบัส I^2C

หลักการของบัส I^2C

ก่อนอื่นต้องรู้โปรโตคอล (Protocal) ข้อตกลงพื้นฐาน

-ตัวส่ง (Transmitter) คืออุปกรณ์ที่เป็นผู้สร้างหรือส่งข้อมูล

-ตัวรับ (Receiver) คืออุปกรณ์ที่เป็นผู้รับข้อมูล

-อุปกรณ์บนบัส I^2C สามารถเป็นได้ทั้งตัวรับและตัวส่ง

-มาสเตอร์ (Master) คือ อุปกรณ์ที่ทำหน้าที่ควบคุมจังหวะการทำงานหรือการติดต่อบนบัส I^2C

-สเลฟ (Slave) คืออุปกรณ์ที่ถูกควบคุมหรืออุปกรณ์ที่ต่อพ่วงเข้าไปบนบัส I^2C

ข้อกำหนดที่สำคัญของการติดต่อบนบัส I^2C คือ

1. การถ่ายทอดข้อมูลจะเกิดขึ้นได้เมื่อบัสว่างเท่านั้น
2. ในระหว่างการถ่ายทอดข้อมูล เมื่อใดก็ตามที่สาย SCL มีสถานะเป็นลอจิกสูง สายข้อมูลต้องรักษาข้อมูลไว้ อย่าให้เกิดการเปลี่ยนแปลงขึ้นเด็ดขาด มิฉะนั้นสัญญาณที่เกิดขึ้นจะได้รับการแปลความหมายให้เป็นสัญญาณควบคุมแทน

สภาวะที่เกิดขึ้นบนบัส I^2C มี 5 สภาวะ

1. บัสว่าง คือสาย SDA และ SCL มีสถานะลอจิกสูง

2. สภาวะเริ่มต้น คือสาย SDA เปลี่ยนจากสถานะลอจิกสูงไปต่ำ ขณะที่สาย SCL มีสถานะลอจิกสูง

3.สภาวะหยุด คือสาย SDA เปลี่ยนจากลอจิกต่ำไปสูง ขณะที่สาย SCL มีสถานะลอจิกสูง

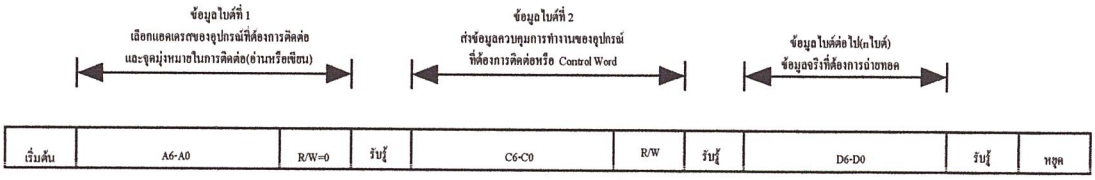
4. ข้อมูลดำรงอยู่บนบัส เกิดถัดจากสภาวะเริ่มต้น โดยสาย SDA เป็นสายข้อมูลที่จะรับข้อมูลเมื่อขา SCL มีสถานะลอจิกสูง ในขณะที่สาย SCL มีสถานะลอจิกสูง สถานะที่สาย SDA จะต้องคงที่

5. รับรู้ข้อมูล เกิดขึ้นหลังจากที่การถ่ายทอดข้อมูลจากตัวส่งมายังตัวรับเกิดขึ้นอย่างสมบูรณ์ โดยตัวส่งจะส่งบิตรับรู้ (acknowledge bit)ที่มีสถานะลอจิกสูง อุปกรณ์มาสเตอร์ก็จะส่งสัญญาณรับรู้พิเศษ ซึ่งสัมพันธ์กับสัญญาณนาฬิกา เพื่อตอบสนองบิตรับรู้จากตัวส่ง ทางด้านตัวรับจะส่งบิตรับรู้ที่มีสถานะลอจิกต่ำลงบนบัสอุปกรณ์สเลฟที่ถูกอ้างถึงในการติดต่อหรือกำลังติดต่ออยู่ในขณะนั้นก็จะกำหนดบิตรับรู้เพื่อตอบสนองให้ทราบว่าได้รับข้อมูลในแต่ละไบต์เรียบร้อยแล้ว การทำงานบนบัส I^2C สิ่งสำคัญอันดับแรกของการทำงานบนบัส I^2C คือการอ้างถึงอุปกรณ์แต่ละตัวมี 2 รูปแบบ คือ

1. การอ้างถึงแบบ 7 บิต

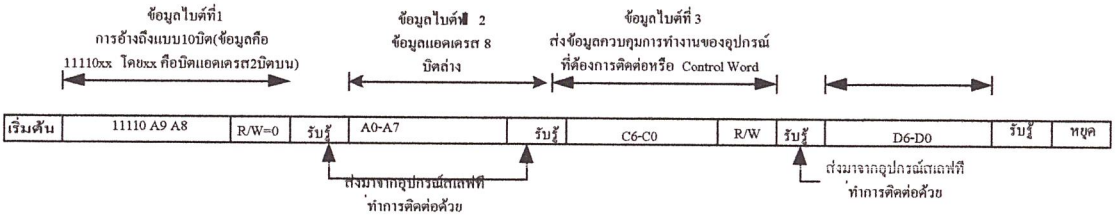
บิต7	บิต6	บิต5	บิต4	บิต3	บิต2	บิต1	บิต0
X	X	X	X	A2	A1	A0	R/W

รูปที่ 2.3 รูปแบบของข้อมูลกำหนดแอดเดรสที่ใช้ในการอ้างถึงแบบ 7 บิต



รูปที่ 2.4 รูปแบบของข้อมูลอนุกรมที่ใช้ติดต่อกับอุปกรณ์บัส I^2C เมื่อใช้การอ้างถึงแบบ 7 บิต

2. การอ้างถึงแบบ 10บิต



รูปที่ 2.5 รูปแบบของข้อมูลอนุกรมที่ใช้ติดต่อกับอุปกรณ์บัส I^2C เมื่อใช้การอ้างถึงแบบ 10 บิต

รูปแบบคล้าย 7 บิต โดย 5 บิตแรกมีข้อมุลเป็น 11110 2 บิตถัดมาเป็นแอดเดรสของอุปกรณ์ที่ต้องการติดต่อ LSB กำหนดว่าจะอ่านหรือเขียนข้อมูล

การต่ออุปกรณ์ระบบ I^2C กับพอร์ตอนุกรม

ใช้ขา CTS สำหรับข้อมูลที่ป้อนเข้ามายังคอมพิวเตอร์ ผ่านทาง SDA

ใช้ขา RTS สำหรับส่งข้อมูลจากคอมพิวเตอร์ไปยังอุปกรณ์บัส I^2C ผ่านทางขา SDA

ใช้ขา DTR สำหรับส่งสัญญาณนาฬิกาออกจากคอมพิวเตอร์ เข้าไปในระบบบัส I^2C ทางขา SCL

การทำงานในส่วนนี้จะใช้สัญญาณของพอร์ตอนุกรมที่ผ่านวงจรบัฟเฟอร์มาแล้วต่อร่วมกับทรานซิสเตอร์ เพื่อกำหนดสถานะบนบัสในสภาวะปกติ มีสถานะเป็นลอจิกสูงหรือเป็นสถานะบัสว่างนั่นเอง

2.4 การเชื่อมต่อกับสัญญาณอะนาลอกผ่านพอร์ตอนุกรม

ปกติแล้วข้อมูลในการติดต่อกับพอร์ตอนุกรมของคอมพิวเตอร์นั้นจะเป็นสัญญาณดิจิตอลทั้งสิ้น แต่เมื่อนำมาเชื่อมต่อกับอุปกรณ์ภายนอกแล้ว ย่อมต้องเชื่อมต่อและประมวลผลสัญญาณอะนาลอกด้วย ในการเชื่อมต่อกับสัญญาณอะนาลอกต้องใช้ไอซีพิเศษที่ทำหน้าที่ในการแปลงสัญญาณอะนาลอกเป็นดิจิตอลหรือที่เรียกว่าไอซี ADC ในการทดลองต่อไปจะเป็นการใช้ไอซี ADC ร่วมกับบอร์ด ST-29 เพื่อเชื่อมต่อสัญญาณอะนาลอกเข้ากับพอร์ตอนุกรม โดยไอซี ADC ที่ใช้เป็นเบอร์ PCF8591ซึ่งมีรูปแบบการเชื่อมต่อแบบบัส I^2C

ข้อมูลเบื้องต้นของ PCF8591

เป็นไอซีแปลงสัญญาณอะนาลอกเป็นดิจิตอลและแปลงสัญญาณดิจิตอลเป็นสัญญาณอะนาลอกในตัวเดียวกันทำให้สามารถนำไปประยุกต์ใช้งานได้อย่างกว้างขวาง โดยมีรายละเอียดดังนี้

- ทำงานโดยใช้แหล่งจ่ายไฟชุดเดียว
- ทำงานที่แรงดัน 2.5 ถึง 6 โวลต์
- กินกระแสขณะอยู่ในภาวะสแตนบายค์ต่ำ
- ติดต่อกับไมโครคอมพิวเตอร์หรือไมโครคอนโทรลเลอร์ผ่านระบบบัส I^2C
- สามารถเลือกตำแหน่งแอดเดรสทางฮาร์ดแวร์จากขา ทำให้สามารถต่อพ่วงกันได้สูงสุดถึง 8 ตัว
- อัตราการสุ่มข้อมูล(Samping) ขึ้นอยู่กับความเร็วของสัญญาณอะนาลอกได้ 4 ช่อง ทั้งยังเลือกได้ว่าให้ทำงานแบบแยกช่องหรือทำงานแบบวงจรดิฟเฟอเรนเชียล
- การอ่านค่าสามารถกำหนดให้เลื่อนช่องอินพุทโดยอัตโนมัติได้
- สัญญาณอะนาลอกมีระดับแรงดันตั้งแต่ V_{SS} ไปจนถึง V_{DD}
- วงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอลเป็นแบบซักเซสซีฟ แอปพร็อกซิเมชั่นขนาด 8 บิต
- มีวงจรแปลงสัญญาณดิจิตอลเป็นอะนาลอกขนาด 8 บิต 1 ช่อง

PCF8591 สามารถทำหน้าที่เป็นไอซีแปลงสัญญาณอะนาลอกเป็นดิจิตอลขนาด 8 บิต 4 ช่อง และสามารถทำหน้าที่เป็นไอซีแปลงสัญญาณดิจิตอลเป็นสัญญาณอะนาลอกได้ในคราวเดียวกันด้วยการควบคุมผ่านระบบบัส I^2C ทำให้สามารถต่อพ่วงไอซี PCF8591 ได้สูงสุดถึง 8 ตัว รองรับการอ่านค่าสัญญาณอะนาลอกอินพุทได้สูงสุดถึง 32 ช่อง และสามารถส่งสัญญาณอะนาลอกเอาท์-พุทสูงสุดได้ถึง 8 ช่องด้วยการกำหนดแอดเดรสจากขา A0,A1 และ A2 ข้อมูลควบคุมทั้งหมดจะถูกเก็บไว้ในรีจิสเตอร์ควบคุมภายใน PCF8591 เมื่อจ่ายไฟให้แก่ PCF8591 ครั้งแรก บิตต่างๆของข้อมูลภายในรีจิสเตอร์ควบคุมจะเป็น “0”

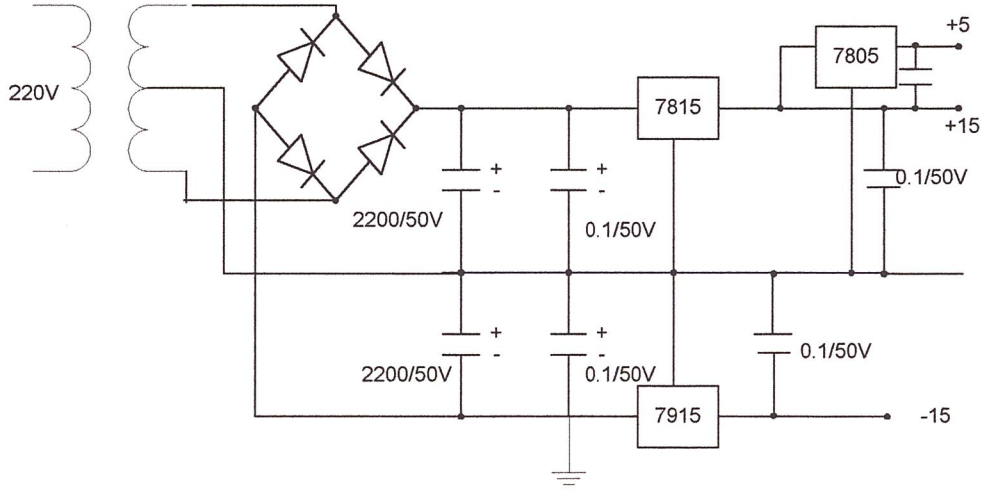
ออสซิเลเตอร์

วงจรออสซิเลเตอร์ภายใน PCF8591 จะสร้างสัญญาณนาฬิกาสำหรับแปลงสัญญาณอะนาลอกเป็นดิจิตอล เมื่อต้องการใช้วงจรออสซิเลเตอร์ภายใน ขา EXT ต้องต่อลงกราวด์ ถ้าต้องการใช้ออสซิเลเตอร์จากภายนอกขา EXT ต้องต่อเข้ากับไฟบวก และป้อนสัญญาณนาฬิกาเข้าที่ขา OSC ของ PCF8591 โดยความถี่ของสัญญาณนาฬิกาสูงสุดที่ป้อนให้กับวงจรออสซิเลเตอร์เท่ากับ 1.25 MHz

2.5 วงจรจ่ายไฟกระแสตรง, วงจรเปลี่ยนสัญญาณแรงดันไฟฟ้าเป็นสัญญาณกระแสไฟฟ้าและวงจรเปลี่ยนแรงดันไฟฟ้าจาก 1 ถึง 5 เป็น 0 ถึง 5 โวลท์

วงจรจ่ายไฟกระแสตรง

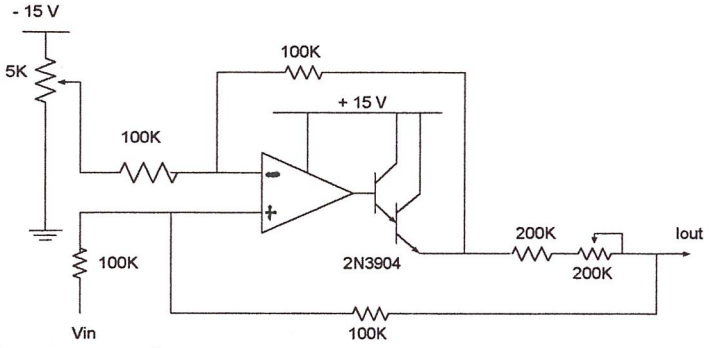
ในการออกแบบแหล่งจ่ายไฟกระแสตรงชนิดต่างๆ ตั้งแต่แบบง่ายๆ ไปจน ถึงแบบที่มีวงจรรักษาระดับแรงดันให้คงที่(VOTAGE REGURATOR)ทั้งแบบใช้ทรานซิสเตอร์จนกระทั่งถึงไอซี จุดมุ่งหมายของการรักษาระดับแรงดันให้คงที่ก็คือ การรักษาระดับแรงดันเอาท์พุทให้คงที่ถึงแม้ว่ากระแสของโหลดเอาท์พุท ระดับแรงดันอินพุทหรือทั้งสองอย่างมีการเปลี่ยนแปลงไปก็ตาม จากรูป 2.6 เป็นวงจรรักษาระดับแรงดันที่ใช้ไอซีแบบ 3 ขา ประกอบด้วยขาอินพุท ขาเอาท์พุท และขากราวด์



รูปที่ 2.6 วงจรจ่ายไฟกระแสตรง

วงจรเปลี่ยนสัญญาณแรงดันไฟฟ้าเป็นสัญญาณกระแสไฟฟ้า

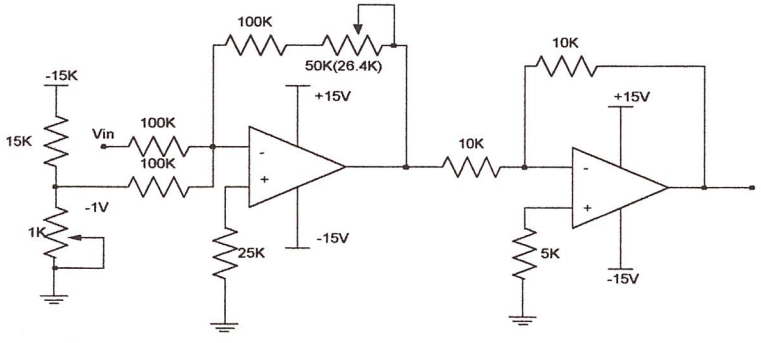
จากรูปที่ 2.6 นำแรงดันเอาท์พุทของวงจรแหล่งจ่ายไฟกระแสตรงที่มีการรักษาระดับแรงดันให้คงที่ มาใช้งานในวงจรส่วนแรกซึ่งประกอบด้วย ไอซี 741(เป็นออปแอมป์),ความต้านทานค่าต่างๆ ทั้งแบบธรรมดาและปรับค่าได้, ตัวเก็บประจุไฟฟ้า(CAPACITOR)ทั้งแบบไม่มีขั้วและแบบมีขั้ว โดยในวงจรนี้จะรับอินพุทมาจากไอซีพีเอฟ 8591(PCF 8591)เป็นสัญญาณแรงดันไฟตรง 0-5 โวลต์สามารถทำการปรับแต่งได้โดยการปรับ ซีโร่ ที่ 5K(ZERO) และสแปนที่ 50 K(ZERO-SPAN) โดยใช้แหล่งจ่ายสัญญาณแรงดันที่สามารถปรับค่าได้ อย่างต่อเนื่อง



รูปที่ 2.7 วงจรเปลี่ยนสัญญาณแรงดันไฟฟ้าเป็นสัญญาณกระแสไฟฟ้า

วงจรเปลี่ยนแรงดันไฟฟ้าจาก 1 ถึง 5 เป็น 0 ถึง 5 โวลท์

จากรูป 2.8 วงจรนี้จะรับอินพุทจากอุปกรณ์วัดของกระบวนการ ซึ่งจะมีค่าอยู่ระหว่าง 1 ถึง 5 โวลท์ และทำการเปลี่ยนแรงดันไฟฟ้าเป็น 0 ถึง 5 โวลท์ โดยผ่านไอซีแอลเอ็ม 324 (LM 324) ค่าความต้านทานต่างๆ ทั้งแบบปรับค่าได้และปรับค่าไม่ได้ สามารถทำการปรับ ZERO ได้โดยการปรับค่าความต้านทานแบบปรับค่าได้ 1K และสามารถปรับ SPAN ได้โดยการปรับค่าความต้านทานแบบปรับค่าได้ 50K และนำสัญญาณเอาท์พุทไปเข้าไอซี PCF 8591 เพื่อกำหนดมาตรฐานการส่งข้อมูลเป็น ไอสแควร์ซี(I^2C)



รูปที่ 2.8 วงจรเปลี่ยนแรงดันไฟฟ้าจาก 1 ถึง 5 เป็น 0 ถึง 5 โวลท์

2.6 การเขียนโปรแกรมด้วย Visual Basic เพื่อติดต่อพอร์ตอนุกรมและคัสตอมคอนโทรล

จุดมุ่งหมาย

1. สามารถติดต่อกับพอร์ตอนุกรมผ่านการเขียนโปรแกรมด้วย Visual Basic ได้
2. สามารถนำคัสตอมคอนโทรล MSComm ไปใช้ประโยชน์ในงานควบคุม

คอนโทรล MSComm

สำหรับการใช้งาน Visual Basic ตั้งแต่เวอร์ชัน 2 เป็นต้นมา ใน Visual Basic จะมีคัสตอมคอนโทรลสำหรับการสื่อสารอนุกรมผ่านทางพอร์ตอนุกรมของคอมพิวเตอร์มาให้ โดยใน Visual Basic เวอร์ชัน 2 และเวอร์ชัน 3 จะใช้ชื่อว่า MSCOMM.VBX ส่วนเวอร์ชัน 4 ใช้ชื่อว่า MSCOMM.OCX สำหรับการทำงานกับระบบปฏิบัติการ 32 บิต สำหรับในVisual Basic เวอร์ชัน 5 จะมีเพียง MSCOMM32.OCX เท่านั้นเพราะถูกออกแบบมาให้ใช้งานกับระบบปฏิบัติการ 32 บิต

MSComm จัดเตรียมทางเลือกเอาไว้ 2 ทางเพื่อความสะดวกในการสื่อสารข้อมูล ทางแรกคือการสื่อสารข้อมูลที่กระตุ้นด้วยเหตุการณ์(event-driven communications)เป็นรูปแบบการใช้งานที่มีประสิทธิภาพมากสำหรับการตอบสนองแบบทันทีทันใด เช่น เมื่ออักษรถูกส่งมาที่พอร์ตอนุกรม หรือเกิดการเปลี่ยนแปลงที่ขา Data Carrier(DCD) หรือขา Request To Send (RTS) เหตุการณ์ ONCOMM ของ MSComm จะสามารถตรวจจับสัญญาณนั้นได้ทันทีทันใด ซึ่งจะกล่าวถึงรายละเอียดในหัวข้อคุณสมบัติ CommEvent ต่อไป ส่วนทางเลือกที่ 2 เป็นการคอยตรวจสอบค่าเหตุการณ์และความผิดพลาดที่เกิดขึ้นด้วยการดูค่าที่เปลี่ยนแปลงภายในคุณสมบัติ CommEvent หลังจากให้โปรแกรมทำงานในฟังก์ชันต่างๆไปเรียบร้อยแล้ว ซึ่งวิธีนี้ใช้งานได้ดีในกรณีที่โปรแกรมมีขนาดเล็ก คอนโทรล MSComm 1 สามารถควบคุมการทำงานของพอร์ตอนุกรมได้ 1 พอร์ตถ้าในโปรแกรมที่ใช้งานต้องการติดต่อกับพอร์ตอนุกรมมากกว่า 1 พอร์ต จะต้องใช้ MSComm มากกว่า 1 ตัว เพื่อควบคุมพอร์ตอนุกรมในแต่ละพอร์ต แอดเดรสของพอร์ตอนุกรมและแอดเดรสของการเกิดอินเตอร์รัปต์สามารถเปลี่ยนแปลงได้จากการแก้ไขที่ Control Panel

ถึงแม้ว่า คอนโทรล MSComm จะมีคุณสมบัติ (Property) มากมายหลายตัว แต่สามารถทำความเข้าใจได้ไม่ยากดังนี้

●CommPort

ใช้ในการกำหนดและอ่านค่าพอร์ตอนุกรมที่ต่ออยู่(COM1,COM2,COM3,COM4)

รูปแบบการใช้งาน

object.CommPort[=value]

โดย Value เป็นค่าของพอร์ตอนุกรม ชนิดของข้อมูลเป็น Integer ค่า Value สามารถกำหนดได้ใน

ช่วง1-16(ค่าเริ่มต้นกำหนดไว้ที่1)เมื่อมีการกำหนดค่าแล้วทำการเปิดพอร์ตโดยใช้คุณสมบัติ PortOpen แต่ว่าพอร์ตนั้นไม่มีอยู่ในระบบ MSCOMM จะสร้างสัญญาณแสดงข้อผิดพลาด error 68 ขึ้นมาซึ่งหมายถึงอุปกรณ์ตัวนี้ไม่มีอยู่ในระบบ ดังนั้นการเขียนโปรแกรมจึงจำเป็นต้องกำหนดตำแหน่งของพอร์ตอนุกรมก่อนที่ใช้คุณสมบัติ PortOpen

●Setting

ใช้ในการกำหนดและอ่านค่าอัตราบอด,พาริตี้,จำนวนของบิตข้อมูล,จำนวนของบิตปิดท้าย

-รูปแบบการใช้งาน

Object.Settings[=value]

ค่า Value มีชนิดข้อมูลเป็นแบบ “BBBB,P,D,S” โดย BBBBเป็นค่าอัตราบอด, Pเป็นค่าพาริตี้, D เป็นจำนวนของบิตข้อมูลและ S เป็นจำนวนของบิตปิดท้ายปกติแล้วค่านี้ถูกกำหนดไว้เป็น “9600,N,8,1”

●PortOpen

ใช้ในการกำหนดและอ่านค่าสถานะของพอร์ตอนุกรม เพื่อเปิดและปิดพอร์ตอนุกรม

-รูปแบบการใช้งาน

Object.PortOpen[=value]

ค่า Value มีชนิดข้อมูลเป็นแบบบูลีนคือ True กับ False โดย True หมายถึงการเปิดพอร์ตอนุกรม และ False หมายถึงการปิดพอร์ตอนุกรม สำหรับการปิดพอร์ตนั้นจะมีการเคลียร์บัฟเฟอร์รับบข้อมูลและบัฟเฟอร์ส่งข้อมูล คอนโทรล MSComm จะปิดพอร์ตอนุกรมโดยอัตโนมัติเมื่อออกจากโปรแกรม ก่อนที่จะใช้คุณสมบัติ PortOpen ต้องตรวจสอบให้แน่ใจก่อนว่าคุณสมบัติ CommPort นั้นได้ทำการกำหนดตำแหน่งของพอร์ตอนุกรมไว้ถูกต้องหรือไม่ไม่เช่นนั้น MSComm จะแสดงข้อผิดพลาด Error 68 แจ้งแก่ผู้ใช้งาน คุณสมบัติ DTREnable หรือ RTSEnable ถูกกำหนดให้เป็น True ก่อนที่จะทำการเปิดพอร์ต ค่าคุณสมบัติของ DTREnable หรือ RTSEnable จะถูกเซตเป็น False หลังจากปิดพอร์ต แต่ถ้าเซตเป็น False หลังจากปิดโปรแกรมแล้วค่าที่กำหนดไว้จะเป็นค่าเดิม

●Input

อ่านค่าและลบค่าขบวนข้อมูลจากบัฟเฟอร์ภาครับ

-รูปแบบการใช้งาน

Object.Input

คุณสมบัติ InputLen เป็นตัวกำหนดจำนวนของตัวอักษรที่จะอ่านโดยคุณสมบัติ Input การกำหนดค่าให้ InputLen เท่ากับ 0 เป็นการกำหนดให้คุณสมบัติ Input ทำการอ่านข้อมูลในบัฟเฟอร์รับข้อมูลทั้งหมด

คุณสมบัติ InputMode เป็นตัวกำหนดชนิดของข้อมูลที่คุณสมบัติ Inputรับเข้ามา InputMode ถูกกำหนดเป็น

ComInputModeText คุณสมบัติ Input จะส่งค่ากลับมาในรูปแบบข้อความชนิดข้อมูลเป็นแบบ Varaint ถ้า InputMode กำหนดเป็น comInputModeBinary คุณสมบัติ Input จะส่งข้อมูลกลับมาในรูปของไบนารี่และชนิดข้อมูลเป็นแบบ Varaint ในตัวอย่างโปรแกรมที่แสดงให้เห็นถึงวิธีในการรับข้อมูลจากบัฟเฟอร์รับข้อมูล

```
Private Sub Command_Click()
Dim InString as String
'Retrieve all available data.
MSComm1.InputLen = 0
'Check for data.
If MSComm1.InBufferCount Then
'Read data.
InString = MSComm1.Input
End If
End Sub
```

●OutPut

ใช้ในการส่งขบวนของข้อมูลไปยังบัฟเฟอร์ส่งข้อมูล

-รูปแบบการใช้งาน

Object.Output[=value]

ค่า Value เป็นค่าของตัวอักษรที่เขียนไปยังบัฟเฟอร์ส่งข้อมูล คุณสมบัติ Output สามารถใช้ในการเขียนส่งข้อมูลตัวอักษรหรือข้อมูลไบนารี่ก็ได้ โดยการส่งข้อมูลเป็นรูปแบบตัวอักษรจะต้องกำหนดชนิดของข้อมูลเป็นแบบVaraint และมีข้อมูลภายในเป็นแบบ String สำหรับการส่งข้อมูลไบนารี่จะต้องกำหนดชนิดของข้อมูลเป็นแบบVaraint และมีข้อมูลภายในเป็นแบบ Byte ตัวอย่างโปรแกรมที่ เป็นการส่งค่าที่ป้อนจากคีย์บอร์ดทุกตัวไปยังพอร์ตอนุกรม

```
Private Sub Form_KeyPress(KeyAscii As Integer)
Dim Buffer as Varaint
'Set and open port
MSComm1.CommPort = 1
MSComm1.PortOpen = True
```

```
Buffer = Chr$(KeyAscii)
MSComm1.Output = Buffer
End  Sub
```

●DTREnable

ใช้ในการอินาเบิลขา Data Terminal Ready(DTR) โดยสัญญาณของขา DTR จะส่งจากคอมพิวเตอร์ไปยังโมเด็มเพื่อแสดงว่าคอมพิวเตอร์พร้อมที่จะรับข้อมูลแล้ว ชนิดของข้อมูลเป็นแบบ Bollean

-รูปแบบการใช้งาน

Object.DTREnable[=value]

ค่า Value เป็นสถานะ True หรือ False เพื่ออินาเบิลขา DTR โดย True หมายถึง อินาเบิลขาDTR False หมายถึง ดิสเอเบิลขา DTR(เป็นค่าปกติ) เมื่อขา DTR ถูกกำหนดสถานะให้เป็น True ที่ขา DTR จะมีสถานะลอจิก “1” เมื่อทำการเปิดพอร์ต และจะมีสถานะลอจิก “0” เมื่อทำการปิดพอร์ต เมื่อขา DTR ถูกกำหนดสถานะเป็น False ที่ขา DTRจะมีสถานะลอจิก “0” ตลอดเวลาไม่ว่าจะใช้คำสั่งเปิดพอร์ตหรือปิดพอร์ต

สำหรับการใช้งานกับโมเด็ม การทำให้ขา DTR เป็นลอจิก “0” เป็นการยกเลิกการติดต่อ(วางหู)

●RTSEnable

ใช้เพื่ออินาเบิลขา Requst To Send(RTS) โดยขา RTS จะเป็นสัญญาณที่ส่งจากคอมพิวเตอร์ไปยังโมเด็มเพื่อร้องขอส่งข้อมูล ชนิดข้อมูลเป็นแบบ Boolean

-รูปแบบการใช้งาน

Object.RTSEnable[=value]

ค่า Value เป็นค่าสถานะ True หรือ False เพื่ออินาเบิลหรือดิสเอเบิลขา RTS โดย

True หมายถึง อินาเบิลขา RTS

False หมายถึง ดิสเอเบิลขา RTS(เป็นค่าปกติ)

เมื่อขา RTSEnable ถูกกำหนดให้เป็น True ขา RTS จะมีสถานะลอจิก “1”เมื่อเปิดพอร์ตและมีสถานะลอจิก“0”เมื่อปิดพอร์ต

●EOFEnable

เป็นการกำหนดให้ MSComm รอรับสัญลักษณ์แสดงส่วนท้ายสุดของไฟล์ (End of file:EOF) ระหว่างการรับอินพุทเข้ามา ถ้าพบสัญลักษณ์ EOF ภาคอินพุทจะหยุดรับข้อมูล และเหตุการณ์ OnComm จะถูกกระตุ้นให้ทำงาน คุณสมบัติ CommEvent จะมีค่าเท่ากับ 7 หรือ ComEvEOF

-รูปแบบการใช้งาน

Object.EOFEnable[=value]

โดย Value เป็นสถานะ True หรือ False เพื่ออินาเบิลหรือดิสเอเบิลการทำงานของเหตุการณ์ OnComm เมื่อตรวจพบสัญลักษณ์ EOF โดย

True หมายถึง เหตุการณ์ OnComm จะถูกกระตุ้นให้ทำงานด้วย EOF

False หมายถึง เหตุการณ์ OnComm จะไม่ถูกกระตุ้นให้ทำงานด้วย EOF(เป็นค่าปกติ)

เมื่อ EOFEnable กำหนดให้เป็น False ส่วนควบคุมจะไม่มีการตรวจสอบสัญลักษณ์ EOF

●CTSHolding

ผู้ใช้งานสามารถตรวจสอบการทำงานของขา Clear To Send(CTS) ได้ว่ามีสถานะลอจิก"0"หรือ "1"โดยค่าที่อ่านได้จะเป็นบูลีน True และ False ถ้าค่า CTSHolding เป็น True ขา CTS จะมีสถานะลอจิกเป็น "1"ถ้าค่า CTSHolding เป็น Falseขา CTS จะมีสถานะลอจิกเป็น "0"

-รูปแบบการใช้งาน

object.CTSHolding

เมื่อขา CTS เป็นลอจิก "0" (CTSHolding=False)และเกิดไทม์เอาต์ คอนโทรลMSComm จะกำหนดคุณสมบัติ CommEvent มีค่าเป็น comEventCTSTO(Clear To Send Timeout) และกระตุ้นให้เกิดเหตุการณ์ OnComm

●CDHolding

ผู้ใช้งานสามารถตรวจสอบการทำงานของขา Data Carrter Detect(DCD) ได้ว่ามีสถานะลอจิกเป็น "1" หรือ "0" โดยค่าที่อ่านได้จะเป็นบูลีน True และ False ถ้าค่า CDHolding เป็น True ขา DCD จะมีสถานะลอจิก "1" ถ้าค่า CDHolding เป็น False ขา DCD จะมีสถานะลอจิก "0"

-รูปแบบการใช้งาน

object.CDHolding

เมื่อขาDCD มีลอจิก "1"(CDHolding = True) และเกิดไทม์เอาต์คอนโทรล MSCommจะกำหนดให้คุณสมบัติ CommEvent มีค่าเป็น comEventCDTO(Carrier Detect Timeout Error) และกระตุ้นให้เกิดเหตุการณ์ OnComm

●DSRHolding

ผู้ใช้งานสามารถตรวจสอบการทำงานของขา DSR ได้ว่ามีสถานะลอจิก "1" หรือ "0"โดยค่าที่อ่านได้จะเป็นบูลีนTrue และ False ถ้าค่า DSRHolding เป็น True ขา DSRจะมีสถานะลอจิก "1" ถ้าค่า DSRHolding เป็น False ขา DSR จะมีสถานะลอจิก "0"

-รูปแบบการใช้งาน

Object.DSRHolding

เมื่อขา DSR เป็นลอจิก "1"(DSRHolding=True) และเกิดไทม์เอาต์ คอนโทรล MSComm จะกำหนดคุณสมบัติCommEvent มีค่าเป็น comEventDSRTO(Data Set Ready Timeout)และกระตุ้นให้เกิดเหตุการณ์ OnComm

●Break

ใช้ในการเซตและเคลียร์ค่าสัญญาณ Break ชนิดของข้อมูลเป็นแบบ Boolean

-รูปแบบการใช้งาน

Object.Break[=value]

โดย Value เป็นค่า Boolean ถ้า Value=True หมายถึง การส่งสัญญาณ Break ออกไป ถ้า Value=False หมายถึงการเคลียร์สัญญาณ Break

เมื่อกำหนดให้สัญญาณ Break เป็น True จะเป็นการหยุดการส่งข้อมูลชั่วคราวจนกว่าจะมีการสั่งให้สัญญาณ Break เป็น False ตัวอย่างโปรแกรมที่ เป็นวิธีการส่งสัญญาณ Break ออกไปเป็นช่วงเวลาสั้นๆที่ 1/10 ของวินาที

```
'Set the Break condition.
MSComm1.Break=True
'Set duration to 1/10 second.
Duration!=Timer+.1
'Wait for the duration to pass.
Do Until Timer > Duration!
Dummy = DoEvents()
Loop
'Clear the Break condition.
MSComm1.Break = False
```

● เหตุการณ์ OnComm

เหตุการณ์ OnComm จะถูกสร้างขึ้นเมื่อรู้ค่าคุณสมบัติ CommEvent มีการเปลี่ยนแปลงเพื่อแสดงผลการเปลี่ยนแปลงเหล่านั้นแบบทันทีทันใดหรือแสดงข้อผิดพลาดที่เกิดขึ้น

การใช้ MSComm เพื่อติดต่อกับฮาร์ดแวร์

จากรายละเอียดของ MSComm ที่กล่าวไปในตอนต้นนั้น จะเห็นได้ว่าวิธีการที่จะอ่านค่าหรือเขียนค่าไปยังสถานะและขาควบคุมของพอร์ตอนุกรมสามารถทำได้ง่ายมาก โดยใช้คำสั่งเหล่านี้

DTREnable สำหรับสั่งให้ขา DTR มีลอจิก "0" หรือ "1"

RTSEnable สำหรับสั่งให้ขา RTS มีลอจิก "0"หรือ "1"

CTSHolding สำหรับอ่านค่าสถานะจากขา CTS ว่ามีลอจิก "0" หรือ "1"

CDHolding สำหรับอ่านค่าจากสถานะ CD ว่ามีลอจิก "0" หรือ "1"

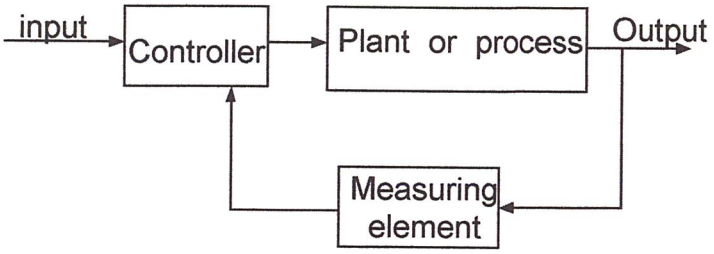
DSRHolding สำหรับอ่านค่าจากสถานะ DSR ว่ามีลอจิก "0" หรือ "1"

Break สำหรับการสั่งให้ขา TxD มีลอจิก "0" หรือ "1"

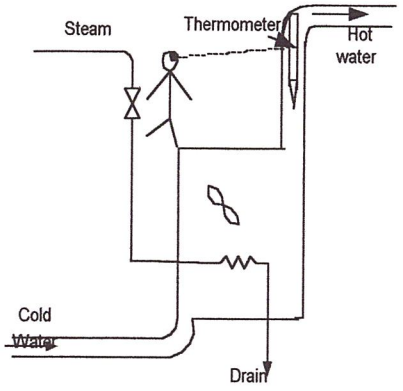
2.7 การควบคุมแบบ CLOSED-LOOP และแบบ OPEN-LOOP

ระบบควบคุมอาจแบ่งอย่างง่ายๆออกเป็นสองแบบคือระบบควบคุมแบบ closed-loop และระบบควบคุมแบบ open-loop

ระบบควบคุมแบบ closed-loop เป็นระบบควบคุมแบบหนึ่งซึ่งสัญญาณเอาท์พุทจะมีผลโดยตรงต่อการควบคุม ดังนั้นระบบควบคุมแบบ closed-loop ก็คือระบบควบคุมป้อนกลับนั่นเอง สัญญาณค่าความคลาดเคลื่อน (actuating error signal) ซึ่งเป็นสัญญาณแตกต่างระหว่างสัญญาณอินพุทแบบป้อนกลับ (feedback signal) จะถูกป้อนให้กลับตัวควบคุม (controller) เพื่อที่จะลดค่าความคลาดเคลื่อนให้น้อยลงและทำให้เอาท์พุทของระบบมีค่าตามที่ต้องการ สัญญาณป้อนกลับนี้อาจจะเป็นสัญญาณเอาท์พุทโดยตรงหรือเป็นสัญญาณที่เป็นฟังก์ชันของสัญญาณเอาท์พุท หรือเป็นค่าอนุพันธ์ของสัญญาณเอาท์พุทก็ได้ รูป 2.9 เป็น block diagram ที่แสดงถึงความสัมพันธ์ระหว่างอินพุทและเอาท์พุทของระบบควบคุมแบบ closed-loop ซึ่งจะเห็นได้ว่าเทอม closed-loop ก็หมายถึงการนำเอาวิธีการป้อนกลับมาใช้เพื่อลดค่าความคลาดเคลื่อนนั้นเอง และเพื่อให้เข้าใจถึงระบบควบคุมแบบ closed-loop ได้ดียิ่งขึ้น ให้พิจารณาถึงระบบทำความร้อน (thermal system) ซึ่งแสดงดังรูป2.10 ในกรณีนี้คนจะเปรียบเสมือนเป็นตัวควบคุม เขาต้องการที่จะรักษาอุณหภูมิของน้ำร้อนให้มีค่าคงที่ตามที่ต้องการ ดังนั้นเอาท์พุทของระบบนี้คือ ค่าของอุณหภูมิที่ต้องการการควบคุมอุณหภูมิในกรณีนี้จึงใช้เทอร์โมมิเตอร์(thermometer) ซึ่งติดไว้ที่ท่อทางออกของน้ำร้อนเป็นอุปกรณ์วัดอุณหภูมิ (measuring element) ถ้าผู้ควบคุมพบว่าเทอร์โมมิเตอร์แสดงค่าอุณหภูมิของน้ำร้อนสูงกว่าอุณหภูมิที่ต้องการ เขาก็จะต้องทำการลดปริมาณของไอน้ำที่ป้อนให้กับระบบเพื่อลดอุณหภูมิของน้ำร้อนลง ในทางตรงกันข้ามถ้าอุณหภูมิของน้ำร้อนต่ำกว่าค่าอุณหภูมิที่ต้องการ เขาก็จะทำการเปิดวาล์ว(valve) เพื่อให้ไอน้ำไหลผ่านเข้าระบบมากขึ้น ดังนั้นอุณหภูมิของน้ำร้อนก็จะร้อนขึ้นตามที่ต้องการได้



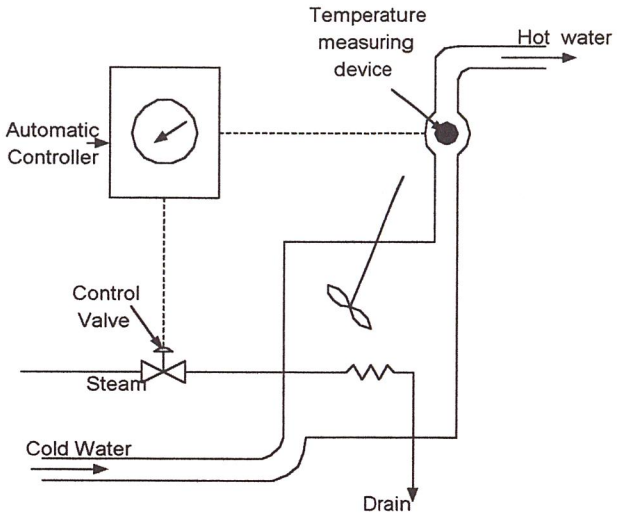
รูปที่ 2.9 ระบบควบคุมแบบ closed-loop



รูปที่ 2.10 ระบบควบคุมแบบป้อนกลับ manual ของระบบความร้อน

การควบคุมของระบบในรูป 2.10 นี้ถือเป็นการควบคุมแบบ closed-loop ทั้งนี้เพราะได้มีการนำเอาสัญญาณเอาท์พุท(อุณหภูมิของน้ำ) ป้อนกลับไปเทียบสัญญาณอินพุทอ้างอิงและตัวควบคุมในกรณีนี้ก็คือคนที่ทำหน้าที่ปิด-เปิดวาล์ว แต่อย่างไรก็ตามระบบควบคุมแบบ closed-loop นี้ถือเป็นระบบควบคุมแบบ manual ไม่ใช่ระบบควบคุมอัตโนมัติ

ถ้านำตัวควบคุมแบบอัตโนมัติ (automatic controller) มาใช้แทนคนซึ่งทำหน้าที่ปิด-เปิดวาล์วดังแสดงในรูป 2.11 แล้ว ระบบควบคุมแบบ closed-loop นี้จะเป็นระบบควบคุมแบบอัตโนมัติหรือระบบควบคุมแบบป้อนกลับแบบอัตโนมัติ จากรูป 2.11 ตำแหน่งของ dial บนตัว ควบคุมอัตโนมัติจะเป็นตัวกำหนดค่าของอุณหภูมิที่ต้องการ ส่วนเอาท์พุทซึ่งเป็นค่าจริงของน้ำร้อนจะถูกทำการวัดอุณหภูมิโดยอุปกรณ์วัดอุณหภูมิ ค่าของอุณหภูมิที่ต้องการและอุณหภูมิจริงจะถูกนำมาเปรียบเทียบกันเพื่อที่จะสร้างสัญญาณค่าความคลาดเคลื่อนขึ้น ในการเปรียบเทียบกันนี้เอาท์พุทซึ่ง



รูปที่ 2.11 ระบบควบคุมป้อนกลับแบบอัตโนมัติของระบบความร้อน

เป็นค่าของอุณหภูมิจะต้องถูกแปลงให้มีหน่วยเดียวกันกลับอินพุทหรือค่าที่กำหนดไว้ set point โดยใช้ transducer ซึ่งเป็นอุปกรณ์ที่ใช้แปลงพลังงานรูปหนึ่งไปเป็นพลังงานอีกรูปหนึ่ง

สัญญาณค่าความคลาดเคลื่อนที่ถูกสร้างขึ้นมาในตัวควบคุมแบบอัตโนมัติจะถูกส่งไปยังภาคขยายเพื่อให้มีขนาดเพียงพอที่จะนำไปทำให้วาล์วควบคุม(control valve) เปลี่ยนตำแหน่งคือเปิดมากขึ้นหรือปิดมากขึ้นเพื่อที่จะทำให้อุณหภูมิของน้ำร้อนมีค่าคงที่ตามต้องการได้และเมื่อสัญญาณค่าความคลาดเคลื่อนมีค่าเท่ากับอุณหภูมิที่กำหนดไว้ วาล์วควบคุมก็จะไม่มีการเปลี่ยนแปลงแบบเปิดปิดอีก

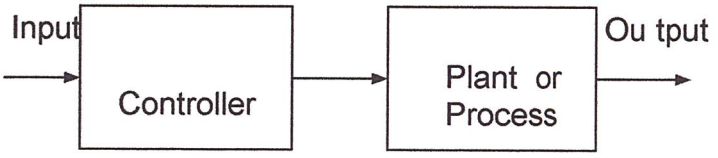
การเปลี่ยนแปลงของอุณหภูมิสภาพแวดล้อม ,อุณหภูมิของน้ำเย็นทางด้านเข้า และอื่นๆ ของระบบความร้อนดังที่กล่าวมานั้นจะถือว่าเป็นอินพุทที่เกิดจากภายนอก นอกจากนี้การทำงานของระบบควบคุมป้อนกลับแบบ manual ในรูป นั้นก็คล้ายคลึงกับการทำงานของระบบควบคุมป้อนกลับแบบอัตโนมัติด้วย กล่าวคือตา,สมองและกล้ามเนื้อของผู้ควบคุมวาล์ว นั้นจะคล้ายคลึง(analog)หรือเทียบได้กับอุปกรณ์วัดค่าความคลาดเคลื่อน,ตัวควบคุมอัตโนมัติและตัวกระทำ(actuator) ของระบบ

ในกรณีที่ระบบต้องการจะควบคุมเป็นระบบที่ยุ่งยากซับซ้อน การควบคุมโดยใช้คน(หรือแบบ manual)จะไม่มีประสิทธิภาพเพราะจะมีตัวแปรหลายตัว และตัวแปรแต่ละตัวจะมีความสัมพันธ์ซึ่งกันและกันด้วย ดังนั้นการควบคุมจึงควรจะเป็นการควบคุมแบบอัตโนมัติ การควบคุมแบบอัตโนมัตินั้นทำให้สามารถกำจัดความคลาดเคลื่อนอันเนื่องมาจากการปฏิบัติการผิดพลาดของคนได้ และยังทำให้การควบคุมมีความเที่ยงตรงแม่นยำสูง

ระบบควบคุมแบบ closed-loop นั้นสามารถจะพบเห็นได้ทั่วไปทั้งในงานอุตสาหกรรม หรือในบ้าน ตัวอย่างของระบบควบคุมแบบ closed-loop ได้แก่ระบบ servomechanisms ทั้งหมด ระบบควบคุมกระบวนการ ตู้เย็นที่ใช้ในบ้าน การควบคุมอุณหภูมิในห้อง เป็นต้น

ระบบควบคุมแบบ Open-Loop

ระบบควบคุมแบบ Open-Loop เป็นระบบควบคุมที่เอาท์พุทของระบบจะไม่มีผลต่อการควบคุมเลย นั่นคือในกรณีของระบบควบคุมแบบ Open-Loop นั้น เอาท์พุทของระบบจะไม่ถูกวัดหรือถูกป้อนกลับเพื่อนำมาเปรียบเทียบกลับอินพุท รูป 2.12 เป็น block diagram ซึ่งแสดงถึงความสัมพันธ์ระหว่างอินพุทและเอาท์พุทของระบบควบคุมแบบ Open-Loop ตัวอย่างที่ง่ายๆของระบบควบคุมแบบ Open-Loop ได้แก่ เครื่องซักผ้า กล่าวคือขั้นตอนของการดูดน้ำเข้า การซักและการปล่อยน้ำทิ้งนั้นจะเป็นไปตามเวลาที่ได้กำหนดไว้ล่วงหน้า แต่เครื่องซักผ้าไม่ได้มีการวัดเอาท์พุทซึ่งก็คือความสะอาดของผ้าออกมาแต่อย่างใด



รูปที่ 2.12 ระบบควบคุมแบบ OPEN LOOP

การปรับค่าตัวควบคุม PID

การปรับค่าพารามิเตอร์ของตัวควบคุม เป็นสิ่งจำเป็นเพื่อให้ได้การควบคุมที่ดีที่สุด ซึ่งทำได้หลายวิธี บางวิธีจะพิจารณาจากประสบการณ์ บางวิธีจะอาศัยคณิตศาสตร์ และส่วนใหญ่จะพิจารณาจากเงื่อนไขในโดเมนเวลามากกว่าในโดเมนความถี่

พิจารณาฟังก์ชันการถ่ายโอนของตัวควบคุมแบบ PID ดังนี้

$$G_c(s) = K_p\left(1 + \frac{1}{T_i s} + T_d s\right)$$

โดยที่

K_p Proportional Gain

T_I Integral Time

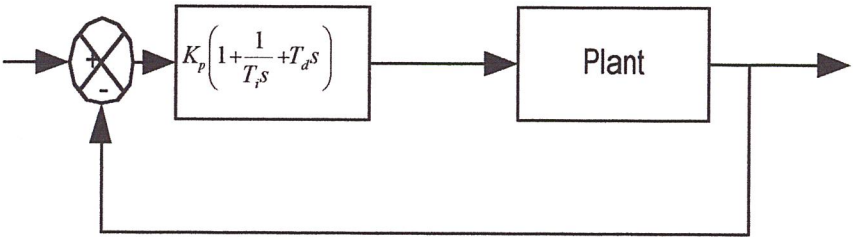
T_d Derivative Time

เมื่อ e(t) และ u(t) คืออินพุทและเอาท์พุทของตัวควบคุม จะได้ว่า

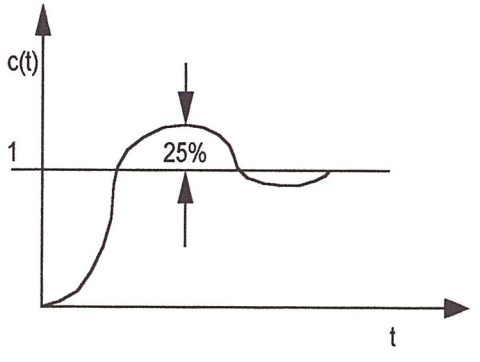
$$u(t) = K_p \left[e(t) + \frac{1}{T_i} \int_{-\infty}^{t} e(t)dt + T_d \frac{de(t)}{dt} \right]$$

การปรับค่าของตัวควบคุม PID โดยวิธีของ Ziegler-Nichols

ในการหาค่า K_p ,T_I และ T_d จะขึ้นอยู่กับคุณลักษณะของผลตอบสนองชั่วครู่ของระบบที่ถูกควบคุม ซึ่งมีอยู่ 2 วิธี แต่ละวิธีมีจุดมุ่งหมายที่จะทำให้ผลตอบสนองเวลาของระบบต่ออินพุทแบบ Unit Step มีค่าของ Maximum Overshoot ไม่เกิน 25% ดังรูป



รูปที่ 2.13 BLOCK DIAGRAM ของกระบวนการ



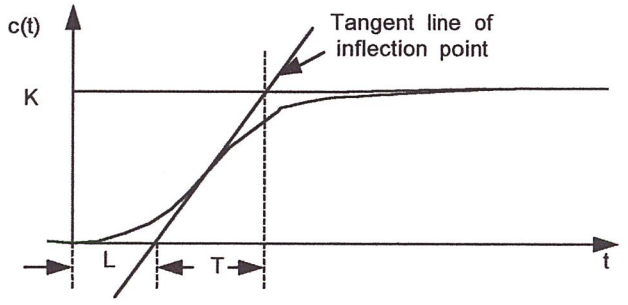
รูปที่ 2.14 ผลตอบสนองของกระบวนการ

<u>วิธีที่ 1</u> Process Reaction Curve

วิธีนี้จะหาค่าพารามิเตอร์ของตัวควบคุม PID จากผลตอบสนองเวลาของระบบหรือกระบวนการที่ถูกควบคุมต่ออินพุทแบบ Step โดยที่ระบบหรือกระบวนการในกรณีนี้จะไม่มี Pole ที่จุด Origin หรือไม่มี Dominant Complex-Conjugate Poles และไม่มีตัวควบคุมต่อร่วมอยู่ ดังนั้นผลตอบสนองเวลาจะเป็นรูปตัว S ดังในรูป(ถ้าผลตอบสนองไม่เป็นรูปตัว S วิธีนี้จะใช้ไม่ได้) ซึ่งสามารถจะแยกพิจารณาเป็นค่าคงที่ 2 ตัว คือค่าของ Delay Time L และเวลาคงที่ T(Time Constant) จากนั้นให้ลากเส้นสัมผัสกับจุดที่มีอัตราการเบี่ยงเบนสูงสุดเส้นนี้จะตัดกับแกนเวลาและเส้นตรงของเอาต์พุท c(t)=K ที่

$$\frac{C(s)}{U(s)} = \frac{Ke^{-Ls}}{Ts+1}$$

ดังนั้น ฟังก์ชันการถ่ายโอนจะประมาณได้ด้วยระบบอันดับหนึ่งที่มี Transport lag คือ



รูปที่ 2.15 ผลตอบสนองของกระบวนการ

Ziegler-Nichols ได้กำหนดค่าของ K_p,T_i และ T_d สำหรับตัวควบคุมแบบต่างๆดังนี้

1. ตัวควบคุมแบบ P

$K_p = T/L$

2. ตัวควบคุมแบบ PI

$K_p=0.9T/L$

$T_i=L/0.3$

3.ตัวควบคุมแบบ PID

$K_p = 1.2T/L$

$T_i=2L$

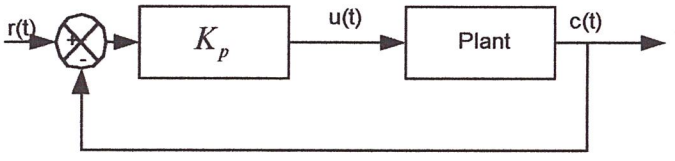
$T_d=0.5L$

วิธีที่ 2 Ultimate Method

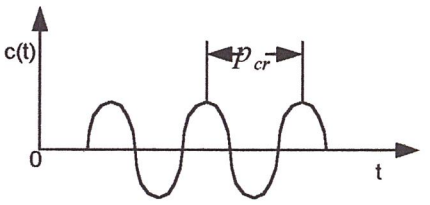
วิธีนี้จะหาค่าพารามิเตอร์ของตัวควบคุม PID จากผลตอบสนองเวลาของระบบหรือกระบวนการที่ถูกควบคุมด้วยตัวควบคุมแบบ P ต่ออินพุทแบบ Unit Step โดยปรับค่าของ K_p ไปเรื่อยๆ จนผลตอบสนองต่อเวลาเกิดการแกว่งอย่างต่อเนื่อง ดังในรูป(ถ้าผลตอบสนองเวลาไม่เกิดการแกว่งอย่างต่อเนื่องวิธีนี้จะใช้ไม่ได้) จากนั้นหาค่าของ

K_{cr}(Critical Gain) เป็นอัตราการขยายที่ทำให้ผลตอบสนองเวลาเกิดการแกว่งอย่างต่อเนื่อง

P_{cr}(Oscillation Period) เป็นคาบเวลาของการแกว่งอย่างต่อเนื่อง



รูปที่ 2.16 BLOCK DIAGRAM ของกระบวนการ



รูปที่ 2.17 ผลตอบสนองของกระบวนการ

Ziegler-Nichols ได้กำหนดค่าของ K_p,T_i และ T_dสำหรับตัวควบคุมแบบต่างๆดังนี้

1.ตัวควบคุมแบบ P

K_{cr}=0.5 K_{cr}

2. ตัวควบคุมแบบPI

K_p= 0.45 K_{cr}

T_i=P_{cr}/1.2

3.ตัวควบคุมแบบ PD

K_p= 0.6 K_{cr}

T_d=0.125P_{cr}

4.ตัวควบคุมแบบ PID

$K_p = 0.6\,K_{cr}$

$T_i = 0.5P_{cr}$

$T_d = 0.125P_{cr}$

การปรับค่าของตัวควบคุม PID โดยวิธี Damped Oscillation

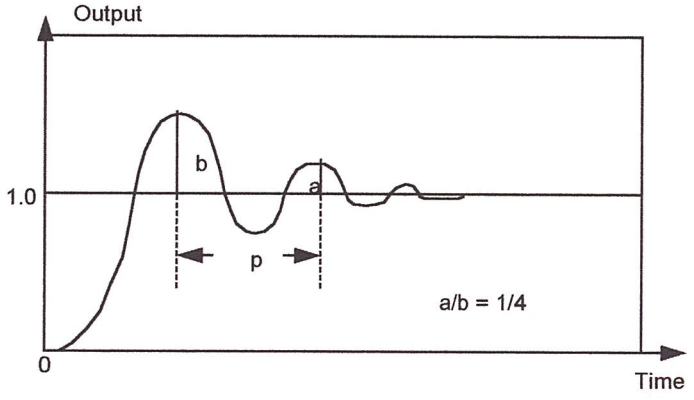
วิธีนี้ปรับปรุงมาจาก Ultimate Method โดย Harriott เพื่อใช้ในกรณีที่การปรับค่าของ K_p ไปอย่างไรก็ตามแต่ผลตอบสนองเวลาไม่เกิดการแกว่งอย่างต่อเนื่อง

วิธี Damped Oscillation นี้จะปรับค่าของ K_p ไปจนผลตอบสนองเวลาของระบบควบคุมแบบลูปปิดมีอัตราการเสื่อม 1/4 ดังรูป จากนั้นวัดค่าของ P และใช้ค่าของ $K_p(1/4)$ เพื่อคำนวณหาค่า K_p, T_i และ T_d ดังนี้

$K_p = K_p(1/4)$

$T_i = P/1.5$

$T_d = P/6$



รูปที่ 2.18 ผลตอบสนองของกระบวนการ

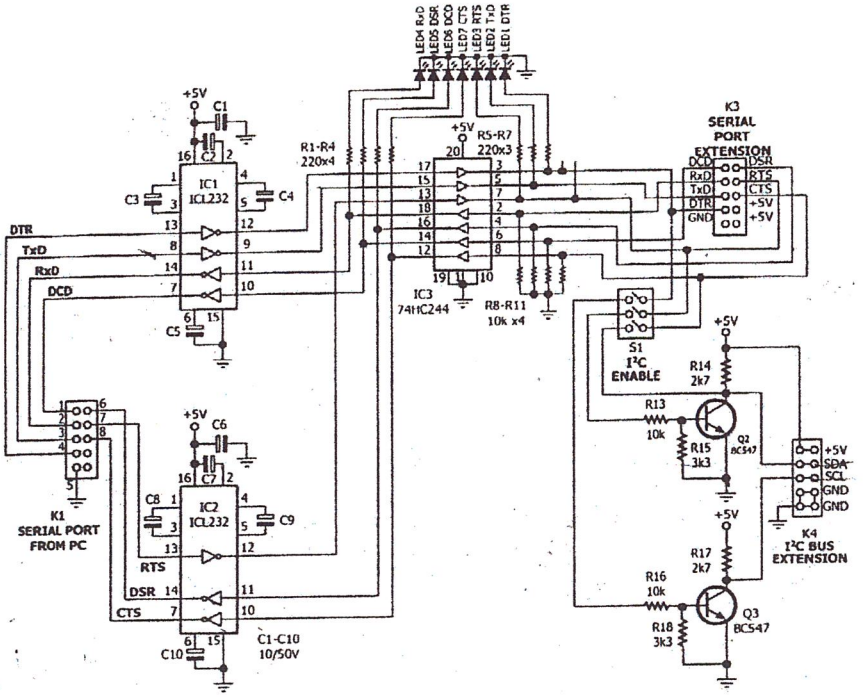
จะต้องระลึกอยู่เสมอว่า การคำนวณพารามิเตอร์ของ PID โดยวิธีของ Ziegler-Nichols นั้น ไม่ใช่ค่าที่เที่ยงตรงที่จะนำไปใช้งานได้ทันที และมี Maximum Overshoot 25% ตามที่กล่าวไว้แต่จะเป็นเพียงค่าที่ใกล้เคียงเท่านั้น ผู้ควบคุมจะต้องทำการปรับค่าพารามิเตอร์เหล่านี้แบบละเอียด (Fine Tuning) อีกครั้งหนึ่ง

บทที่ 3

การสร้างอุปกรณ์ และโปรแกรม

3.1 ทางฮาร์ดแวร์

3.1.1 บอร์ดแปลงสัญญาณแบบ I^2C เป็น RS-232 สามารถต่อได้ตามวงจรดังรูป



รูปที่ 3.1 บอร์ดแปลงสัญญาณแบบ I^2C เป็น RS-232

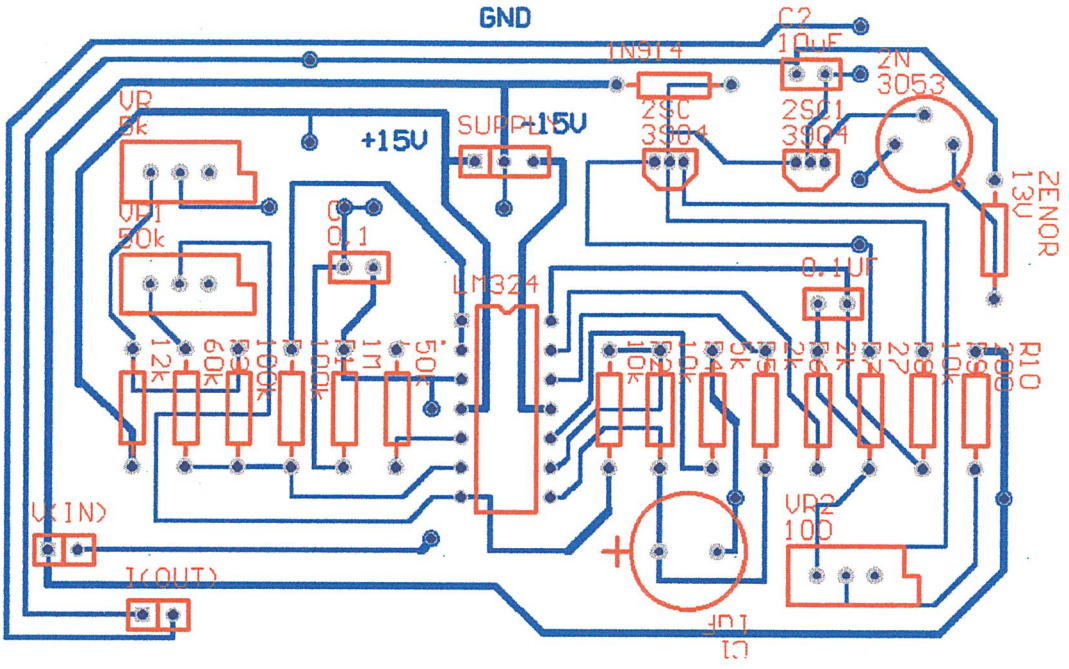
3.1.2 บอร์ด อินพุท-เอาท์พุท ที่ประกอบด้วย

-ตัวแปลงแรงดันเป็นแรงดัน(1-5 เป็น 0-5 โวลท์)

-ตัวแปลงแรงดันไฟฟ้าเป็นกระแสไฟฟ้า

-ไอซี PCF 8591

ได้ทำการออกแบบวงจรใน พีซีบี(PCB)ได้วงจรดังรูป 3.2



รูปที่ 3.2 บอร์ด อินพุท-เอาท์พุท

3.2 ทางด้านซอฟแวร์

3.2.1 การเขียนโปรแกรมเกี่ยวกับมาตรฐาน I^2C

- โปรแกรมย่อยการสร้างสัญญาณ Start ให้กับ I^2C สถานะเริ่มต้นของ I^2C จะเริ่มจาก
  1) ขา SCL และขา SDA ต้องมีลอจิก “1” ก่อนเพื่อกำหนดให้อยู่ในสถานะบัสว่าง
  2) กำหนดให้ขา SDA มีลอจิก “0” ก่อน
  3) จากนั้นจึงกำหนดให้ขา SCL มีลอจิก “0” ตามมา

  สามารถเขียนโปรแกรมได้ดังนี้

```
Private Sub I2CStart()
MSComm1.RTSEnable = True   'SDA = 1
MSComm1.DTREnable = True   'SCL = 1
MSComm1. RTSEnable = False   'SDA = 0
MSComm1. DTREnable = False   'SCL = 0
End Sub
```

-โปรแกรมย่อยการสร้างสัญญาณ Stop ให้กับ I^2C มีขั้นตอนดังนี้

1) กำหนดให้ขา SDA มีลอจิก "0" ก่อน
2) จากนั้นกำหนดให้ขา SCL มีลอจิกเป็น "1"
3) ให้ขา SDA มีสถานะลอจิกเป็น "1"

เขียนโปรแกรมย่อยได้ดังนี้

```
Private Sub I2CStop()
MSComm1. RTSEnable = False  'SDA = 0
MSComm1.DTREnable = True  'SCL = 1
MSComm1.RTSEnable = True  'SDA = 1
End Sub
```

-โปรแกรมย่อยเพื่อสร้างสถานะอื่นบนบัส I^2C

โปรแกรมย่อยสถานะ ACK

```
Private Sub ACK()
MSComm1.RTSEnable = True  'SDA = 1
MSComm1.DTREnable = True  'SCL = 1
MSComm1. DTREnable = False  'SCL = 0
End Sub
```

-โปรแกรมย่อยสถานะ MACK

```
Private Sub MAck()
MSComm1.RTSEnable = False  'SDA=0
MSComm1.DTREnable = True  'SCL=1
MSComm1.DTREnable = False  'SCL=0
MSComm1.RTSEnable = True  'SDA=1
End Sub
```

-โปรแกรมย่อยส่งข้อมูลลอจิก "0"

```
Private Sub Send0 ()
MSComm1. RTSEnable = False  'SDA = 0
MSComm1.DTREnable = True  'SCL = 1
MSComm1. DTREnable = False  'SCL = 0
End Sub
```

-โปรแกรมย่อยส่งข้อมูลลอจิก “1”

```
Private  Sub  Send1 ()
MSComm1.RTSEnable = True   'SDA = 1
MSComm1.DTREnable = True   'SCL = 1
MSComm1. DTREnable = False   'SCL = 0
End  Sub
```

-โปรแกรมย่อยในการส่งข้อมูล 8 บิต

```
Private Sub Send8BIT(Add As Integer)
  For I = 7 To 0 Step -1              ' Loop 7 Cycle
     If (Add And 2 ^ I) = 2 ^ I Then   'Test Bit 0 OR 1
       Call Send1
     Else
       Call send0
     End If
  Next I
End Sub
```

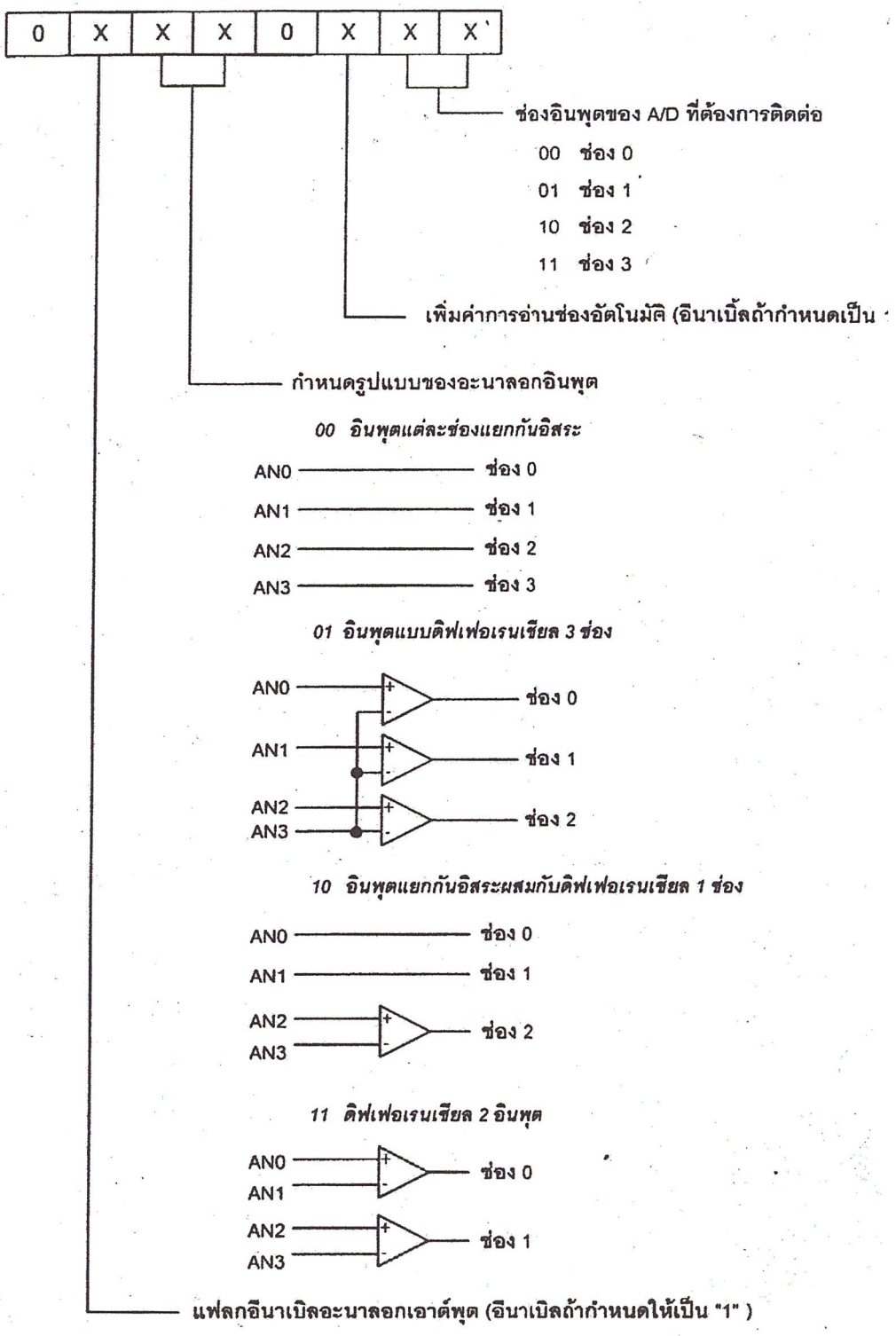
3.2.2 การรับค่าและการส่งค่าของ IC PCF8591

<u>รายละเอียดฟังก์ชันต่างๆของ IC PCF8591</u>

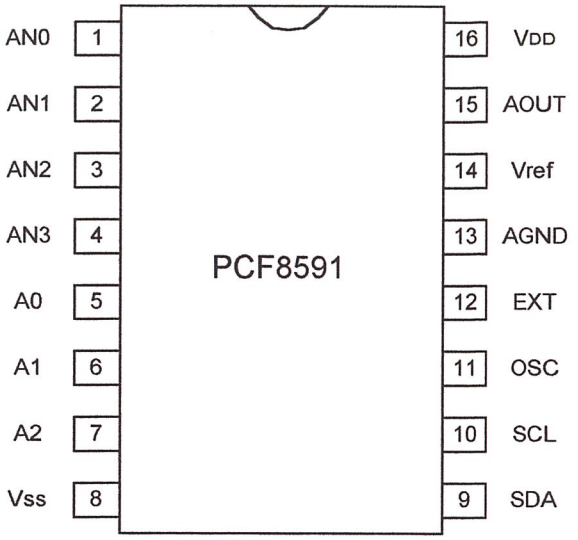
ตำแหน่งแอดเดรสดังได้กล่าวในบทที่ 2 เรื่องรูปแบบข้อมูลกำหนดแอดเดรสที่ใช้การอ้างถึงแบบ 7 บิต ข้อมูลกำหนดแอดเดรส 4 บิตบน จะเป็นค่าแอดเดรสเฉพาะของอุปกรณ์ตัวนั้นๆที่กำหนดมาจากผู้ผลิต ผู้ใช้งานไม่สามารถเปลี่ยนแปลงได้ สำหรับ IC PCF8591จะมีค่าเท่ากับ 1001_2 ข้อมูล 3 บิตถัดมา จะเป็นค่าของแอดเดรสที่ผู้ใช้งานสามารถกำหนดได้ ทางฮาร์ดแวร์ ส่วนบิต LSB ใช้ในการกำหนดว่าต้องการอ่านหรือเขียนข้อมูล ถ้าเป็น 0 หมายถึงต้องการเขียนข้อมูล ถ้าเป็น 1 หมายถึงต้องการอ่านข้อมูล

ข้อมูลควบคุม

หลังจากส่งข้อมูลกำหนดแอดเดรสแล้วต้องส่งข้อมูลควบคุมตามไปด้วย เพื่อกำหนดคุณสมบัติ ของ IC PCF8591 รายละเอียดแสดงดังรูป



รูปที่ 3.3 แสดงรายละเอียดข้อมูลควบคุมที่เขียนลงในรีจิสเตอร์ควบคุมภายในไอซี PCF 8591



รูปที่ 3.4 รายละเอียดของ IC PCF8591

การอ่านข้อมูลอินพุทอะนาลอกจาก PCF8591 มีลำดับขั้นตอนดังนี้

1) ส่งสัญญาณ Start
2) ส่งข้อมูลกำหนดแอดเดรส บิตที่กำหนดแอดเดรสสามารถกำหนดได้ว่าจะติดต่อกับ IC PCF8591ตัวใด เช่น ให้ขา A0,A1,A2 มีลอจิกเป็น 000 และให้ทำงานในโหมดเขียนข้อมูล คือ LSB มีลอจิกเป็น "0" จะได้ข้อมูลกำหนดแอดเดรสเป็น 10010000_2 หรือเท่ากับ 90H
3) รอรับสัญญาณ ACK หรือรอรับการตอบกลับจาก IC PCF8591
4) ส่งข้อมูลควบคุม IC PCF8591

   ตัวอย่างเช่นต้องการอินาเบิลอะนาลอกเอาท์พุท,กำหนดให้อินพุทอะนาลอกทำงานในโหมดซิงเกิล,กำหนดให้ใช้การเลื่อนอ่านข้อมูลแบบต่อเนื่อง และเริ่มอ่านข้อมูลจากช่องที่ 1 จะได้ข้อมูลควบคุมดังนี้ 01000101_2 หรือเท่ากับ 45H
5) รอรับสัญญาณ ACK จาก PCF8591
6) ส่งสัญญาณ Stop
7) ส่งสัญญาณ Start
8) ส่งข้อมูลกำหนดแอดเดรสอีกครั้ง โดยครั้งนี้กำหนดให้เป็นโหมดอ่านข้อมูลคือให้ LSB มีลอจิกเป็น "1"
9) รอรับสัญญาณ ACK จาก PCF8591

10) อ่านค่าจากขาอินพุทของวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอลช่องที่ 1 โดยโปรแกรมย่อยที่ใช้ในการรับค่า

11) ส่งสัญญาณ MACK (Master ACK) ไปยัง PCF8591

12) อ่านค่าจากขาอินพุทของวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอลช่องที่ 2

13) ส่งสัญญาณ MACK (Master ACK) ไปยัง PCF8591

14) อ่านค่าจากขาอินพุทของวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอลช่องที่ 3

15) ส่งสัญญาณ MACK (Master ACK) ไปยัง PCF8591

16) อ่านค่าจากขาอินพุทของวงจรแปลงสัญญาณอะนาลอกเป็นดิจิตอลช่องที่ 4

17) รอรับสัญญาณ ACK

18) ส่งสัญญาณ STOP

จากขั้นตอนดังกล่าวสามารถเขียนโปรแกรมย่อยได้ดังนี้

```
Private Sub Timer1_Timer()
Call I2CStart
Call Send8BIT(&H90)
Call Ack
Call Send8BIT(&H45)
Call Ack
Call I2CStop
Call I2CStart
Call Send8BIT(&H91)
Call Ack
'Debug.Print DAT
Text1.Text = (DAT * 5) / 255
Call MAck
Text2.Text = (DAT * 5) / 255
Call MAck
Text3.Text = (DAT * 5) / 255
Call MAck
Text4.Text = (DAT * 5) / 255
Call Ack
```

```
Call I2CStop
End Sub
```

โปรแกรมของฟังก์ชัน DAT เป็นดังนี้

```
Private Function DAT()
 For I = 7 To 0 Step -1
    MSComm1.RTSEnable = True        'SDA=1
    MSComm1.DTREnable = True        'SCL=1
    If Not MSComm1.CTSHolding Then   'Read SDA
      DAT1 = 2 ^ I Or DAT1
    End If
    MSComm1.DTREnable = False       'SCL=0
 Next I
 DAT = DAT1                         'Data 8 Bit
End Function
```

การเขียนข้อมูลไปยังวงจรแปลงสัญญาณดิจิตอลเป็นอะนาลอกของ PCF 8591

มีขั้นตอนดังนี้

1. ทำการส่งสัญญาณ START
2. ส่งข้อมูลกำหนดแอดเดรสโดยให้ทำงานในโหมดเขียนข้อมูลคือ บิต LSB มีลอจิก “0”
3. รอรับสัญญาณ ACK
4. ส่งข้อมูลควบคุม 01000100_2 หรือ 44Hไปยัง PCF 8591 เพื่ออินาเบิลขาอะนาลอกเอาท์พุท
5. รอรับสัญญาณ ACK
6. ส่งข้อมูลไปยังเอาท์พุทอะนาลอก โดยค่าที่ส่งออกไปจะต้องมีค่าอยู่ระหว่าง 0-255
7. รอรับสัญญาณ ACK
8. ส่งสัญญาณ STOP

สามารถนำขั้นตอนดังกล่าวมาเขียนเป็นโปรแกรมย่อยได้ดังนี้

โปรแกรมย่อยส่งข้อมูลไปยังอะนาลอกเอาท์พุท

```
Private Sub Text5_Change()
If Val(Text5.Text) > 5 Then Text5.Text = 5
Call I2CStart
```

```
Call Send8BIT(&H90)
Call Ack
Call Send8BIT(&H44)
Call Ack
'Debug.Print Val(Text5.Text)
Call Send8BIT(CInt(Val(Text5.Text) * 51))
Call Ack
Call I2CStop
End Sub
```

3.2.3 สมการ PID

จากข้อ 3.2.1 และ 3.2.2 เราจะทำการควบคุมกระบวนการอย่างอัตโนมัติโดยจะนำค่าข้อมูลที่รับเข้ามาไปคำนวณในสมการ PID และส่งค่าที่คำนวณได้ไปควบคุมกระบวนการ จะเขียนโปรแกรมฟังก์ชัน PID ได้ดังนี้

```
Private Function PID(b As Single) As Single
Mvp = Kp * (SV - b)
Mvi = ((Kp * 0.5) / (2 * Ti)) * ((2 * SV) - (3 * b) + PV1) + Mvi
Mvd = Kp * Td * (PV1 - b)
c = Mvp + Mvi + Mvd
PV1 = b
 If c > 5 Then
  c = 5
  End If
PID = c
End Function
```

3.2.4 การแสดงผลด้วยกราฟ

เมื่อมีการรับค่าข้อมูลและคำนวณในสมการ PID และมีการส่งค่าไปควบคุมกระบวนการแล้ว จะนำค่าข้อมูลที่รับ(PV),ข้อมูลที่ส่ง(MV)และค่าข้อมูลที่ต้องการ (SV) ไปแสดงผลด้วยกราฟ โดยจะมีรูปแบบของโปรแกรมดังนี้

TChart1.Series(*หมายเลขของเส้นกราฟ*).Add *ค่าที่ต้องการให้แสดง*, "", *สีที่กำหนด*

3.2.5 การบันทึกค่า

เราสามารถบันทึกค่าของกราฟด้วยรูปแบบของโปรแกรมดังนี้

TChart1.Export.SaveToFile (*ตำแหน่งและชื่อของไฟล์ที่ต้องการนำกราฟไปเก็บ*)

3.2.6 Data base

การสร้างและการจัดการฐานข้อมูล

เราจะใช้ Visual Data Manager ในการสร้างฐานข้อมูลและทำการสร้างตารางในฐานข้อมูล การสร้างตารางฐานข้อมูลทำได้ดังนี้

1. เปิดโปรแกรม Visual Data Manager โดยเลือกเมนูคำสั่ง Add-Insแล้วเลือก Visual Data Manager จะปรากฏหน้าจอของ Visual Data Manager
2. การสร้างฐานข้อมูล หลังจากที่เปิดโปรแกรม Visual Data Manager ให้เลือกเมนูคำสั่ง File แล้วเลือก New แล้วเลือก Microsoft Access แล้วเลือก Version 7.0 MDB

- เลือกไดร์ฟ,ไดเรคทอรี่ที่ต้องการจะสร้างฐานข้อมูล แล้วใส่ชื่อฐานข้อมูลที่ต้องการในช่อง File name คลิกเมาส์ Save

3. การสร้างตาราง เมื่อสร้างฐานข้อมูลแล้วจากนั้นเลือกเมนูคำสั่ง File แล้วเลือก Open Database แล้วเลือก Microsoft Access เลือกไดร์ฟ,ไดเรคทอรี่และไฟล์ที่ต้องการเปิด แล้วคลิกปุ่ม Open

คลิกเมาส์ปุ่มขวาบริเวณใดก็ได้ใน Database Window หลังจากนั้นเลือกคำสั่ง New Table ใส่ชื่อตารางที่ต้องการในช่อง Table Name คลิกปุ่ม Add Field เพื่อเพิ่มฟิลด์ใหม่ในตาราง ใส่ชื่อฟิลด์ที่ต้องการในช่อง Name เลือกชนิดของฟิลด์ที่ต้องการในช่อง Type ใส่ขนาดของข้อมูลที่ต้องการในช่อง Size คลิกปุ่ม OK ทำการเพิ่มฟิลด์ใหม่เท่าที่ต้องการ ในโครงงานนี้ได้สร้าง Database ที่D:\Database\Process และทำการสร้างตารางชื่อ ProcessControl ซึ่งประกอบด้วยฟิลด์ 3 ฟิลด์คือ Plant,DATE และ Time

<u>การจัดการฐานข้อมูล</u>

1. การบันทึกค่าลงในฐานข้อมูล สามารถเขียนโปรแกรมโดยใช้ DAO(Data Access Objects)ได้ดังนี้

กำหนดตัวแปรต่างๆดังนี้

```
Public ProcessWs As Workspace, ProcessDb As Database, ProcessDyn As Recordset
Dim R2 As Date, R3 As Variant
```

เขียนโปรแกรมดังนี้

```
Private Sub Command5_Click()
```

```
Text6.Text = Date
Text7.Text = Time
A11 = Format(Text6.Text, "mmmddyyyy")
A21 = Format(Text7.Text, "hhmmss")
A31 = "c:\" & Label1.Caption & A11 & A21 & ".tee"
TChart1.Export.SaveToFile (A31)
Set ProcessWs = DBEngine.Workspaces(0)
Set ProcessDb = ProcessWs.OpenDatabase("D:\Database\Process.mdb", False, False)
Set ProcessDyn = ProcessDb.OpenRecordset("SELECT * FROM ProcessControl ",
dbOpenDynaset, dbAppendOnly)
ProcessDyn.AddNew
R2 = Text6.Text
R3 = Text7.Text
ProcessDyn("Plant") = Label1.Caption
ProcessDyn("DATE") = R2
ProcessDyn("Time") = R3
ProcessDyn.Update
ProcessDyn.Requery
ProcessDyn.Close
End Sub
```

อธิบายโปรแกรม เมื่อมีการคลิกที่ปุ่ม RECORD ที่ Text 6 จะมีการแสดงวันที่ ,Text7 แสดงเวลา จากนั้นจะมีการเปิด Database จากนั้นทำการเปิดตาราง และใช้คำสั่ง AddNew จากนั้นรับค่าต่างๆ โดยฟิลด์ Plant จะรับค่ามาจาก Label1 ซึ่งแสดงชื่อ Plant, ฟิลด์ DATE จะรับค่ามาจาก Text 6 ซึ่งแสดงวันที่ และฟิลด์ Time จะรับค่ามาจาก Text 7 ซึ่งแสดงเวลา จากนั้นใช้คำสั่ง Update เพื่อบันทึกค่าลงในตาราง ใช้คำสั่ง Close ทำการปิดตาราง

2. การดึงข้อมูลจากตารางจะมีรูปแบบโปรแกรมดังนี้

ที่ Form Load พิมพ์คำสั่ง

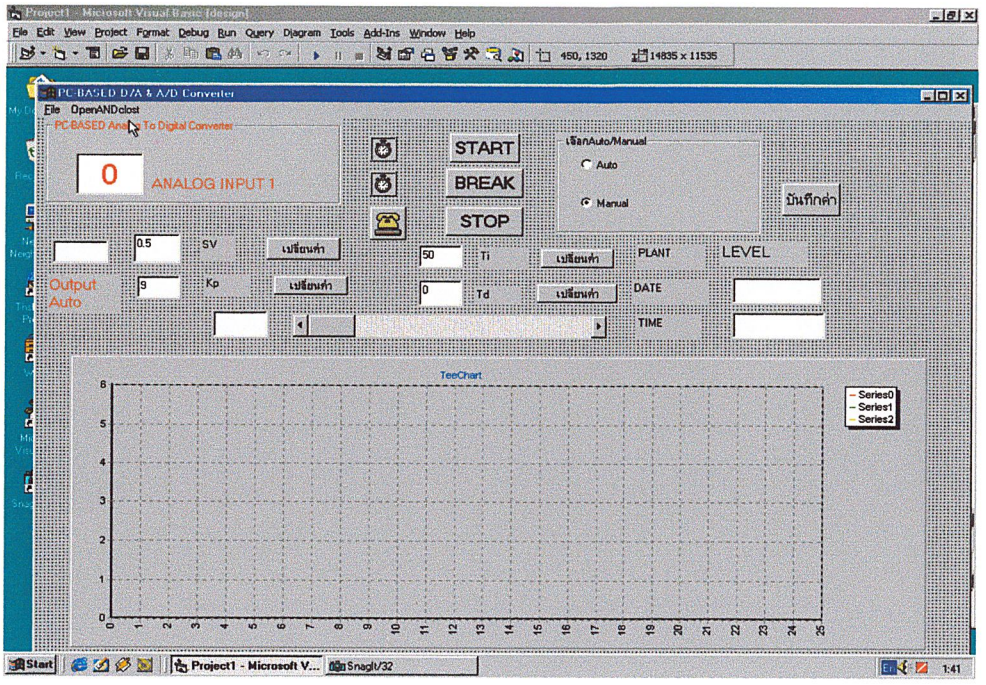
```
Set ProcessWs = DBEngine.Workspaces(0)
Set ProcessDb = ProcessWs.OpenDatabase("D:\Database\Process.Mdb", False, False)
```

```
Private Sub CmdOKDATE_Click()
If Combo2.Text <> "" Then
D = Combo2.Text
Combo2.Enabled = False
SQLCommand = "SELECT  Plant,DATE,Time  FROM ProcessControl"
Set ProcessSnap = ProcessDb.OpenRecordset(SQLCommand, dbOpenSnapshot)
ProcessSnap.MoveFirst
Combo3.Clear
Do While Not ProcessSnap.EOF
If ProcessSnap.Fields("Plant") = P And ProcessSnap.Fields("DATE") = D Then
Combo3.AddItem (ProcessSnap.Fields("Time"))
End If
ProcessSnap.MoveNext
Loop
End If
End Sub
```

อธิบายโปรแกรม ที่ Form Load ทำการเปิดโปรแกรมและตาราง จากโปรแกรมข้างต้นเมื่อมีการเลือก Plant,DATE จากนั้นกดปุ่ม OK จะทำการดึงข้อมูลเวลา Time เอาไปใส่ใน Combo3

3.2.7 **จากโปรแกรมย่อยต่างๆที่สร้างเอาไว้สามารถเขียนโปรแกรมที่ทำการควบคุม 1 กระบวนการตามขั้นตอนต่อไปนี้**

1) ออกแบบฟอร์มให้ได้ดังรูป



รูปที่ 3.5 แบบฟอร์มสำหรับการควบคุม 1 กระบวนการ

2) โปรแกรมเขียนได้ดังนี้

```
Dim K, DA, PAT, A As Integer
Dim SV As Single, c As Single, PV1 As Single
Dim Kp As Single, Ti As Single, Td As Single, T As Single
Dim Mvi As Single, Mvp As Single, Mvd As Single, G As Single, H As Single
Dim A1 As String, A2 As String, A3 As String
Private Sub Command1_Click() 'ปุ่ม STARTมีโค้ดดังนี้
If Option1.Value = True Then 'มีการเลือก Auto
SV = CSng(Text2.Text)
Kp = CSng(Text4.Text)
Ti = CSng(Text5.Text)
Td = CSng(Text6.Text)
If HScroll1.Enabled = True Then
```

```
 HScroll1.Enabled = False
 End If
MSComm1.PortOpen = True
Timer1.Enabled = True
ElseIf Option2.Value = True Then 'มีการเลือก Manual
SV = CSng(Text2.Text)
If HScroll1.Enabled = False Then
HScroll1.Enabled = True
End If
MSComm1.PortOpen = True
Text3_Change
Timer2.Enabled = True
End If
End Sub

Private Sub Command2_Click()  'ปุ่ม BREAK  มีโค้ดดังนี้
 Call I2CStart
Call Send8BIT(&H90)
Call Ack
Call Send8BIT(&H40)
Call Ack
Call Send8BIT(0)
Call Ack
Call I2CStop
  Timer2.Enabled = False
    Timer1.Enabled = False
If MSComm1.PortOpen = True Then
  MSComm1.PortOpen = False
  End If
  End Sub
```

```
Private Sub Command3_Click() 'ปุ่มเปลี่ยนค่าSV มีโค้ดดังนี้
SV = CSng(Text2.Text)

End Sub

Private Sub Command4_Click() 'ปุ่มเปลี่ยนค่าKp มีโค้ดดังนี้
Kp = CSng(Text4.Text)
End Sub

Private Sub Command5_Click() 'ปุ่มเปลี่ยนค่าTi มีโค้ดดังนี้
Ti = CSng(Text5.Text)
End Sub

Private Sub Command6_Click() 'ปุ่มเปลี่ยนค่า Td มีโค้ดดังนี้
Td = CSng(Text6.Text)
End Sub

Private Sub Command7_Click() 'ปุ่ม STOP มีโค้ดดังนี้
If MSComm1.PortOpen Then
   Call I2CStart
Call Send8BIT(&H90)
Call Ack
Call Send8BIT(&H40)
Call Ack
Call Send8BIT(0)
Call Ack
Call I2CStop
  Timer2.Enabled = False
MSComm1.PortOpen = False
```

```
End If
Unload Me
End Sub

Private Sub Command8_Click() 'ปุ่มบันทึกค่า มีโค้ดดังนี้
  Text8.Text = Date
  Text9.Text = Time
  A1 = Format(Text8.Text, "mmmddyyyy")
  A2 = Format(Text9.Text, "hhmmss")
  A3 = "c:\" & Label10.Caption & A1 & A2 & ".tee"
TChart1.Export.SaveToFile (A3)
End Sub
Private Sub Command9_Click()
End Sub
Private Sub HScroll1_Change()
Text3.Text = HScroll1.Value / 10
End Sub
Private Sub Mexit_Click()
  Unload Me
End Sub
Private Sub Send8BIT(Add As Integer)
 For i = 7 To 0 Step -1              ' Loop 7 Cycle
    If (Add And 2 ^ i) = 2 ^ i Then   'Test Bit 0 OR 1
      Call Send1
    Else
      Call send0
    End If
 Next i
End Sub
Private Function DAT()
```

```
  For i = 7 To 0 Step -1
     MSComm1.RTSEnable = True        'SDA=1
     MSComm1.DTREnable = True        'SCL=1
     If Not MSComm1.CTSHolding Then   'Read SDA
       dat1 = 2 ^ i Or dat1
     End If
     MSComm1.DTREnable = False       'SCL=0
  Next i
  DAT = dat1                         'Data 8 Bit
End Function
Private Sub I2CStart()
MSComm1.RTSEnable = True   'SDA=1
MSComm1.DTREnable = False  'SCL=0
MSComm1.DTREnable = True   'SCL=1
MSComm1.RTSEnable = False  'SDA=0
MSComm1.DTREnable = False  'SCL=0
End Sub
Private Sub I2CStop()
MSComm1.RTSEnable = False  'SDA=0
MSComm1.DTREnable = True   'SCL=1
MSComm1.RTSEnable = True   'SDA=1
End Sub
Private Sub send0()
MSComm1.RTSEnable = False  'SDA=0
MSComm1.DTREnable = True   'SCL=1
MSComm1.DTREnable = False  'SCL=0
End Sub
Private Sub Send1()
MSComm1.RTSEnable = True   'SDA=1
MSComm1.DTREnable = True   'SCL=1
```

```
MSComm1.DTREnable = False   'SCL=0
End Sub
Private Sub Ack()
MSComm1.RTSEnable = True    'SDA=1
MSComm1.DTREnable = True    'SCL=1
MSComm1.DTREnable = False   'SCL=0
End Sub
Private Sub MAck()
MSComm1.RTSEnable = False   'SDA=0
MSComm1.DTREnable = True    'SCL=1
MSComm1.DTREnable = False   'SCL=0
MSComm1.RTSEnable = True    'SDA=1
End Sub

Private Sub mnuclose2_Click()
Form1.Hide
End Sub
Private Sub mnuopen2_Click()
Form1.Show
End Sub

Private Sub Option1_Click()

End Sub

Private Sub Text3_Change() 'เป็น text ที่แสดงค่า MV ของ Manual
'จะทำงานเมื่อมีการเปลี่ยนแปลงค่า
If Val(Text3.Text) > 5 Then Text3.Text = 5
Call I2CStart
Call Send8BIT(&H90)
```

```
Call Ack
Call Send8BIT(&H40)
Call Ack
'Debug.Print Val(Text5.Text)
K = Val(Text3.Text)
Call Send8BIT(K * 51)
Call Ack
Call I2CStop
End Sub
Private Sub Timer1_Timer() ' Timer ที่ใช้ในการรับอินพุทของ Auto
G = 0
For i = 1 To 5
Call I2CStart
Call Send8BIT(&H90)
Call Ack
Call Send8BIT(&H40)
Call Ack
Call I2CStop
Call I2CStart
Call Send8BIT(&H91)
Call Ack
A = DAT
Call Ack
Call I2CStop
G = A + G
Next i
H = G / 5  'เฉลี่ยค่าinput
Text1.Text = Round((H * 5) / 255, 3)
TChart1.Series(0).Add (H * 5) / 255, "", clTeeColor
TChart1.Series(2).Add SV * 5, "", clTeeColor
```

```
If Val(Text1.Text) > 5 Then Text1.Text = 5
Timer1.Enabled = False
SendOutput
End Sub
Private Function PID(b As Single) As Single
Mvp = Kp * (SV - b)
Mvi = ((Kp * 0.5) / (2 * Ti)) * ((2 * SV) - (3 * b) + PV1) + Mvi
Mvd = Kp * Td * (PV1 - b)
 c = Mvp + Mvi + Mvd
PV1 = b
 If c > 5 Then
  c = 5
  End If
PID = c
End Function
Private Sub Form_Load()
If MSComm1.PortOpen Then      'Open Port
    MSComm1.PortOpen = False
 End If
 With Timer1
      .Enabled = False
      .Interval = 1000
 End With
Mvi = 0: Mvp = 0: Mvd = 0: SV = 0.5: Kp = 7: Td = 0: Ti = 100
PV1 = 0
  With Timer2
      .Enabled = False
      .Interval = 1000
 End With
```

```
End Sub
Private Sub Form_UnLoad(Cancel As Integer)
  If MSComm1.PortOpen Then        'Close Port
  MSComm1.PortOpen = False
  End If
End Sub
Private Sub SendOutput()  'การส่งOutputของAuto
Call I2CStart
Call Send8BIT(&H90)
Call Ack
Call Send8BIT(&H40)
Call Ack
Call Send8BIT(CInt(PID((H * 5 / 255) * 0.2) * 51))
T = PID((H * 5 / 255) * 0.2)
Text7.Text = T
TChart1.Series(1).Add T, "", clTeeColor
Call Ack
Call I2CStop
Timer1.Enabled = True
End Sub

Private Sub Timer2_Timer() 'Timer ที่ใช้ในการรับอินพุทของ Manual
G = 0
For i = 1 To 5
Call I2CStart
Call Send8BIT(&H90)
Call Ack
Call Send8BIT(&H40)
Call Ack
Call I2CStop
```

```
Call I2CStart
Call Send8BIT(&H91)
Call Ack
A = DAT
Call Ack
Call I2CStop
G = A + G
Next i
H = G / 5 'เฉลี่ยค่าinput
Text1.Text = Round((H * 5) / 255, 3)
PvTime = Time
TChart1.Series(0).Add (H * 5) / 255, "", clTeeColor
TChart1.Series(2).Add SV * 5, "", clTeeColor
TChart1.Series(1).Add K, "", clTeeColor
If Val(Text1.Text) > 5 Then Text1.Text = 5
End Sub
Private Sub Text3_Click()
Text3.SelStart = 0
Text3.SelLength = Len(Text5.Text)
End Sub
```

3.2.8 การสร้างโปรแกรมควบคุม 4 กระบวนการ

1) สร้าง Main Form ซึ่งเป็น MDI Form และใน MDI Form นี้จะมี Timer และ MSComm โดย Main Form จะเป็นฟอร์มควบคุมหลัก ซึ่ง Timer จะมีการทำงานทุกๆ 1 วินาที เมื่อ timer ทำงานจะมีการตรวจเช็คค่าว่ากระบวนการใดมีการกด Start และมีการเลือกการทำงานแบบ Auto หรือ Manual ก็จะทำการรับค่าอินพุทและส่งค่าอินพุทของกระบวนการนั้นๆ แสดงได้ดังโปรแกรม

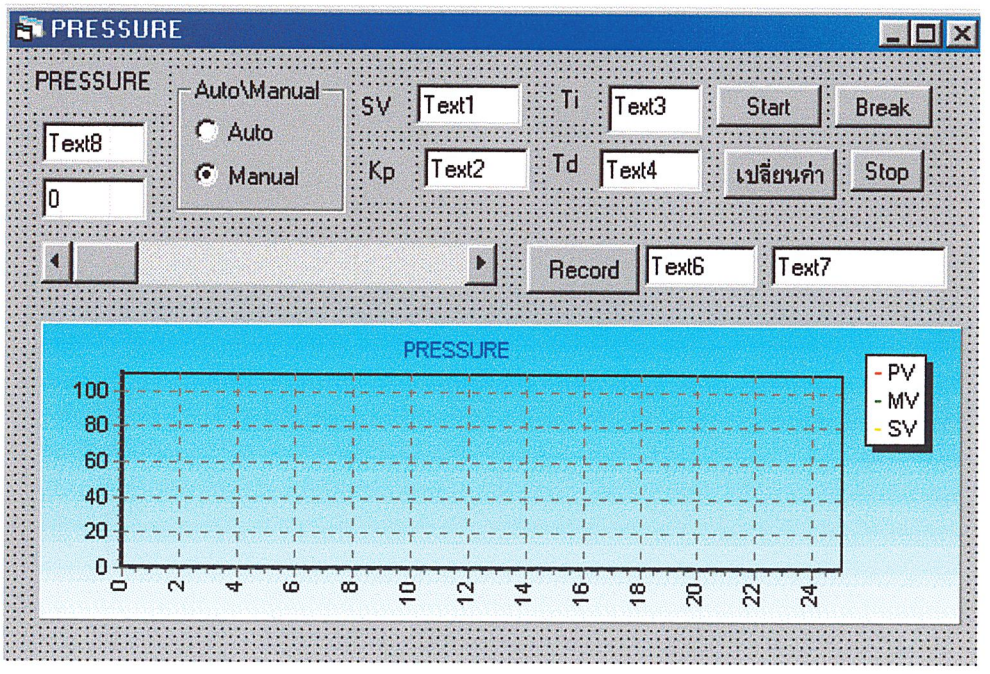
```
Private Sub Timer1_Timer()
If LevelForm.LEVSTART And LevelForm.Option1.Value Then
If MSComm1.PortOpen = False Then
MSComm1.PortOpen = True
```

```
    End If
    LevelForm.ReciveInput1
    LevelForm.SendOutputAuto1
  End If
  If LevelForm.LEVSTART And LevelForm.Option2.Value Then
    If MSComm1.PortOpen = False Then
    MSComm1.PortOpen = True
    End If
      LevelForm.ReciveInput1
    LevelForm.SendOutputManual1
    End If
 If LevelForm.LEVSTOP Then
    If MSComm1.PortOpen = False Then
    MSComm1.PortOpen = True
    End If
  Call I2CStart
 Call Send8BIT(&H90)
 Call Ack
 Call Send8BIT(&H44)
 Call Ack
 Call Send8BIT(0)
 Call Ack
 Call I2CStop
 End If

 If FlowForm.FLOWSTART And FlowForm.Option1.Value Then
    If MSComm1.PortOpen = False Then
    MSComm1.PortOpen = True
    End If
   FlowForm.ReciveInput2
```

```
 FlowForm.SendOutputAuto2
End If
If FlowForm.FLOWSTART And FlowForm.Option2.Value Then
 If MSComm1.PortOpen = False Then
  MSComm1.PortOpen = True
  End If
 FlowForm.ReciveInput2
  FlowForm.SendOutputManual2
  End If
      .
      .
Timer1.Enabled = False
Timer1.Enabled = True
End Sub
```

2) สร้างฟอร์มของกระบวนการต่างๆ โดยกำหนด Property MDIChild ให้เป็น True การสร้างฟอร์มต่างๆแสดงดังรูปตัวอย่างของการควบคุมระดับ



รูปที่ 3.6 แบบฟอร์มสำหรับการควบคุมความดัน

ส่วนโปรแกรมควบคุมต่างๆก็จะอาศัยหลักการเหมือนกับโปรแกรมควบคุมกระบวนการที่ได้ทำในหัวข้อ 3.2.7 สามารถดูโปรแกรมได้ในแผ่นดิสก์ ในการกำหนดแอดเดรสของ IC PCF8591 แต่ละตัว จะกำหนดดังนี้

-ที่กระบวนการควบคุมระดับ จะกำหนดเป็น 000 คำสั่งกำหนดแอดเดรสจะเป็น 90H ในการเขียนข้อมูล และจะเป็น 91H ในการอ่านค่า

-ที่กระบวนการควบคุมอัตราการไหล จะกำหนดเป็น 001 คำสั่งกำหนดแอดเดรสจะเป็น 92H ในการเขียนข้อมูล และจะเป็น 93H ในการอ่านค่า

-ที่กระบวนการควบคุมความดัน จะกำหนดเป็น 010 คำสั่งกำหนดแอดเดรสจะเป็น 94H ในการเขียนข้อมูล และจะเป็น 95H ในการอ่านค่า

-ที่กระบวนการควบคุมอุณหภูมิ จะกำหนดเป็น 011 คำสั่งกำหนดแอดเดรสจะเป็น 96H ในการเขียนข้อมูล และจะเป็น 97H ในการอ่านค่า

และข้อมูลควบคุมจะกำหนดให้เป็น 40H คือจะให้อินาเบิลอะนาลอกเอาท์พุท อินพุทแต่ละช่องแยกอิสระกัน ไม่มีการอ่านช่องอัตโนมัติ กำหนดให้อ่านที่ช่อง 0

3.2.9 โปรแกรมควบคุมแบบ Cascade

ในกระบวนการควบคุมระดับและควบคุมอัตราการไหลที่มีการควบคุมแบบ Cascade สมการ PID จะเขียนได้ดังนี้

```
Public Function PID1C(b1L As Single, b1F As Single) As Single
Mvp1L = Kp1L * (SV1 - b1L)
Mvi1L = ((Kp1L * 0.5) / (2 * Ti1L)) * ((2 * SV1) - (3 * b1L) + PV11L) + Mvi1L
Mvd1L = Kp1L * Td1L * (PV11L - b1L)
c1L = Mvp1L + Mvi1L + Mvd1L
PV11L = b1L
SV1F = c1L
Mvp1F = Kp1F * (SV1F - b1F)
Mvi1F = ((Kp1F * 0.5) / (2 * Ti1F)) * ((2 * SV1F) - (3 * b1F) + PV11F) + Mvi1F
Mvd1F = Kp1F * Td1F * (PV11F - b1F)
c1F = Mvp1F + Mvi1F + Mvd1F
 If c1F > 5 Then
  c1F = 5
```

```
  ElseIf c1F < 0 Then
  c1F = 0
End If
PV11F = b1F
PID1C = c1F
End Function
```

ส่วนการรับส่งข้อมูลก็อาศัยหลักการดังในหัวข้อ 3.2.7 การรับข้อมูลอินพุทจะใช้ช่องรับอินพุท 2 ช่องโดยในโปรแกรมจะกำหนดให้ช่องที่ 0 ใช้ในการวัดระดับ และช่องที่ 1 ใช้ในการวัดอัตราการไหล

จะกำหนดข้อมูลควบคุมเป็น 44 H คือมีการกำหนดให้วนรับค่าแบบอัตโนมัติ

บทที่ 4

การทดลอง

จากหลักและทฤษฎีในบทที่ 2 และการสร้างโปรแกรมในบทที่ 3 เราจะนำมาใช้ในการทดลอง การทดลองต่างๆ จะช่วยให้เข้าใจ และได้ให้เห็นผลจริงว่า โครงงานนี้สามารถทำได้ตามวัตถุประสงค์

ในบทนี้จะมีการทดลองที่เรียงลำดับขึ้นของการพัฒนาโครงการดังนี้

1. การทดลองการใช้งานไอซี PCF 8591
2. การทดลองตัวแปลงแรงดันและตัวแปลงแรงดันเป็นกระแส
3. การทดลองโปรแกรมกระบวนการควบคุมระดับน้ำ
4. การทดลองโปรแกรมกระบวนการ 4 กระบวนการ
5. การทดลองโปรแกรมควบคุมกระบวนการ 4 กระบวนการโดยมีการควบคุมแบบ Cascade

ในเรื่องแรก IC PCF 8591 เป็นไอซีที่ทำหน้าที่แปลงสัญญาณอะนาลอก(Analog)เป็นสัญญาณดิจิตอล (Digital) และแปลงสัญญาณดิจิตอลเป็นสัญญาณอะนาลอก เพราะคอมพิวเตอร์ (Computer) จะใช้สัญญาณดิจิตอลและจะต้องมีการติดต่อกับอุปกรณ์วัดและอุปกรณ์ควบคุมตัวสุดท้ายที่ใช้สัญญาณอะนาลอก จึงต้องมีตัวแปลงสัญญาณต่างๆ

การทดลองที่ 1

<u>การใช้งานไอซี พีซีเอฟ (IC PCF 8591)</u>

ประกอบด้วย

การทดลองที่ 1.1 การทดลอง IC PCF 8591 วัดระดับแรงดันที่ตัวมันเองส่งออกมา

การทดลองที่ 1.2 การทดลอง IC PCF 8591 ใช้วัดแรงดันจากแหล่งจ่าย

การทดลองที่ 1.3 การทดลอง IC PCF 8591 ใช้มิเตอร์วัดแรงดันที่จ่ายจาก IC PCF 8591

การทดลองที่ 1.1

<u>การใช้ IC PCF 8591 วัดแรงดันที่ตัวมันเองส่งออกมา</u>

<u>วัตถุประสงค์</u>

1. เพื่อดูว่า IC PCF 8591 สามารถรับแรงดันได้หรือไม่
2 เพื่อดูว่า IC PCF 8591 สามารถส่งแรงดันได้หรือไม่

อุปกรณ์

1. บอร์ด ST-29 Serail port interfacing starter board
2. คอมพิวเตอร์ PC ที่มีพอร์ทอนุกรมอย่างน้อย 1 พอร์ท
3. สายต่อวงจร

ขั้นตอนการทดลอง

1. ที่บอร์ด ST-29 ให้เลือกจั๊มเปอร์ที่ใช้กำหนดแอดเดรสของ PCF8591 ไว้ที่ตำแหน่ง 000 (A0,A1,A2 ต่อลงกราวด์ทั้งหมด)
2. ต่อบอร์ด ST-29 กับ Computer
3. เปิดโปรแกรม Visual Basic เรียกโปรแกรม PCF8591.VBP รันให้ทำงาน
4. ต่ออินพุทอะนาลอกทั้ง 3 ช่องลงกราวด์ โดยที่ CH4 ต่อกับเอาท์พุทในกรอบ PC-based Voltage Source กำหนดค่า Voltage ในช่องเท็กบ็อกให้มีค่า 0-5V บันทึกค่าที่อ่านได้จากช่อง PC-based Analog To Digital Converter บนหน้าจอมอนิเตอร์

ผลการทดลอง

Output ที่กำหนดจากคอมพิวเตอร์	Input ที่แสดงบนหน้าจอคอมพิวเตอร์ วัดจาก CH4
1	0.98
1.5	1.47
2	1.98
2.5	2.49
3	2.98
3.5	3.47
4	3.98
4.5	4.49
5	4.98

ตารางที่ 4.1 ตารางแสดงผลการทดลอง การใช้ IC PCF 8591 วัดแรงดันที่ตัวมันเองส่งออกมา

การทดลองที่ 1.2

การใช้ IC PCF 8591วัดแรงดันจากแหล่งจ่าย

วัตถุประสงค์

เพื่อดูความเที่ยงตรงในการวัดค่าของ IC PCF 8591

อุปกรณ์

1. บอร์ด ST-29 Serail port interfacing starter board
2. คอมพิวเตอร์ PC ที่มีพอร์ทอนุกรมอย่างน้อย 1 พอร์ท
3. สายต่อวงจร
4. แหล่งจ่ายไฟแรงดันตรง

ขั้นตอนการทดลอง

1. ต่อบอร์ด ST-29 กับ Computer
2. ต่อสายไฟบวกของแหล่งจ่ายกับช่อง CH1 ไฟลบหรือกราวด์ต่อกับช่อง GND ส่วนช่องอื่นต่อลงกราวด์
3. เปิดโปรแกรม Visual Basic เรียกโปรแกรม PCF 8591.VBF รันให้ทำงาน
4. จ่ายไฟจากแหล่งจ่ายแรงดันตรง 0-5 โวลต์ แล้วบันทึกผล

ผลการทดลอง

Output จากแหล่งจ่ายภายนอก	Input ที่แสดงบนหน้าจอ
0.51	0.509
0.90	0.921
0.99	1.0
1.49	1.50
1.50	1.52
1.95	1.95
2.06	2.07
2.35	2.37

2.50	2.52
3.01	3.0
3.45	3.49
3.47	3.5
3.96	4.0
4.52	4.56
4.45	4.49
4.87	4.92
5.09	5.0

ตารางที่ 4.2 แสดงผลการใช้ IC PCF 8591วัดแรงดันจากแหล่งจ่าย

การทดลองที่ 1-3

<u>การใช้ IC PCF 8591 จ่ายแรงดันแล้วใช้มิเตอร์วัด</u>

<u>วัตถุประสงค์</u>

เพื่อดูความเที่ยงตรงในการจ่ายแรงดันของ IC PCF 8591

<u>อุปกรณ์</u>

1. บอร์ด ST-29 Serail port interfacing starter board
2. คอมพิวเตอร์ PC ที่มีพอร์ทอนุกรมอย่างน้อย 1 พอร์ท
3. สายต่อวงจร
4. มิเตอร์วัดแรงดัน

<u>ขั้นตอนการทดลอง</u>

1. ที่บอร์ด ST-29 ให้เลือกจั๊มเปอร์ที่ใช้กำหนดแอดเดรสของ PCF8591 ไว้ที่ตำแหน่ง 000 (A0,A1,A2 ต่อลงกราวด์ทั้งหมด)
2. ต่อทุกช่องอินพุท(input)ลงกราวด์
3. เปิดโปรแกรม

4. เปลี่ยนค่าแรงดันภายในกรอบของ PC-base Voltage เป็นค่าต่างๆ ใช้มิเตอร์วัดแรงดัน บันทึกผล

<u>ผลการทดลอง</u>

Outputที่กำหนดจากคอมพิวเตอร์	ค่าที่วัดโดยมิเตอร์
1	0.97
1.5	1.45
2	1.96
2.5	2.47
3	2.965
3.5	3.43
4	3.93
4.5	4.45
5	4.93

ตารางที่ 4.3 แสดงผลการใช้ IC PCF 8591 จ่ายแรงดันแล้วใช้มิเตอร์วัด

เนื่องจากสัญญาณแรงดันมาตรฐานของอุปกรณ์วัดจะเป็น1-5 โวลต์ แต่ IC PCF8591 จะรับสัญญาณได้ 0-5 โวลต์ เพื่อให้ IC PCF8591 ถูกใช้อย่างเต็มที่ จึงมีตัวแปลงแรงดัน 1-5 โวลต์เป็น0-5 โวลต์ และเนื่องจากตัวอุปกรณ์ควบคุมตัวสุดท้ายใช้สัญญาณมาตรฐาน 4-20 มิลลิแอมป์ แต่ IC PCF8591 ให้เอาท์พุทเป็นแรงดัน 0-5โวลต์ ดังนั้นจึงมีตัวแปลงแรงดัน 0-5 โวลต์ให้เป็นกระแส 4-20 มิลลิแอมป์

การทดลองที่ 2

<u>การทดลองตัวแปลงแรงดันเป็นแรงดันและตัวแปลงแรงดันเป็นกระแส</u>

ประกอบด้วย

การทดลองที่ 2.1 ตัวแปลงแรงดันเป็นแรงดัน (V/V)

การทดลองที่ 2.2 ตัวแปลงแรงดันเป็นกระแส (V/I)

การทดลองที่ 2.1

การทดลองตัวแปลงแรงดันเป็นแรงดัน(V/V)

วัตถุประสงค์

เพื่อให้ตัวแปลงแรงดันเป็นแรงดัน สามารถแปลงแรงดัน 1-5 โวลต์เป็น 0-5 โวลต์ และสามารถรับ เพื่อให้สัญญาณออกมาถูกต้อง

อุปกรณ์

1. ตัวแปลงแรงดันเป็นแรงดัน
2. มิเตอร์
3. แหล่งจ่ายแรงดัน

ขั้นตอนการทดลอง

1. คำนวณค่าเอาท์พุทที่สัมพันธ์กับอินพุท
2. ต่อแหล่งจ่ายแรงดันที่ขาอินพุทของตัวแปลงแรงดันเป็นแรงดัน
3. ใช้มิเตอร์ต่อกับขาเอาท์พุทของตัวแปลงแรงดันเป็นแรงดัน
4. ทำการปรับ Zero Span ปรับ Zero โดย ให้แรงดันจากแหล่งจ่ายเป็น 1 โวลต์ แล้วปรับ Zero ให้ได้เอาท์พุทให้ได้ 0 โวลต์ ปรับ Span โดยให้แรงดันจากแหล่งจ่ายเป็น 5 โวลต์ แล้วปรับ Span ให้ได้แรงดัน Output เป็น 5โวลต์ ดูค่า Zero อีกครั้ง ถ้าไม่ได้ก็ปรับอีก แล้วดู Span ปรับ Zero Span จนได้ค่าที่ถูกต้อง

ผลการทดลอง

อินพุทที่ควรจะเป็น	อินพุทจริง	ค่าเอาท์พุทที่คำนวณ	เอาท์พุทจริง
1	1.010	0	0.024
1.5	1.502	0.625	0.591
2	2.004	1.25	1.255
2.5	2.496	1.87	1.872
3	2.999	2.5	2.504
3.5	3.506	3.125	3.140
4	3.998	3.75	3.755
4.5	4.48	4.375	4.36
5	5.01	5	5.03
4.5	4.48	4.375	4.36
4	3.99	3.75	3.751
3.5	3.505	3.125	3.140
3	3.012	2.5	2.521
2.5	2.499	1.87	1.878
2	2.007	1.25	1.260
1.5	1.498	0.625	0.616
1	0.988	0	0.024

ตารางที่ 4.4 แสดงผลการทดลองตัวแปลงแรงดันเป็นแรงดัน(V/V)

ค่าแรงดันเอาท์พุทที่สัมพันธ์กับอินพุทที่คำนวณได้

$5/4 \times 0.5 = 0.625$

การทดลองที่ 2.2

<u>การทดลองการแปลงแรงดันเป็นกระแส</u>

<u>วัตถุประสงค์</u>

เพื่อให้ตัวแปลงแรงดันเป็นกระแส สามารถแปลงแรงดัน 1-5 โวลต์เป็นกระแส 4-20 มิลลิแอมป์ และสามารถปรับ Zero Span เพื่อให้สัญญาณออกมาถูกต้อง

<u>อุปกรณ์</u>

1. ตัวแปลงแรงดันเป็นกระแส
2. มิเตอร์
3. ตัวต้านทาน 250 โอห์ม

<u>ขั้นตอนการทดลอง</u>

1. คำนวณเอาท์พุทที่สัมพันธ์กับอินพุท
2. ต่อแหล่งจ่ายแรงดันที่ขาอินพุทของตัวแปลงแรงดันเป็นกระแส
3. ต่อตัวต้านทาน 250 โอห์มอนุกรมกับขาเอาท์พุทของตัวแปลงแรงดันเป็นกระแส ใช้มิเตอร์วัดแรงดันคร่อมตัวต้านทาน
4. ทำการปรับ Zero และ Span การปรับ Zero โดยให้แรงดันจากแหล่งจ่ายเป็น 0 โวลต์ แล้วปรับค่า Zero ให้ได้แรงดันเอาท์พุท 1โวลต์ การปรับ Span โดยให้แรงดันจากแหล่งจ่าย 5 โวลต์ แล้วปรับค่า Span ให้ได้แรงดันเอาท์พุท 5 โวลต์ ทำการปรับ Zero และ Span อีก ถ้ายังไม่ได้ค่าตรงตามต้องการ
5. ทดลองปรับค่าแรงดันจากแหล่งจ่ายให้อยู่ระหว่าง 0-5 โวลต์ แล้วดูค่าเอาท์พุทบันทึกผล

<u>ผลการทดลอง</u>

ค่าแรงดันเอาท์พุทที่วัดคร่อม R 250Ω

ค่าแรงดันอินพุท(โวลต์)	ค่าแรงดันเอาท์พุทจากการคำนวณ(โวลต์)	ค่าแรงดันเอาท์พุท (โวลต์)
0	1	1.005
0.5	1.4	1.406
1	1.8	1.806
1.5	2.2	2.207
2	2.6	2.607
2.5	3	3.008
3	3.4	3.409
3.5	3.8	3.810
4	4.2	4.21
4.5	4.6	4.61
5	5	5.01

รูปที่ 4. 5 แสดงผลการทดลองตัวแปลงแรงดันเป็นกระแส (V/I)

การควบคุมกระบวนการ จะอาศัยโปรแกรมที่เขียนในบทที่ 3 มาใช้

การทดลองที่ 3

<u>การทดลองโปรแกรมกระบวนการควบคุมระดับน้ำ</u>

<u>วัตถุประสงค์</u>

1. เพื่อนำโปรแกรมกระบวนการควบคุมระดับน้ำไปใช้กับกระบวนการจริงได้
2. สามารถปรับค่า PID ที่เหมาะสม ให้กระบวนการมีผลตอบสนองที่ดีได้

<u>อุปกรณ์</u>

1. คอมพิวเตอร์ที่มีพอร์ทอนุกรม
2. บอร์ดแปลงสัญญาณ I^2C กับ RS-232
3. บอร์ด Input และ Output ที่ประกอบด้วย

- ตัวแปลงแรงดันเป็นแรงดัน
- ตัวแปลงแรงดันเป็นกระแส
- IC PCF8591
- แหล่งจ่ายไฟกระแสตรง

4. สายต่อ

ขั้นตอนการทดลอง

1. ต่อบอร์ดแปลงสัญญาณ I^2C กับ RS-232 กับคอมพิวเตอร์
2. ต่อสายสัญญาณ SDA และ SCL กับ IC PCF8591 ที่บอร์ด Input-Output และดูสวิทช์ที่กำหนดแอดเดรสต้องเป็น "0" "0" "0" เพราะโปรแกรมที่เขียนไว้ กำหนดแอดเดรสของกระบวนการไว้แบบนี้
3. ต่อสายสัญญาณแรงดันจากอุปกรณ์วัดกับช่องอินพุทของบอร์ด Input-Output ช่องที่ 1
4. สายต่อสัญญาณเอาท์พุทของบอร์ด Input-Output กับอินพุทของอุปกรณ์ควบคุมสุดท้ายคือวาล์วควบคุม
5. เปิดสวิทช์ของกระบวนการควบคุมระดับ และเปิดวาล์วจ่ายลมให้กับวาล์วควบคุม
6. เปิดโปรแกรม ทำการรัน Run
7. เลือกว่าจะควบคุม Auto หรือ Manual ถ้าเลือก Auto ทำการเลือกค่า SV,K_p,Ti,Td ที่ต้องการแล้วกดปุ่ม START ดูผลจากกราฟ

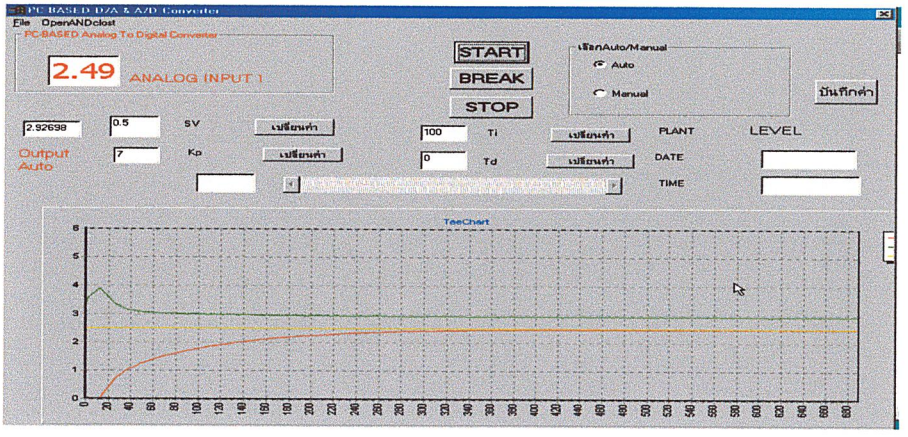
   ถ้าเลือก Manual ก็กดปุ่ม START ได้เลย สามารถควบคุมการเปิดปิดวาล์ว ได้โดยการเลื่อน Score bar ดูผลได้จากกราฟ

   สามารถเปลี่ยนโหมดการควบคุมได้ โดยกดปุ่ม "Break" หรือ "Stop" แล้วเลือก Auto Manual แล้วกด "START" ขณะที่กำลังทำงานอยู่ในโหมด Auto สามารถที่จะเปลี่ยนค่า SV,K_p ,Ti และ Td ได้โดย ใส่ค่าในช่องว่าง แล้วกดปุ่ม "เปลี่ยนค่า"
8. ดูผลจากกราฟว่ากระบวนการเข้าสู่สภาวะคงที่หรือไม่ ใช้เวลาเท่าไร โดยกราฟแกนตั้งจะเป็นร้อยละของสัญญาณ SV,PV และ MV แกนนอนจะเป็นเวลาหน่วยเป็นวินาที แต่ละเส้นของกราฟจะมีสีต่างกันและแทนค่าต่างกันโดยดูได้จากกรอบบนของกราฟ
9. ถ้าต้องการบันทึกผลเก็บไว้ ให้กดปุ่ม "RECORD"

10. สามารถดูผลที่บันทึกได้โดยเปิด FORM Show Data ใส่ชื่อ PLANT วันที่และเวลาที่บันทึก แล้วกดปุ่ม "แสดงค่า"

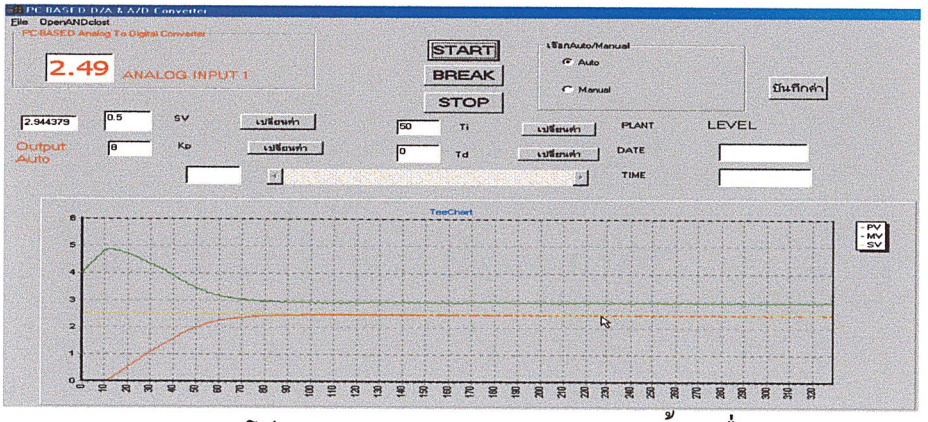
<u>ผลการทดลอง</u>

เมื่อ SV = 0.5 K_p = 7 Ti =100 Td = 0 ได้ผลดังรูป



รูปที่ 4.1 แสดงผลการทดลองโปรแกรมกระบวนการควบคุมระดับน้ำ เมื่อ SV = 0.5 K_p = 7 Ti =100 Td = 0

จะเห็นว่าระบบเข้าสู่สภาวะคงที่ ใช้เวลานาน ต้องทำการปรับค่า PID โดยดูจากสูตรของสมการ PID ถ้าค่า K เพิ่มขึ้น จะทำให้ได้ MV เพิ่มมากขึ้น และถ้าลดค่า Ti ลง จะทำให้ได้ MV เพิ่มมากขึ้น ดังนั้นทดลอง ให้ SV = 0.5, K_p = 8, Ti = 50, Td = 0 ได้ผลดังนี้



รูปที่ 4.2 แสดงผลการทดลองโปรแกรมกระบวนการควบคุมระดับน้ำ เมื่อ SV = 0.5 K_p = 7 Ti =50, Td = 0

จะเห็นว่าระบบเข้าสู่สภาวะคงที่เร็วขึ้น การปรับค่า PID ด้วยวิธีนี้ เรียกว่า ลองผิดลองถูกอย่างมีหลักการ

การควบคุม 1 กระบวนการสามารถทำได้แล้ว ขั้นต่อไปจะทำการควบคุมหลายกระบวนการพร้อมกัน การควบคุมแบบ DCS (Distributed Control Systems) คือระบบควบคุมแบบกระจายส่วนในระบบจะประกอบด้วย โมดูลอินพุทและเอาท์พุทหลายโมดูลตามจำนวนกระบวนการที่ควบคุม ส่วนการควบคุมแบบ ICS (Integrated Control Systems) จะเป็นระบบควบคุมแบบรวมส่วน จะประกอบด้วยโมดูลอินพุทและเอาท์พุทเพียง 1 โมดูล อาศัยการวนรับค่า และส่งค่าระหว่างกระบวนการต่างๆกับตัวควบคุม ต่อไปเราจะทำการทดลองควบคุมกระบวนการหลายกระบวนการแบบ ICS

การทดลองที่ 4

การควบคุมกระบวนการแบบ ICS

<u>วัตถุประสงค์</u>

เพื่อต้องการดูว่าโปรแกรมที่ทำสามารถควบคุมกระบวนการแบบ ICS ได้หรือไม่

<u>วัสดุอุปกรณ์</u>

1. Plant ควบคุมอัตราการไหล
2. Plantควบคุมระดับ
3. คอมพิวเตอร์ที่มีพอร์ทอนุกรม
4. บอร์ดแปลงสัญญาณ I^2C เป็น RS-232
5. บอร์ด Input/Output ที่ประกอบด้วย
   -ตัวแปลงแรงดันเป็นแรงดัน
   -ตัวแปลงแรงดันเป็นกระแส
   -แหล่งจ่ายไฟกระแสตรง
   -IC PCF8591
6. สายต่อ

<u>ขั้นตอนการทดลอง</u>

โปรแกรมที่สร้างไว้นั้นสามารถควบคุมกระบวนการได้ 4 กระบวนการคือ

1) กระบวนการควบคุมระดับ
2) กระบวนการควบคุมอัตราการไหล
3) กระบวนการควบคุมความดัน
4) กระบวนการควบคุมอุณหภูมิ

แต่ในการทดลองนี้จะทำการควบคุมเพียง 2 กระบวนการคือ กระบวนการควบคุมระดับ, กระบวนการควบคุมอัตราการไหล เพื่อเป็นตัวอย่างและแสดงให้เห็นว่า โปรแกรมนี้สามารถควบคุมกระบวนการได้มากกว่า 1 กระบวนการ

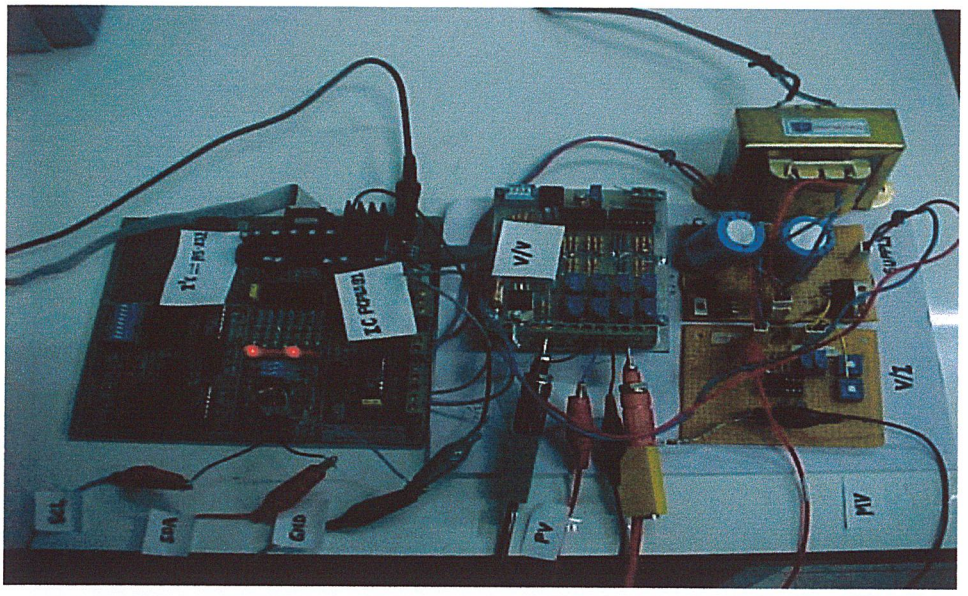
1. ต่อสายสัญญาณ SDA และ SCL กับ IC PCF8591 ของบอร์ด Input/Output ทั้ง 2 กระบวนการโดยสวิตซ์กำหนดแอดเดรสของ IC PCF8591 ที่กระบวนการควบคุมระดับจะกำหนดเป็น 000 ที่กระบวนการควบคุมอัตราการไหลจะกำหนดเป็น 001
2. ต่อสายสัญญาณแรงดันจากอุปกรณ์วัดกับช่องอินพุทช่องที่ 0 ของบอร์ด Input/Output ในแต่ละกระบวนการ
3. ต่อสายสัญญาณเอาท์พุทของบอร์ด Input/Output กับอินพุทของอุปกรณ์ควบคุมสุดท้ายคือวาล์วควบคุมของแต่ละกระบวนการ
4. เปิดสวิซต์ของกระบวนการควบคุมและเปิดวาล์วจ่ายลมให้แก่วาล์วควบคุมของแต่ละกระบวนการ
5. เปิดโปรแกรม ProcessControl02 ทำการรัน
6. ทำการควบคุมกระบวนการควบคุมระดับหรืออัตราการไหล โดยปฏิบัติดังนี้

   -ทำการเลือก Auto หรือ Manual ถ้าเลือก Auto ให้ทำการกำหนดค่า SV,Kp,Ti และ Td แล้วกดปุ่ม START ถ้าทำการเลือก Manual สามารถกดปุ่ม START ได้เลย แล้วสมารถกำหนด MV โดยการเลื่อน Scroll bar

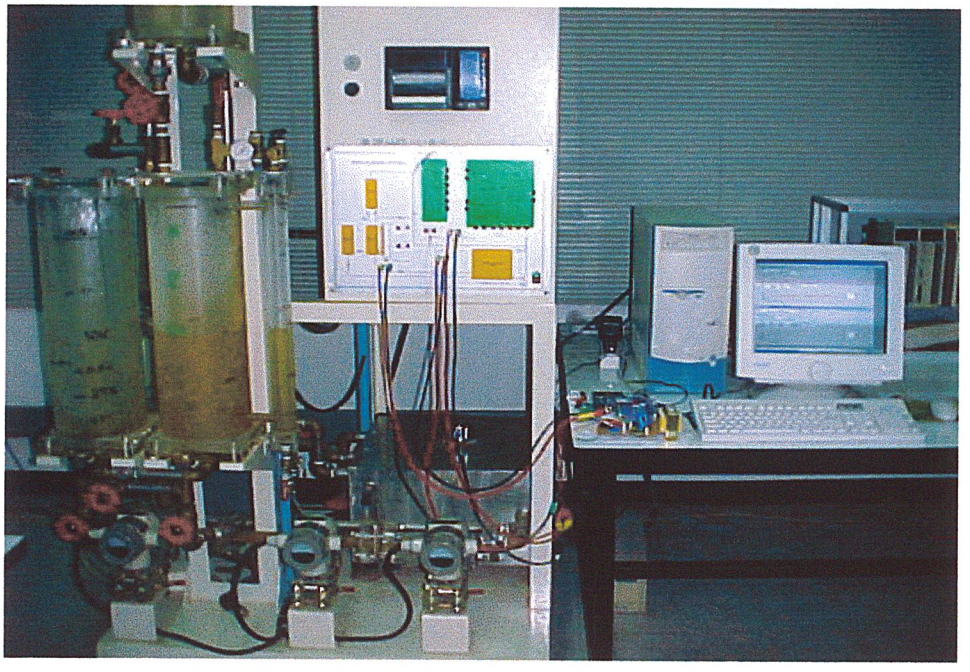
   -ในโหมด Auto ขณะที่โปรแกรมทำงาน สามารถเปลี่ยนค่า SV,Kp,Ti และ Td โดยใส่ค่าในช่องว่างและกดปุ่ม CHANGE

   -สามารถทำการบันทึกกราฟผลการทดลองได้โดยกด ปุ่ม RECORD

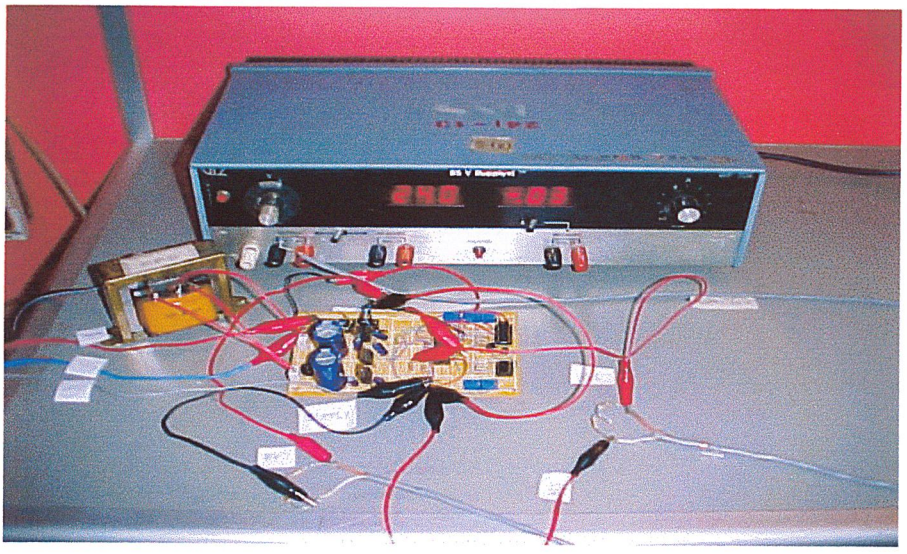
   -ถ้าต้องการดูข้อมูลที่บันทึกไว้ทำได้โดยไปที่เมนู OPEN แล้วเลือก OpenONE แล้วเลือก OpenShowData ทำการเลือกชื่อ Plant แล้วกดปุ่ม OK จากนั้นเลือก DATE แล้วกดปุ่ม OK เมื่อเลือกวันที่ได้แล้วก็ทำการเลือกเวลา Time จากนั้นกดปุ่ม ShowData จะปรากฏกราฟผลการทดลองที่บันทึกไว้



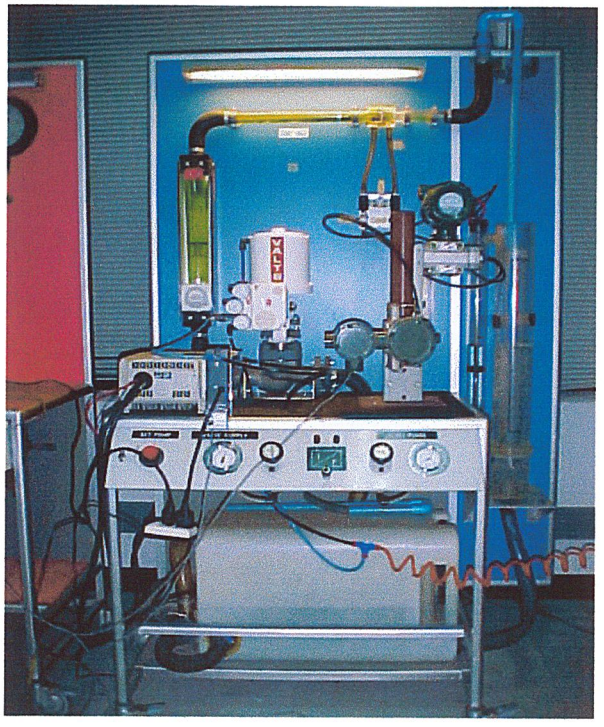
รูปที่ 4.3 แสดงการต่อวงจรควบคุมกระบวนการควบคุมอัตราการไหล



รูปที่ 4.4 แสดงการควบคุมกระบวนการควบคุมอัตราการไหล



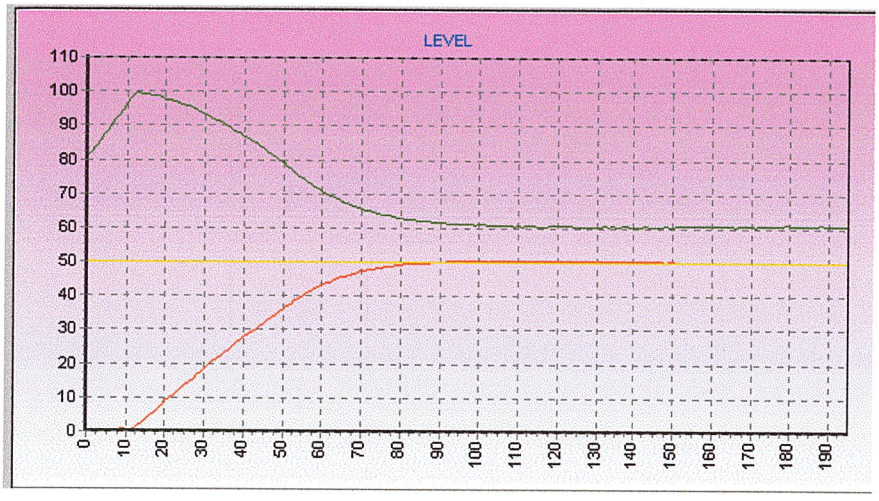
รูปที่ 4.5 แสดงการต่อวงจรควบคุมกระบวนการควบคุมระดับ



รูปที่ 4.6 แสดงการควบคุมกระบวนการควบคุมระดับ

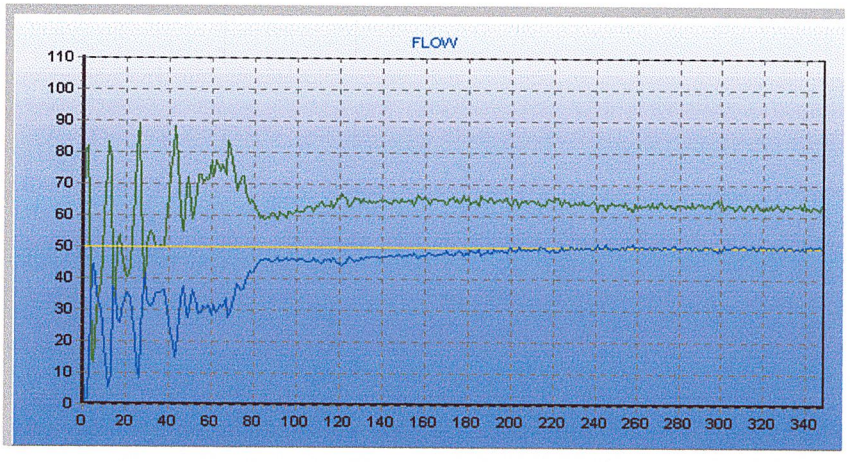
<u>ผลการทดลอง</u>

ผลการทดลองของกระบวนการควบคุมระดับ



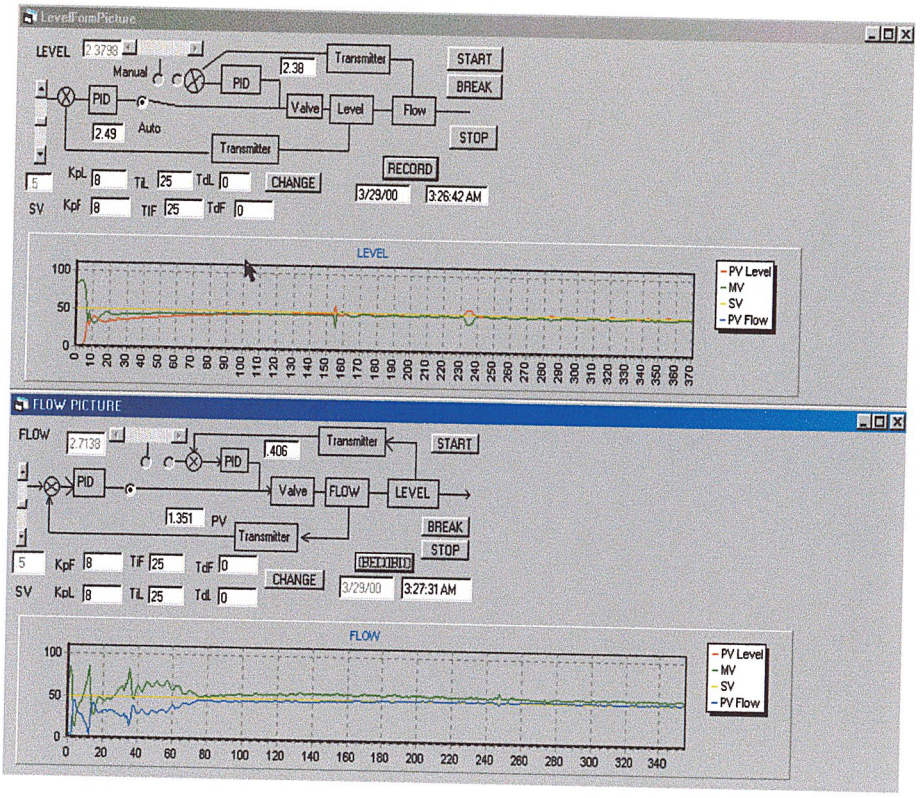
รูปที่ 4.7 แสดงผลการควบคุมกระบวนการควบคุมระดับ SV=50%, Kp=8,Ti=25,Td=0 โดยเส้นสีเขียวแสดง MV, เส้นสีเหลืองแสดง SV, เส้นสีแดงแสดง PV

ผลการทดลองของกระบวนการควบคุมอัตราการไหล



รูปที่ 4.8 แสดงผลการควบคุมกระบวนการควบคุมอัตราการไหล โดยที่ SV=50%, Kp=8,Ti=25,Td=0 โดยเส้นสีเขียวแสดง MV, เส้นสีเหลืองแสดง SV, เส้นสีน้ำเงินแสดง PV

ผลการทดลองของกระบวนการควบคุม 2 กระบวนการพร้อมกัน



รูปที่ 4.9 แสดงผลการทดลองของกระบวนการควบคุม 2 กระบวนการพร้อมกัน

การควบคุมกระบวนการแบบ Cascade ของกระบวนการควบคุมระดับและอัตราควบคุมอัตราการไหล จะแบ่งเป็น

1. การควบคุมระดับ จะกำหนด SV ของระดับ แล้วนำค่า SV ไปคำนวณกับค่า PV ของกระบวนการควบคุมระดับ ในสมการ PID ได้ผลลัพธ์ออกมานำผลลัพธ์นี้เป็นค่า SV ของกระบวนการควบคุมอัตราการไหล แล้วนำค่าSV นี้กับค่า PV ของอัตราการไหล มาคำนวณในสมการ PID จะได้ผลลัพธ์ เป็นค่า MV ไปควบคุมวาล์วควบคุม

2. . การควบคุมอัตราการไหล จะกำหนด SV ของอัตราการไหล แล้วนำค่า SV ไปคำนวณกับค่า PV ของกระบวนการควบคุมอัตราการไหล ในสมการ PID ได้ผลลัพธ์ออกมานำผลลัพธ์นี้เป็นค่า SV ของ

กระบวนการควบคุมระดับ แล้วนำค่า SV นี้กับค่า PV ของระดับมาคำนวณในสมการ PID จะได้ผลลัพธ์ เป็นค่า MV ไปควบคุมวาล์วควบคุม

การทดลองที่ 5

<u>การควบคุมกระบวนการแบบ Cascade</u>

<u>วัตถุประสงค์</u>

เพื่อต้องการควบคุมกระบวนการแบบ Cascade และทดสอบโปรแกรมที่เขียนไว้ว่าสามารถควบคุมกระบวนการแบบ Cascade ได้หรือไม่

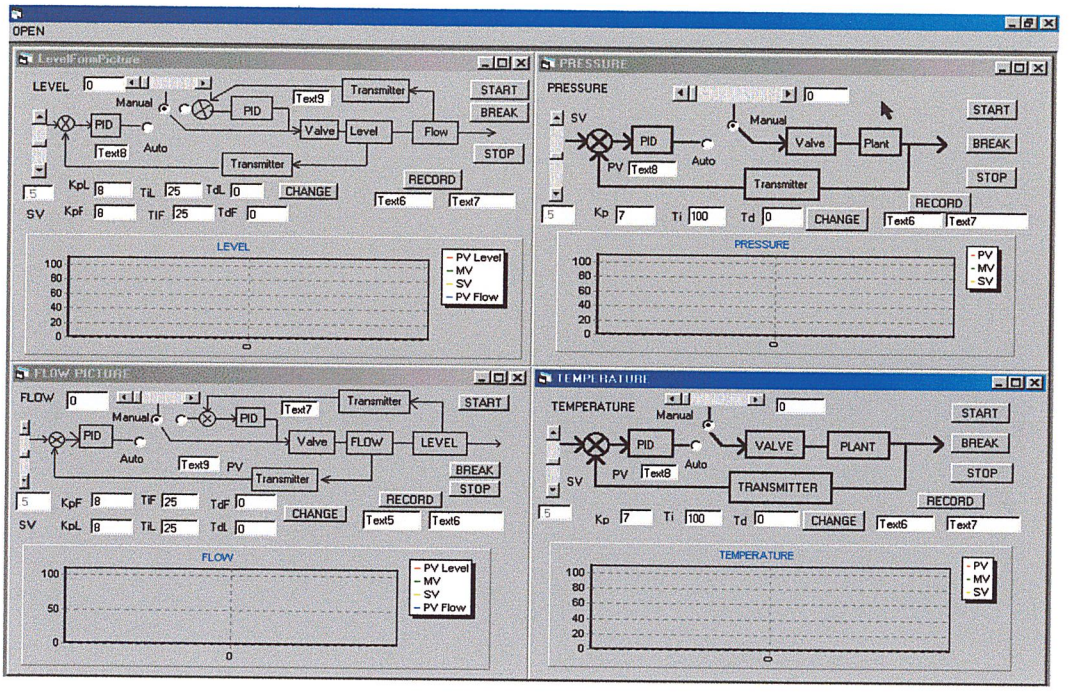
<u>อุปกรณ์</u>

1. Plant ที่มีกระบวนการควบคุมระดับและอัตราการไหลใน Plant เดียวกัน
2. คอมพิวเตอร์ที่มีพอร์ทอนุกรม
3. บอร์ดแปลงสัญญาณ I^2C เป็น RS232
4. บอร์ด Input/Output ที่ประกอบด้วย
   -ตัวแปลงแรงดันเป็นแรงดัน
   -ตัวแปลงแรงดันเป็นกระแส
   -แหล่งจ่ายไฟกระแสตรง
   -IC PCF8591
5. สายต่อ

<u>ขั้นตอนการทดลอง</u>

1. ต่อบอร์ดแปลงสัญญาณ I^2C เป็น RS232 กับคอมพิวเตอร์ ต่อที่พอร์ท Com2
2. ต่อสายสัญญาณ SDA และ SCL กับ IC PCF8591 ของบอร์ด Input/Output ทั้ง 2 กระบวนการที่อยู่ในPlant เดียวกันโดยสวิทช์กำหนดแอดเดรสของ IC PCF8591 ที่กระบวนการควบคุมระดับจะกำหนดเป็น 000 ที่กระบวนการควบคุมอัตราการไหลจะกำหนดเป็น 001
3. ต่อสายสัญญาณแรงดันจากอุปกรณ์วัดระดับกับช่องอินพุทช่องที่ 0 และต่อสายสัญญาณแรงดันจากสายอุปกรณ์วัดอัตราการไหลกับอินพุทช่องที่ 1 ของบอร์ด Input/Output ในแต่ละกระบวนการ
4. ต่อสายสัญญาณเอาท์พุทของบอร์ด Input/Output กับอินพุทของอุปกรณ์ควบคุมสุดท้ายคือวาล์วควบคุมของแต่ละกระบวนการ
5. เปิดสวิทช์ของกระบวนการควบคุมและเปิดวาล์วจ่ายลมให้แก่วาล์วควบคุมของ Plant

6. เปิดโปรแกรม ProcessControl08 ทำการรัน จะปรากฏหน้าจอดังรูป



รูปที่ 4.10 แสดงหน้าจอเมื่อเริ่มรันโปรแกรมควบคุม 4 กระบวนการ

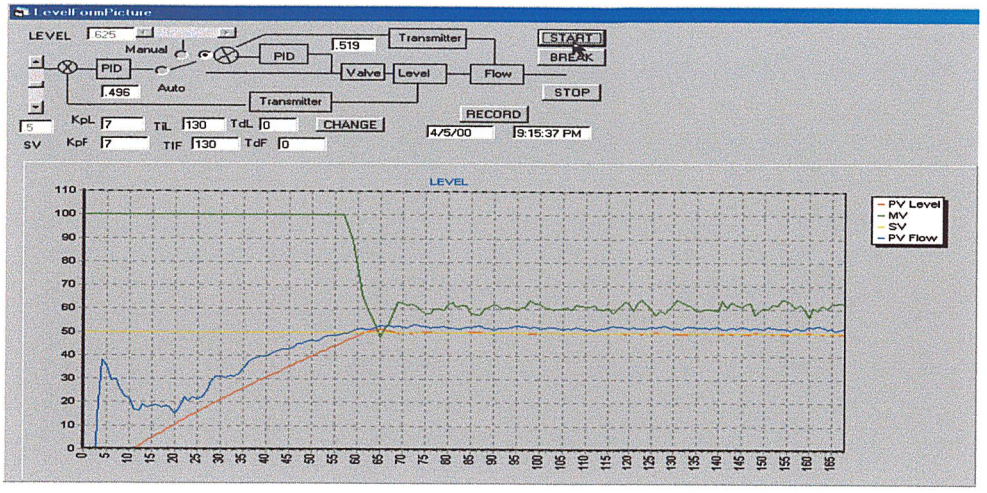
7. ทำการเลือกว่าจะควบคุมระดับให้ไปที่ LEVEL Form แล้วเลือก Cascade จากนั้นกำหนดค่า SV,Kp,Ti และTd จากนั้นกดปุ่ม START ดูผลจากกราฟ

8. ทำการเลือกว่าจะควบคุมอัตราการไหลให้ไปที่ Flow- Form แล้วเลือก Cascade จากนั้นกำหนดค่า SV,Kp,Ti และTd จากนั้นกดปุ่ม START ดูผลจากกราฟ

<u>ผลการทดลอง</u>

การควบคุมระดับแบบ Cascade

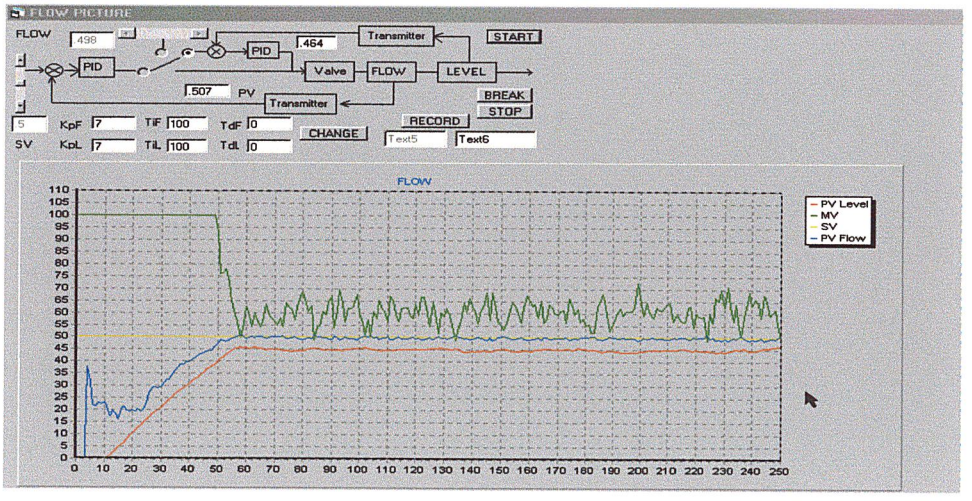
โดยที่ SV=0.5 ,KpL=7,TiL=100,TdL=0,KpF=7,TiF=100,TdF=0



รูปที่ 4.11 แสดงผลการควบคุมระดับแบบ Cascade

โดยที่ SV=0.5 ,KpL=7,TiL=130,TdL=5,KpF=7,TiF=130,TdF=0

การควบคุมอัตราการไหลแบบ Cascade



รูปที่ 4.12 แสดงผลการควบคุมอัตราการไหลแบบ Cascade

โดยที่ SV=0.5 ,KpL=7,TiL=100,TdL=5,KpF=7,TiF=100,TdF=0

สรุปผลการทดลอง

เราสามารถที่จะทำการติดต่อระหว่างคอมพิวเตอร์กับกระบวนการได้ โดยใช้มาตรฐานการเชื่อมต่อแบบ I^2C เพราะสามารถที่จะรับและส่งค่าข้อมูลระหว่างอุปกรณ์ภายนอกกับคอมพิวเตอร์ได้และค่าแรงดันมีความคลาดเคลื่อนเพียงเล็กน้อย เนื่องจากสัญญาณแรงดันมาตรฐานของอุปกรณ์วัดจะเป็น1-5 โวลต์ แต่ IC PCF8591 จะรับสัญญาณได้ 0-5 โวลต์ เพื่อให้ IC PCF8591 ถูกใช้อย่างเต็มที่ จึงมีตัวแปลงแรงดัน 1-5 โวลต์เป็น0-5 โวลต์และเนื่องจากตัวอุปกรณ์ควบคุมตัวสุดท้ายใช้สัญญาณมาตรฐาน 4-20 มิลลิแอมป์ แต่ IC PCF8591 ให้เอาท์พุทเป็นแรงดัน 0-5โวลต์ ดังนั้นสามารถใช้ตัวแปลงแรงดันแปลงแรงดัน 0-5 โวลต์ให้เป็นกระแส 4-20 มิลลิแอมป์จากโปรแกรมที่เขียนเราสามารถควบคุมกระบวนการหลายกระบวนการแบบ ICS ได้ นอกจากนี้ยังสามารถควบคุมแบบ Cascade ได้ด้วย

สรุปหลักการทำงานของการควบคุมกระบวนการ

เมื่อทำการรันโปรแกรมจะปรากฏหน้าจอ 4 กระบวนการทำการเลือกกระบวนการที่จะทำการควบคุม ในที่นี้เราจะเลือกการควบคุมระดับจากนั้นทำการเลือกว่าจะควบคุมแบบ Auto , Manual หรือ Cascade โดยการคลิกปุ่ม option สมมติทำการเลือกแบบ Auto จะทำให้ Option1.Value มีค่าเป็น True จากนั้นทำการกำหนดค่า SV , Kp , Ti และ Td กดปุ่ม START เพื่อเริ่มการทำงาน เมื่อกดปุ่ม START จะเป็นการกำหนดตัวแปร LEVSTART มีค่าเป็น True ทำการเปิดพอร์ทอนุกรมและกำหนดให้ Timer เริ่มทำงานโดยจะหน่วงเวลา 1 วินาทีเมื่อ timer ทำงานจะมีการตรวจเช็คค่าว่ากระบวนการใดมีการกด Start และมีการเลือกการทำงานแบบ Auto หรือManual ก็จะทำการรับค่าอินพุทและส่งค่าเอาท์พุทของกระบวนการนั้นๆ จากตัวอย่างค่า LevelForm.LEVSTART And LevelForm.Option1.Value มีค่าเป็น True แล้วจะเริ่มทำงานตามโปรแกรมดังนี้

```
Private Sub Timer1_Timer()
If LevelForm.LEVSTART And LevelForm.Option1.Value Then
If MSComm1.PortOpen = False Then
MSComm1.PortOpen = True   'เปิดพอร์ท
End If
LevelForm.ReciveInput1    ' รับค่าอินพุท(ค่า PV)
LevelForm.SendOutputAuto1   ' ส่งค่าเอาท์พุท(ค่า MV)
End If
```

ขั้นตอนการรับค่าอินพุทของโปรแกรมย่อย ReciveInput1จะเริ่มจากทรานสมิตเตอร์ส่งสัญญาณแรงดัน 1-5 โวลต์ผ่านตัว V/V ได้แรงดัน 0-5 โวลต์เข้าช่องรับอินพุทที่ 0 (CH0) ของ PCF 8591 ซึ่งได้กำหนดแอดเดรสจากดิปสวิตซ์เป็น 000 เพราะโปรแกรม ReciveInput1 กำหนดให้ติดต่อกับPCF 8591 ที่กำหนดแอดเดรสเป็น 000 จากนั้น PCF 8591จะทำการแปลงสัญญาณจากสัญญาณอะนาลอกเป็นสัญญาณดิจิตอลแบบ I^2C สัญญาณ I^2C จะถูกแปลงเป็นสัญญาณมาตรฐาน RS-232 เข้าสู่คอมพิวเตอร์

ขั้นตอนการส่งค่าเอาท์พุทของโปรแกรมย่อย SendOutputAuto1 จะเริ่มจากคอมพิวเตอร์นำค่าอินพุทที่รับได้มาคำนวณในสมการ PID แล้วทำการส่งค่าเอาท์พุทเป็นสัญญาณมาตรฐานเป็น RS-232 จากนั้นสัญญาณจะถูกแปลงเป็นสัญญาณ I^2C ส่งไปที่ PCF 8591 ซึ่งได้กำหนดแอดเดรสจากดิปสวิตซ์เป็น 000 เพราะโปรแกรม SendOutputAuto1 กำหนดให้ติดต่อกับPCF 8591 ที่กำหนดแอดเดรสเป็น 000 จากนั้น PCF 8591จะทำการแปลงสัญญาณจากสัญญาณดิจิตอลเป็นสัญญาณอะนาลอก 0-5 โวลต์แล้วผ่าน V/I ได้สัญญาณกระแสเป็น 4-20 มิลลิแอมป์ไปควบคุมอุปกรณ์ควบคุมสุดท้าย (วาล์ว)

ถ้ามีการกด START ของกระบวนการอื่นตัว Timer จะมีการเช็คค่าและทำการรับค่าส่งค่าเป็นลำดับ โดยโปรแกรมย่อย ReciveInput และ SendOutput ของแต่ละกระบวนการจะมีการติดต่อกับ PCF 8591 ที่กำหนดแอดเดรสแตกต่างกัน

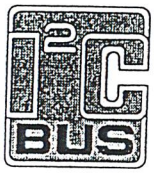
<u>ข้อเสนอแนะ</u>

โปรแกรมนี้เป็นโปรแกรมอย่างง่าย ใช้ทดลองเพื่อให้สามารถใช้งานได้จริง แต่ในการนำไปประยุกต์ใช้งานจริงยังต้องมีการปรับปรุงโปรแกรม เช่นในเรื่องความปลอดภัย และการบันทึกค่า อาจจะเพิ่มข้อมูลให้มากกว่านี้ การแสดงผลของกระบวนการอาจจะแสดงผลในรูปแบบอื่นก็ได้

ภาคผนวก

แสดง Data sheet ของ IC PCF8591

DATA SHEET



PCF8591
8-bit A/D and D/A converter

Product specification
Supersedes data of 1997 Apr 02
File under Integrated Circuits, IC12

1998 Jul 02

Philips
Semiconductors

PHILIPS

CONTENTS

1 FEATURES

- Single power supply
- Operating supply voltage 2.5 V to 6 V
- Low standby current
- Serial input/output via I²C-bus
- Address by 3 hardware address pins
- Sampling rate given by I²C-bus speed
- 4 analog inputs programmable as single-ended or differential inputs
- Auto-incremented channel selection
- Analog voltage range from V_{SS} to V_{DD}
- On-chip track and hold circuit
- 8-bit successive approximation A/D conversion
- Multiplying DAC with one analog output.

2 APPLICATIONS

- Closed loop control systems
- Low power converter for remote data acquisition
- Battery operated equipment
- Acquisition of analog values in automotive, audio and TV applications.

3 GENERAL DESCRIPTION

The PCF8591 is a single-chip, single-supply low power 8-bit CMOS data acquisition device with four analog inputs, one analog output and a serial I²C-bus interface. Three address pins A0, A1 and A2 are used for programming the hardware address, allowing the use of up to eight devices connected to the I²C-bus without additional hardware. Address, control and data to and from the device are transferred serially via the two-line bidirectional I²C-bus.

The functions of the device include analog input multiplexing, on-chip track and hold function, 8-bit analog-to-digital conversion and an 8-bit digital-to-analog conversion. The maximum conversion rate is given by the maximum speed of the I²C-bus.

4 ORDERING INFORMATION

TYPE NUMBER	PACKAGE		
	NAME	DESCRIPTION	VERSION
PCA8591P	DIP16	plastic dual in-line package; 16 leads (300 mil); long body	SOT38-1
PCA8591T	SO16	plastic small outline package; 16 leads; body width 7.5 mm	SOT162-1

5 BLOCK DIAGRAM

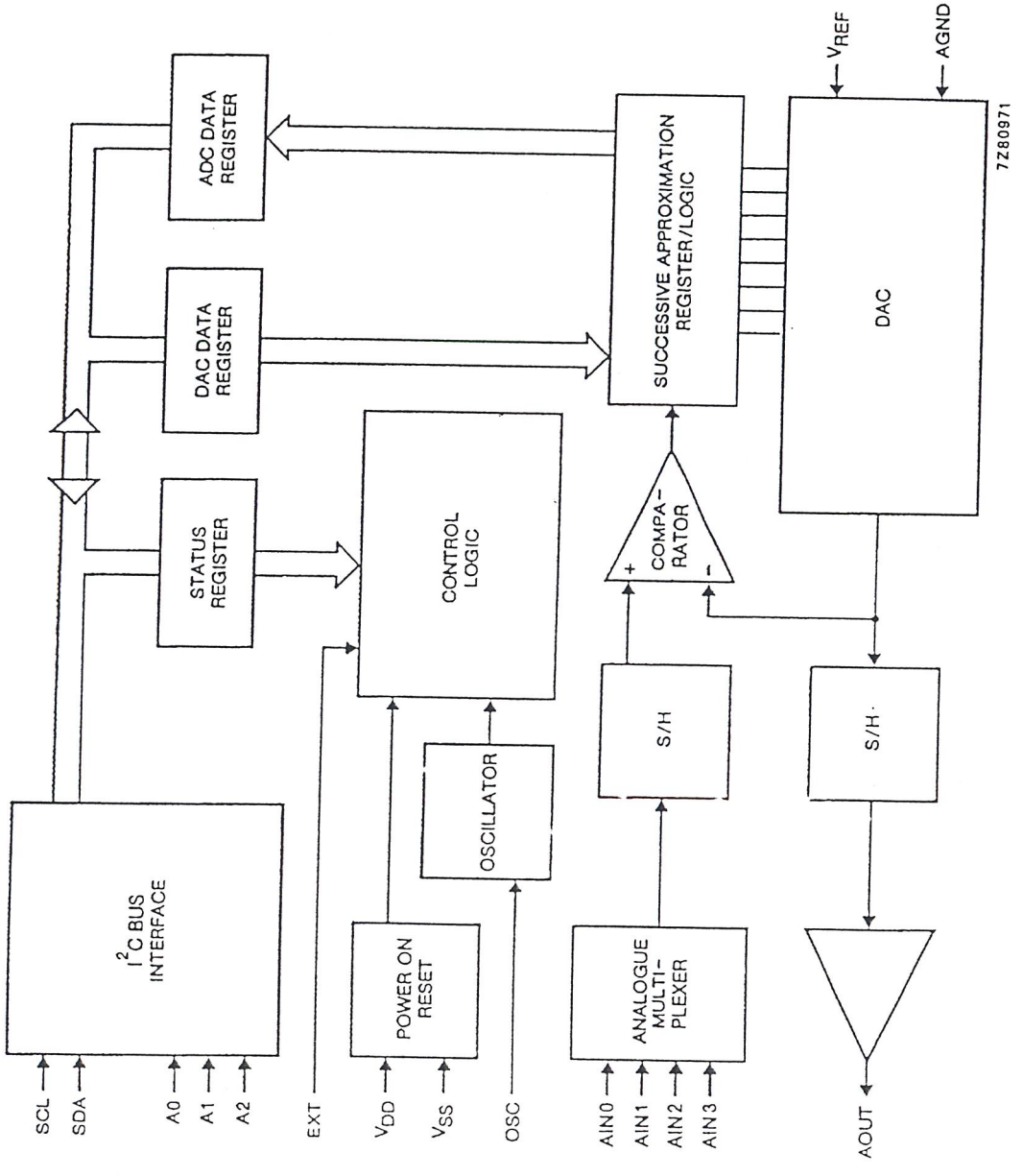


Fig.1 Block diagram.

6 PINNING

SYMBOL	PIN	DESCRIPTION
AIN0	1	analog inputs (A/D converter)
AIN1	2	
AIN2	3	
AIN3	4	
A0	5	hardware address
A1	6	
A2	7	
V_{SS}	8	negative supply voltage
SDA	9	I^2C-bus data input/output
SCL	10	I^2C-bus clock input
OSC	11	oscillator input/output
EXT	12	external/internal switch for oscillator input
AGND	13	analog ground
V_{REF}	14	voltage reference input
AOUT	15	analog output (D/A converter)
V_{DD}	16	positive supply voltage

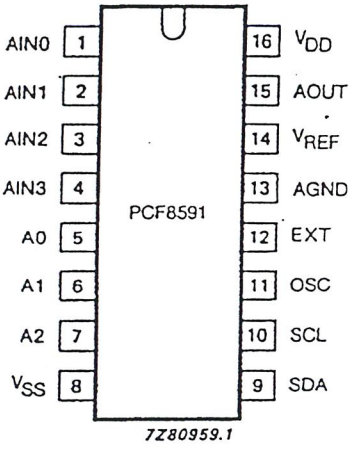


Fig.2 Pinning diagram.

7 FUNCTIONAL DESCRIPTION

7.1 Addressing

Each PCF8591 device in an I²C-bus system is activated by sending a valid address to the device. The address consists of a fixed part and a programmable part. The programmable part must be set according to the address pins A0, A1 and A2. The address always has to be sent as the first byte after the start condition in the I²C-bus protocol. The last bit of the address byte is the read/write-bit which sets the direction of the following data transfer (see Figs 3, 15 and 16).

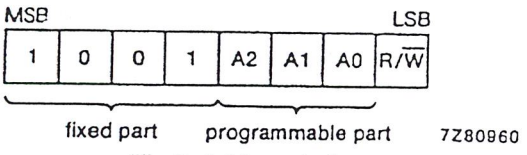


Fig.3 Address byte.

7.2 Control byte

The second byte sent to a PCF8591 device will be stored in its control register and is required to control the device function.

The upper nibble of the control register is used for enabling the analog output, and for programming the analog inputs as single-ended or differential inputs. The lower nibble selects one of the analog input channels defined by the upper nibble (see Fig.4). If the auto-increment flag is set the channel number is incremented automatically after each A/D conversion.

If the auto-increment mode is desired in applications where the internal oscillator is used, the analog output enable flag in the control byte (bit 6) should be set. This allows the internal oscillator to run continuously, thereby preventing conversion errors resulting from oscillator start-up delay. The analog output enable flag may be reset at other times to reduce quiescent power consumption.

The selection of a non-existing input channel results in the highest available channel number being allocated. Therefore, if the auto-increment flag is set, the next selected channel will be always channel 0. The most significant bits of both nibbles are reserved for future functions and have to be set to 0. After a Power-on reset condition all bits of the control register are reset to 0. The D/A converter and the oscillator are disabled for power saving. The analog output is switched to a high-impedance state.

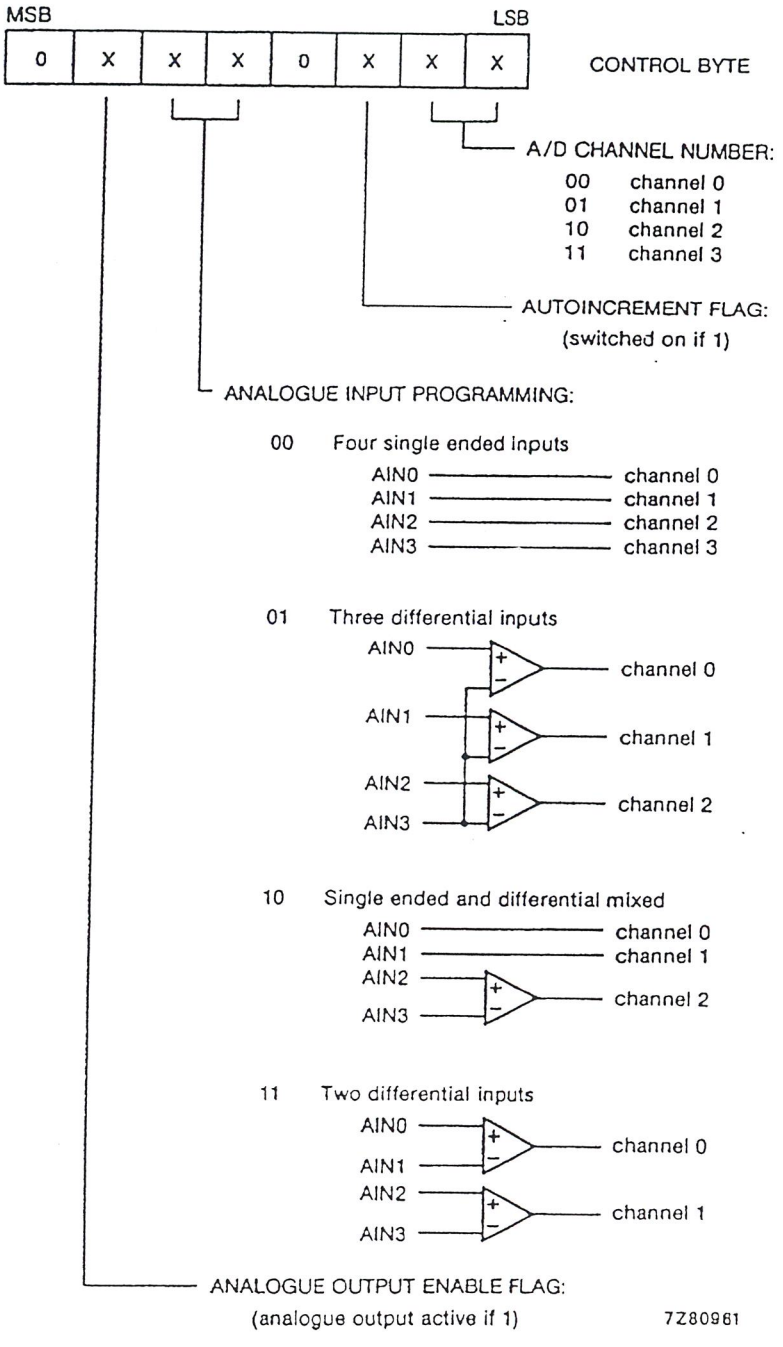


Fig.4 Control byte.

7.3 D/A conversion

The third byte sent to a PCF8591 device is stored in the DAC data register and is converted to the corresponding analog voltage using the on-chip D/A converter. This D/A converter consists of a resistor divider chain connected to the external reference voltage with 256 taps and selection switches. The tap-decoder switches one of these taps to the DAC output line (see Fig.5).

The analog output voltage is buffered by an auto-zeroed unity gain amplifier. This buffer amplifier may be switched on or off by setting the analog output enable flag of the control register. In the active state the output voltage is held until a further data byte is sent.

The on-chip D/A converter is also used for successive approximation A/D conversion. In order to release the DAC for an A/D conversion cycle the unity gain amplifier is equipped with a track and hold circuit. This circuit holds the output voltage while executing the A/D conversion.

The output voltage supplied to the analog output AOUT is given by the formula shown in Fig.6. The waveforms of a D/A conversion sequence are shown in Fig.7.

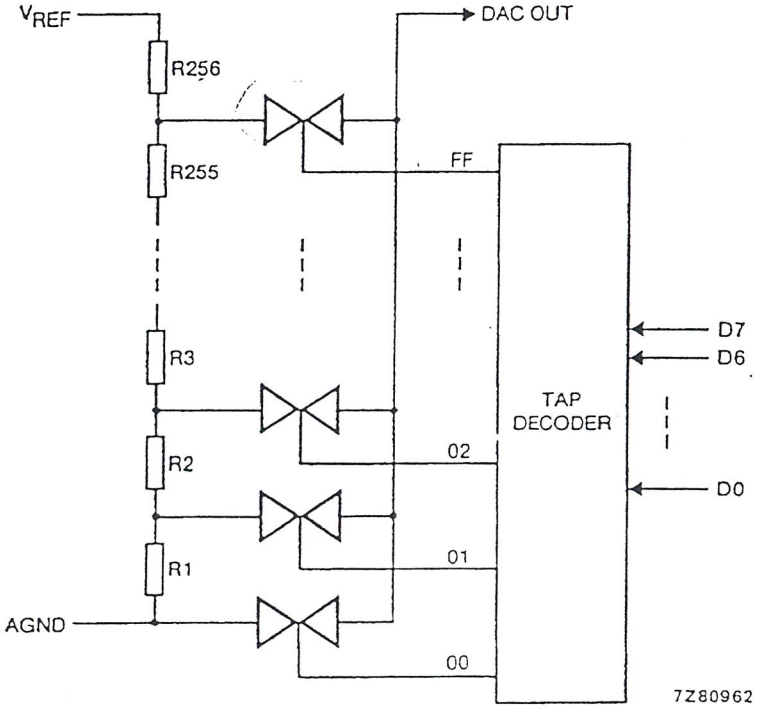


Fig.5 DAC resistor divider chain.

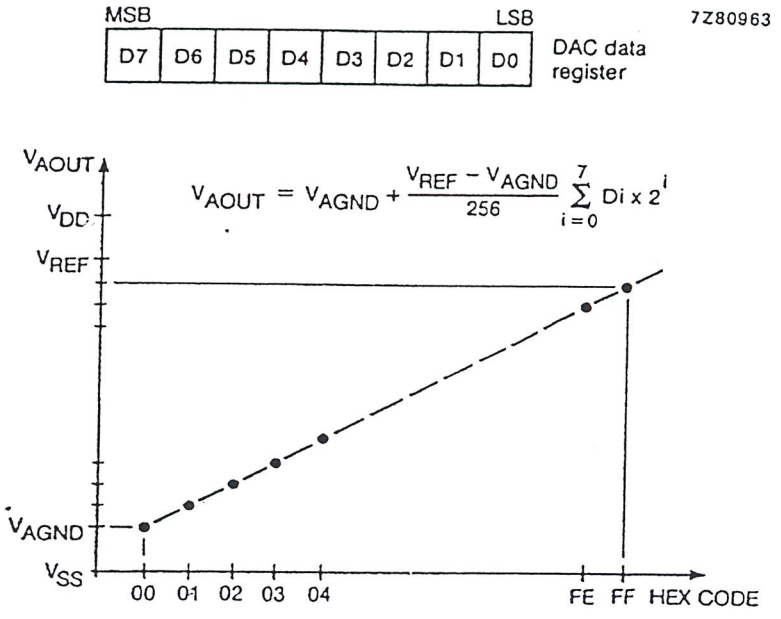


Fig.6 DAC data and DC conversion characteristics.

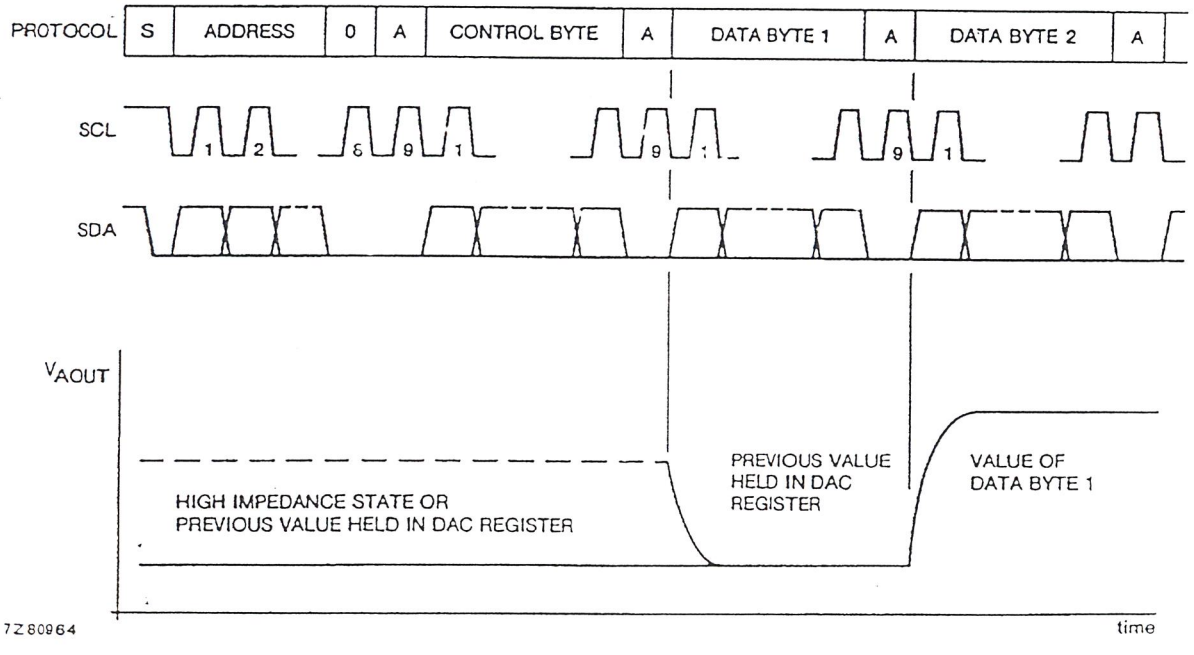


Fig.7 D/A conversion sequence.

7.4 A/D conversion

The A/D converter makes use of the successive approximation conversion technique. The on-chip D/A converter and a high-gain comparator are used temporarily during an A/D conversion cycle.

An A/D conversion cycle is always started after sending a valid read mode address to a PCF8591 device. The A/D conversion cycle is triggered at the trailing edge of the acknowledge clock pulse and is executed while transmitting the result of the previous conversion (see Fig.8).

Once a conversion cycle is triggered an input voltage sample of the selected channel is stored on the chip and is converted to the corresponding 8-bit binary code. Samples picked up from differential inputs are converted to an 8-bit two's complement code (see Figs 9 and 10).

The conversion result is stored in the ADC data register and awaits transmission. If the auto-increment flag is set the next channel is selected.

The first byte transmitted in a read cycle contains the conversion result code of the previous read cycle. After a Power-on reset condition the first byte read is a hexadecimal 80. The protocol of an I²C-bus read cycle is shown in Chapter 8, Figs 15 and 16.

The maximum A/D conversion rate is given by the actual speed of the I²C-bus.

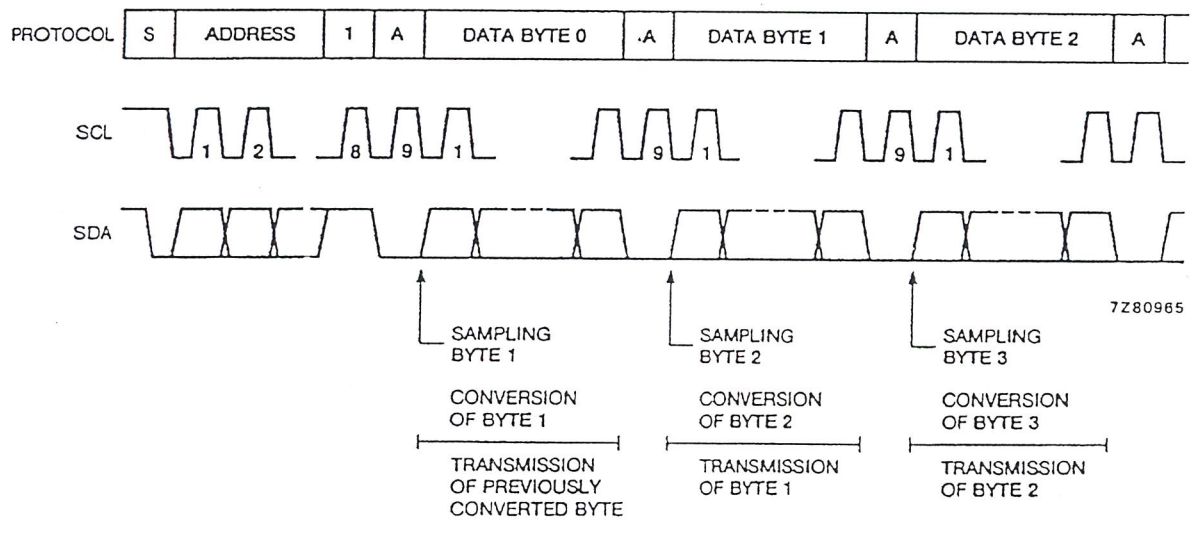


Fig.8 A/D conversion sequence.

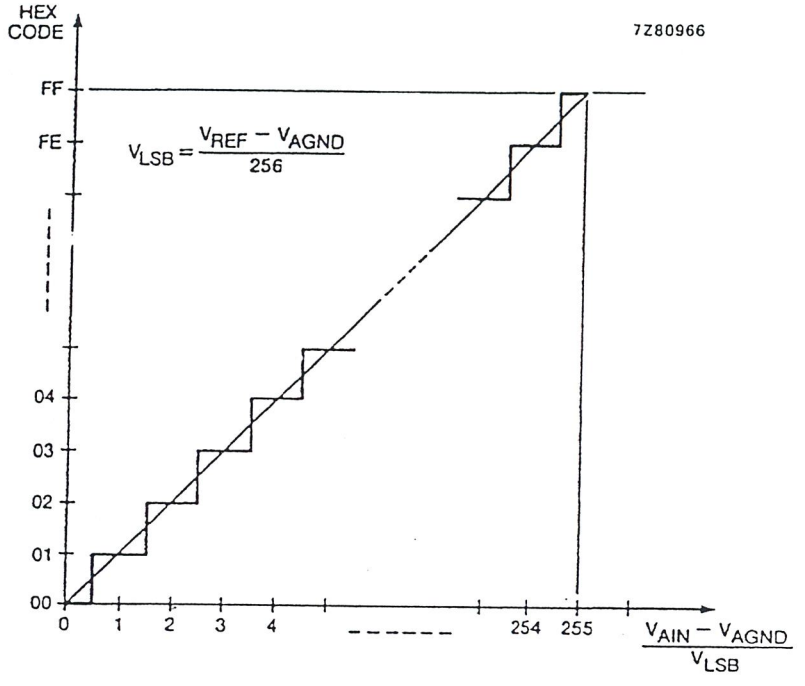


Fig.9 A/D conversion characteristics of single-ended inputs.

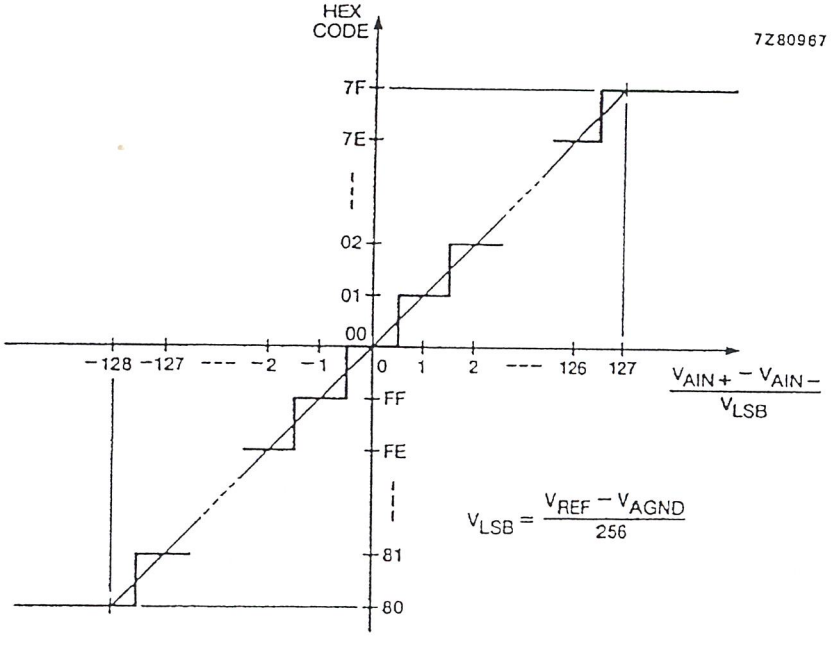


Fig.10 A/D conversion characteristics of differential inputs.

7.5 Reference voltage

For the D/A and A/D conversion either a stable external voltage reference or the supply voltage has to be applied to the resistor divider chain (pins V_{REF} and AGND). The AGND pin has to be connected to the system analog ground and may have a DC off-set with reference to V_{SS}.

A low frequency may be applied to the V_{REF} and AGND pins. This allows the use of the D/A converter as a one-quadrant multiplier; see Chapter 15 and Fig.6.

The A/D converter may also be used as a one or two quadrant analog divider. The analog input voltage is divided by the reference voltage. The result is converted to a binary code. In this application the user has to keep the reference voltage stable during the conversion cycle.

7.6 Oscillator

An on-chip oscillator generates the clock signal required for the A/D conversion cycle and for refreshing the auto-zeroed buffer amplifier. When using this oscillator the EXT pin has to be connected to V_{SS}. At the OSC pin the oscillator frequency is available.

If the EXT pin is connected to V_{DD} the oscillator output OSC is switched to a high-impedance state allowing the user to feed an external clock signal to OSC.

8 CHARACTERISTICS OF THE I²C-BUS

The I²C-bus is for bidirectional, two-line communication between different ICs or modules. The two lines are a serial data line (SDA) and a serial clock line (SCL). Both lines must be connected to a positive supply via a pull-up resistor. Data transfer may be initiated only when the bus is not busy.

8.1 Bit transfer

One data bit is transferred during each clock pulse. The data on the SDA line must remain stable during the HIGH period of the clock pulse as changes in the data line at this time will be interpreted as a control signal.

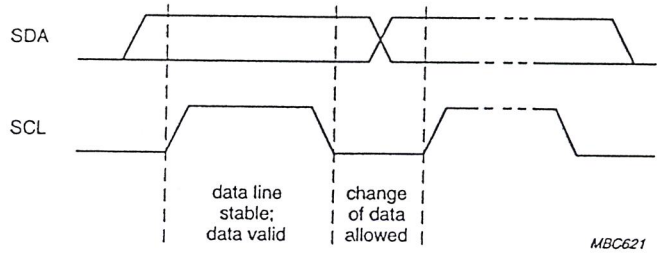


Fig.11 Bit transfer.

8.2 Start and stop conditions

Both data and clock lines remain HIGH when the bus is not busy. A HIGH-to-LOW transition of the data line, while the clock is HIGH, is defined as the start condition (S). A LOW-to-HIGH transition of the data line while the clock is HIGH, is defined as the stop condition (P).

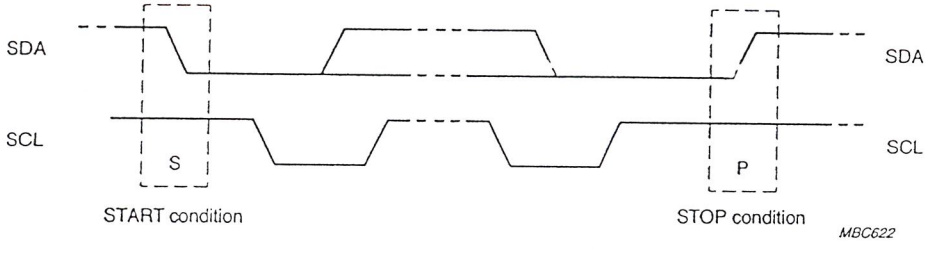


Fig.12 Definition of START and STOP condition.

8.3 System configuration

A device generating a message is a 'transmitter', a device receiving a message is the 'receiver'. The device that controls the message is the 'master' and the devices which are controlled by the master are the 'slaves'.

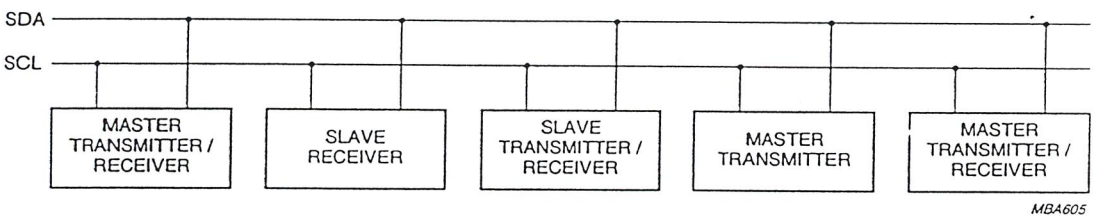


Fig.13 System configuration.

8.4 Acknowledge

The number of data bytes transferred between the start and stop conditions from transmitter to receiver is not limited. Each data byte of eight bits is followed by one acknowledge bit. The acknowledge bit is a HIGH level put on the bus by the transmitter whereas the master also generates an extra acknowledge related clock pulse. A slave receiver which is addressed must generate an acknowledge after the reception of each byte. Also a master must generate an acknowledge after the reception of each byte that has been clocked out of the slave transmitter. The device that acknowledges has to pull down the SDA line during the acknowledge clock pulse, so that the SDA line is stable LOW during the HIGH period of the acknowledge related clock pulse. A master receiver must signal an end of data to the transmitter by **not** generating an acknowledge on the last byte that has been clocked out of the slave. In this event the transmitter must leave the data line HIGH to enable the master to generate a stop condition.

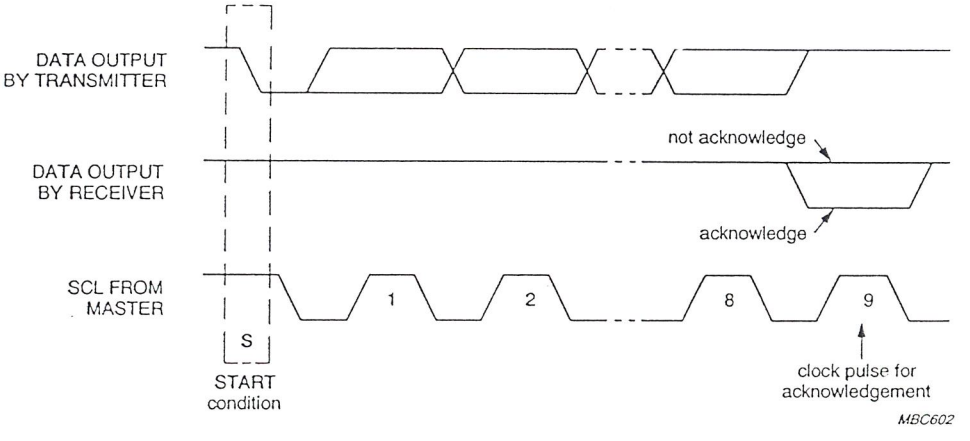


Fig.14 Acknowledgement on the I^2C-bus.

8.5 I²C-bus protocol

After a start condition a valid hardware address has to be sent to a PCF8591 device. The read/write bit defines the direction of the following single or multiple byte data transfer. For the format and the timing of the start condition (S), the stop condition (P) and the acknowledge bit (A) refer to the I²C-bus characteristics. In the write mode a data transfer is terminated by sending either a stop condition or the start condition of the next data transfer.

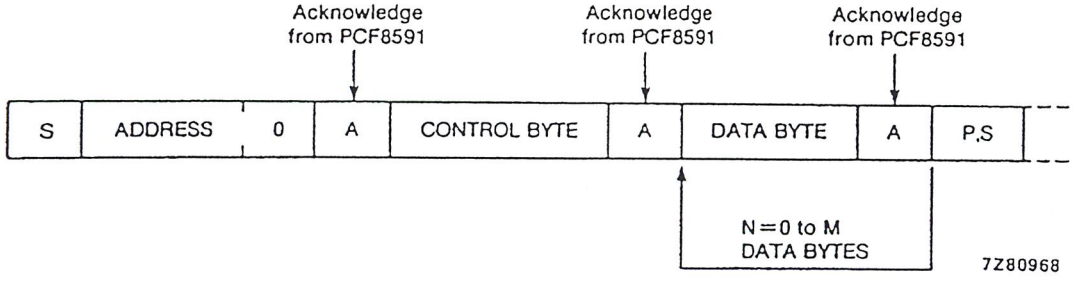


Fig.15 Bus protocol for write mode, D/A conversion.

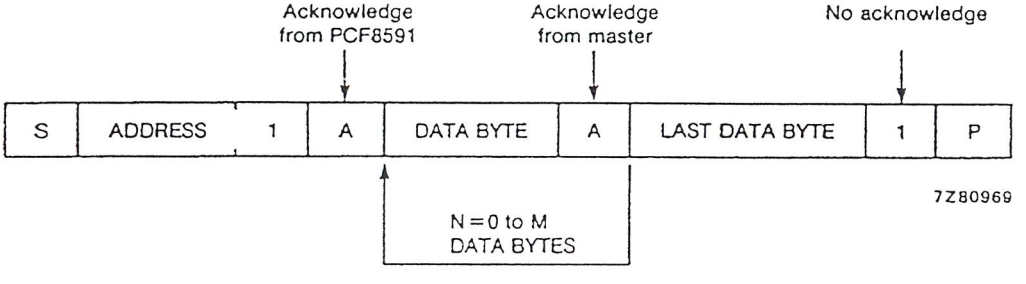


Fig.16 Bus protocol for read mode, A/D conversion.

9 LIMITING VALUES

In accordance with the Absolute Maximum Rating System (IEC 134).

SYMBOL	PARAMETER	MIN.	MAX.	UNIT
V_{DD}	supply voltage (pin 16)	−0.5	+8.0	V
V_I	input voltage (any input)	−0.5	V_{DD} + 0.5	V
I_I	DC input current	–	±10	mA
I_O	DC output current	–	±20	mA
I_{DD}, I_{SS}	V_{DD} or V_{SS} current	–	±50	mA
P_{tot}	total power dissipation per package	–	300	mW
P_O	power dissipation per output	–	100	mW
T_{amb}	operating ambient temperature	−40	+85	°C
T_{stg}	storage temperature	−65	+150	°C

10 HANDLING

Inputs and outputs are protected against electrostatic discharge in normal handling. However, to be totally safe, it is desirable to take precautions appropriate to handling MOS devices. Advice can be found in Data Handbook IC12 under *"Handling MOS Devices"*.

11 DC CHARACTERISTICS

V_{DD} = 2.5 V to 6 V; V_{SS} = 0 V; T_{amb} = −40 °C to +85 °C unless otherwise specified.

SYMBOL	PARAMETER	CONDITIONS	MIN.	TYP.	MAX.	UNIT
Supply						
V_{DD}	supply voltage (operating)		2.5	–	6.0	V
I_{DD}	supply current					
	standby	$V_I = V_{SS}$ or V_{DD}; no load	–	1	15	µA
	operating, AOUT off	f_{SCL} = 100 kHz	–	125	250	µA
	operating, AOUT active	f_{SCL} = 100 kHz	–	0.45	1.0	mA
V_{POR}	Power-on reset level	note 1	0.8	–	2.0	V
Digital inputs/output: SCL, SDA, A0, A1, A2						
V_{IL}	LOW level input voltage		0	–	$0.3 \times V_{DD}$	V
V_{IH}	HIGH level input voltage		$0.7 \times V_{DD}$	–	V_{DD}	V
I_L	leakage current					
	A0, A1, A2	$V_I = V_{SS}$ to V_{DD}	−250	–	+250	nA
	SCL, SDA	$V_I = V_{SS}$ to V_{DD}	−1	–	+1	µA
C_i	input capacitance		–	–	5	pF
I_{OL}	LOW level SDA output current	V_{OL} = 0.4 V	3.0	–	–	mA
Reference voltage inputs						
V_{REF}	reference voltage	$V_{REF} > V_{AGND}$; note 2	V_{SS} + 1.6	–	V_{DD}	V
V_{AGND}	analog ground voltage	$V_{REF} > V_{AGND}$; note 2	V_{SS}	–	V_{DD} − 0.8	V
I_{LI}	input leakage current		−250	–	+250	nA
R_{REF}	input resistance	pins V_{REF} and AGND	–	100	–	kΩ
Oscillator: OSC, EXT						
I_{LI}	input leakage current		–	–	250	nA
f_{OSC}	oscillator frequency		0.75	–	1.25	MHz

Notes

1. The power on reset circuit resets the I²C-bus logic when V_{DD} is less than V_{POR}.
2. A further extension of the range is possible, if the following conditions are fulfilled:

$$\frac{V_{REF} + V_{AGND}}{2} \geq 0.8V, V_{DD} - \frac{V_{REF} + V_{AGND}}{2} \geq 0.4V$$

12 D/A CHARACTERISTICS

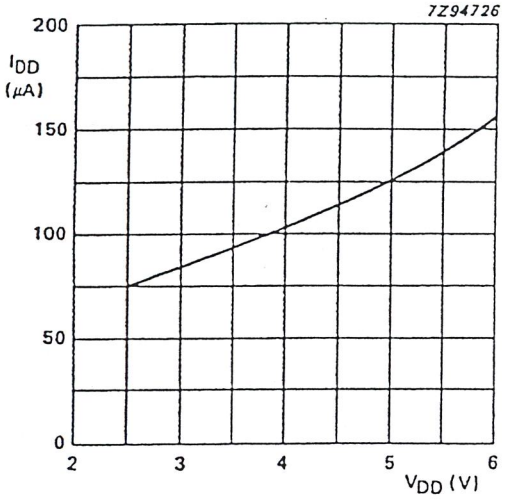
V_{DD} = 5.0 V; V_{SS} = 0 V; V_{REF} = 5.0 V; V_{AGND} = 0 V; R_L = 10 kΩ; C_L = 100 pF; T_{amb} = −40 °C to +85 °C unless otherwise specified.

SYMBOL	PARAMETER	CONDITIONS	MIN.	TYP.	MAX.	UNIT
Analog output						
V_{OA}	output voltage	no resistive load	V_{SS}	–	V_{DD}	V
		R_L = 10 kΩ	V_{SS}	–	0.9 × V_{DD}	V
I_{LO}	output leakage current	AOUT disabled	–	–	250	nA
Accuracy						
OS_e	offset error	T_{amb} = 25 °C	–	–	50	mV
L_e	linearity error		–	–	±1.5	LSB
G_e	gain error	no resistive load	–	–	1	%
t_{DAC}	settling time	to ½ LSB full scale step	–	–	90	µs
f_{DAC}	conversion rate		–	–	11.1	kHz
SNRR	supply noise rejection ratio	f = 100 Hz; V_{DDN} = 0.1 × V_{PP}	–	40	–	dB

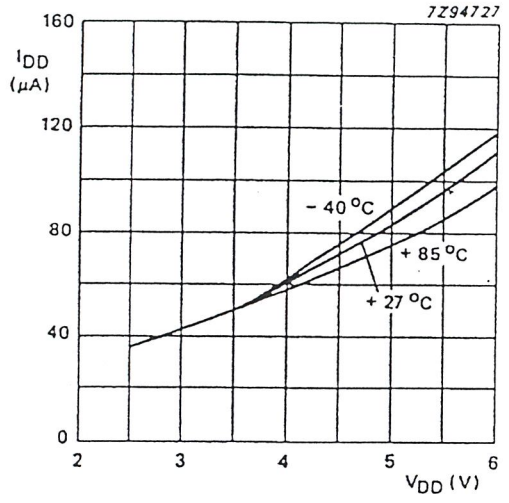
13 A/D CHARACTERISTICS

V_{DD} = 5.0 V; V_{SS} = 0 V; V_{REF} = 5.0 V; V_{AGND} = 0 V; R_S = 10 kΩ; T_{amb} = −40 °C to +85 °C unless otherwise specified.

SYMBOL	PARAMETER	CONDITIONS	MIN.	TYP.	MAX.	UNIT
Analog inputs						
V_{IA}	analog input voltage		V_{SS}	–	V_{DD}	V
I_{LIA}	analog input leakage current		–	–	100	nA
C_{IA}	analog input capacitance		–	10	–	pF
C_{ID}	differential input capacitance		–	10	–	pF
V_{IS}	single-ended voltage	measuring range	V_{AGND}	–	V_{REF}	V
V_{ID}	differential voltage	measuring range; $V_{FS} = V_{REF} - V_{AGND}$	$\frac{-V_{FS}}{2}$	–	$\frac{+V_{FS}}{2}$	V
Accuracy						
OS_e	offset error	T_{amb} = 25 °C	–	–	20	mV
L_e	linearity error		–	–	±1.5	LSB
G_e	gain error		–	–	1	%
GS_e	small-signal gain error	ΔV_i = 16 LSB	–	–	5	%
CMRR	common-mode rejection ratio		–	60	–	dB
SNRR	supply noise rejection ratio	f = 100 Hz; V_{DDN} = 0.1 × V_{PP}	–	40	–	dB
t_{ADC}	conversion time		–	–	90	µs
f_{ADC}	sampling/conversion rate		–	–	11.1	kHz

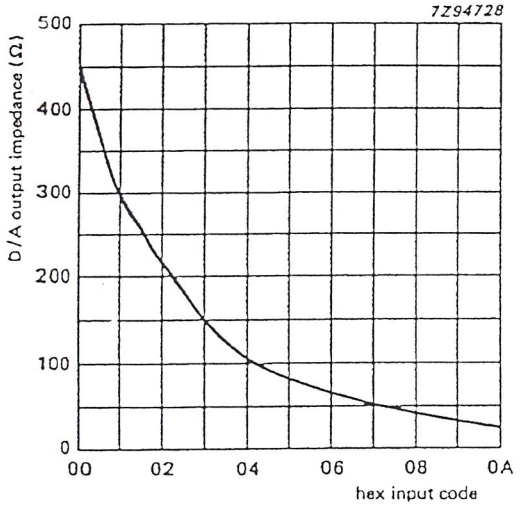


(a) Internal oscillator; T_{amb} = +27 °C.

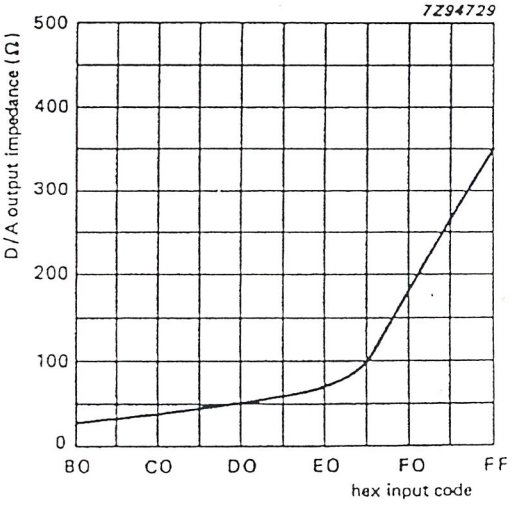


(b) External oscillator.

Fig.17 Operating supply current as a function of supply voltage (analog output disabled).



(a) Output impedance near negative power rail; T_{amb} = +27 °C.



(b) Output impedance near positive power rail; T_{amb} = +27 °C.

The x-axis represents the hex input-code equivalent of the output voltage.

Fig.18 Output impedance of analog output buffer (near power rails).

14 AC CHARACTERISTICS

All timing values are valid within the operating supply voltage and ambient temperature range and reference to V_{IL} and V_{IH} with an input voltage swing of V_{SS} to V_{DD}.

SYMBOL	PARAMETER	MIN.	TYP.	MAX.	UNIT
I²C-bus timing (see Fig.19; note 1)					
f_{SCL}	SCL clock frequency	–	–	100	kHz
t_{SP}	tolerable spike width on bus	–	–	100	ns
t_{BUF}	bus free time	4.7	–	–	µs
$t_{SU;STA}$	START condition set-up time	4.7	–	–	µs
$t_{HD;STA}$	START condition hold time	4.0	–	–	µs
t_{LOW}	SCL LOW time	4.7	–	–	µs
t_{HIGH}	SCL HIGH time	4.0	–	–	µs
t_r	SCL and SDA rise time	–	–	1.0	µs
t_f	SCL and SDA fall time	–	–	0.3	µs
$t_{SU;DAT}$	data set-up time	250	–	–	ns
$t_{HD;DAT}$	data hold time	0	–	–	ns
$t_{VD;DAT}$	SCL LOW-to-data out valid	–	–	3.4	µs
$t_{SU;STO}$	STOP condition set-up time	4.0	–	–	µs

Note

1. A detailed description of the I²C-bus specification, with applications, is given in brochure *"The I²C-bus and how to use it"*. This brochure may be ordered using the code 9398 393 40011.

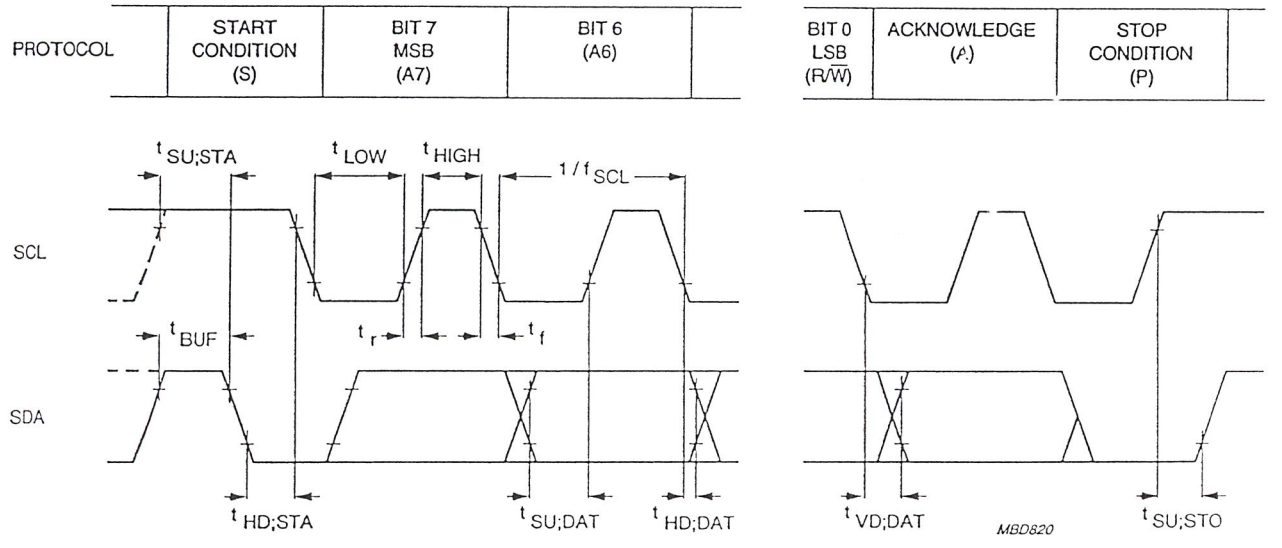


Fig.19 I²C-bus timing diagram; rise and fall times refer to V_{IL} and V_{IH}.

15 APPLICATION INFORMATION

Inputs must be connected to V_{SS} or V_{DD} when not in use. Analog inputs may also be connected to AGND or V_{REF}.

In order to prevent excessive ground and supply noise and to minimize cross-talk of the digital to analog signal paths the user has to design the printed-circuit board layout very carefully. Supply lines common to a PCF8591 device and noisy digital circuits and ground loops should be avoided. Decoupling capacitors (>10 µF) are recommended for power supply and reference voltage inputs.

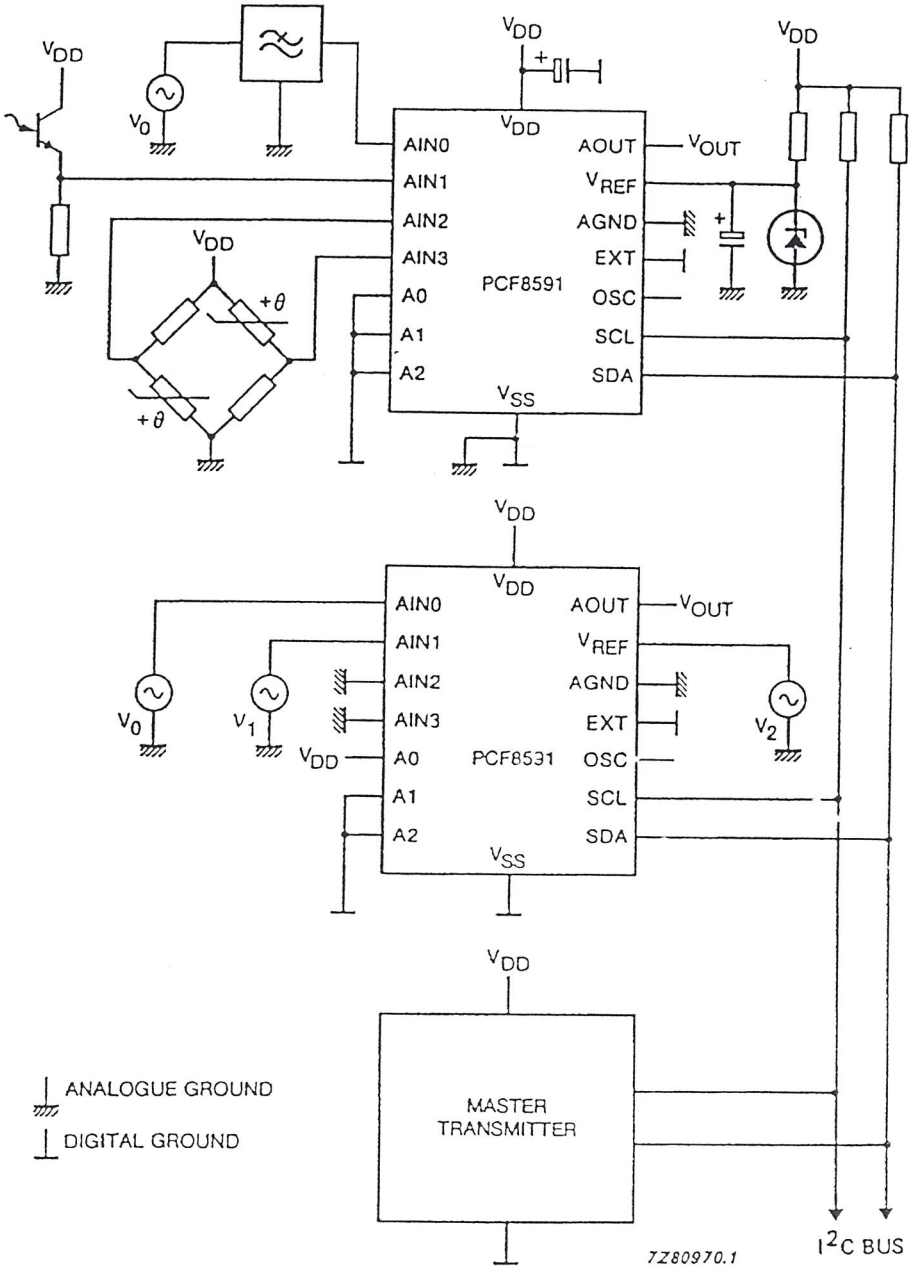
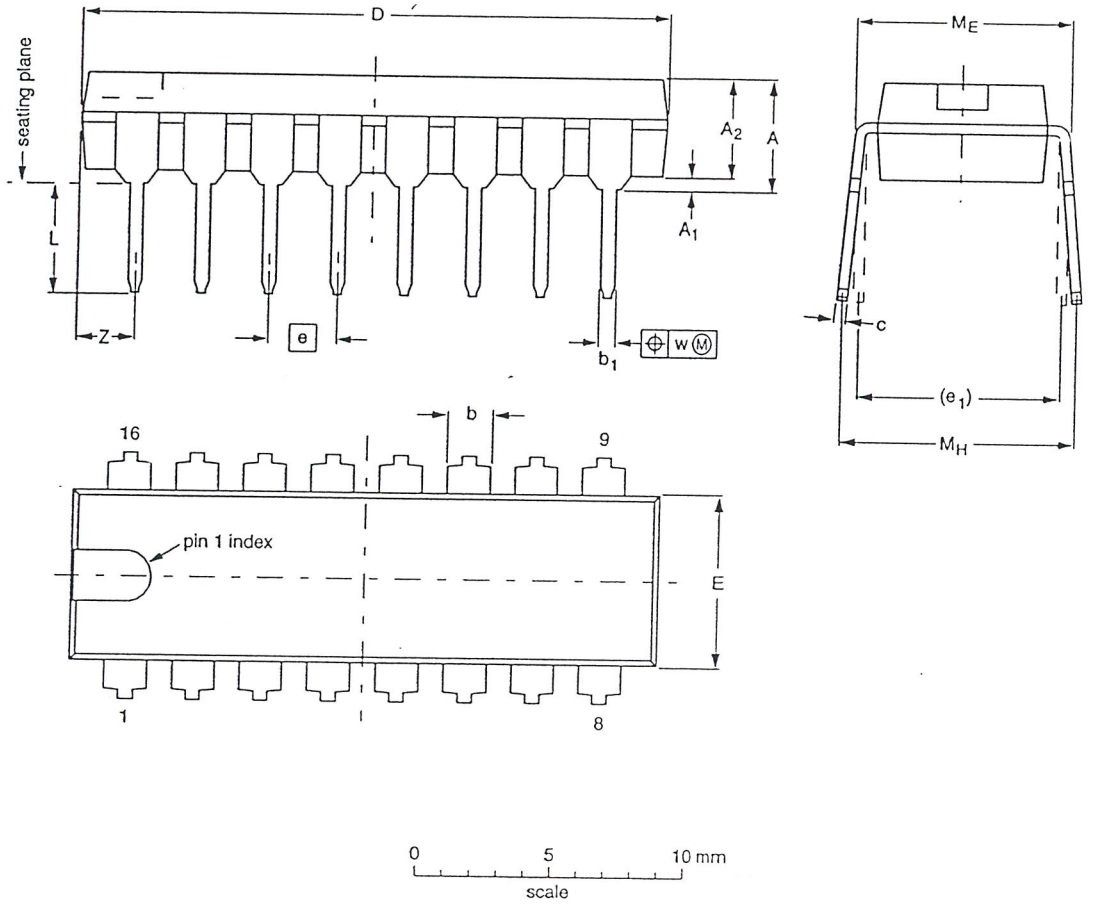


Fig.20 Application diagram.

16 PACKAGE OUTLINES

DIP16: plastic dual in-line package; 16 leads (300 mil); long body

SOT38-1



DIMENSIONS (inch dimensions are derived from the original mm dimensions)

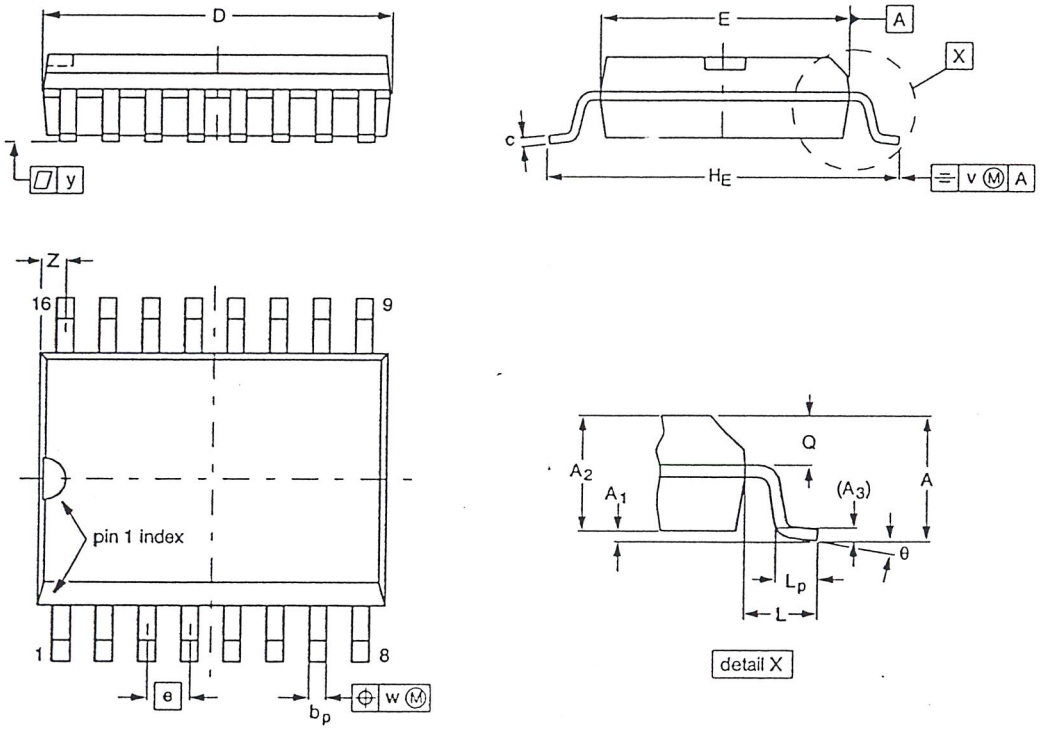
UNIT	A max.	A_1 min.	A_2 max.	b	b_1	c	D (1)	E (1)	e	e_1	L	M_E	M_H	w	Z (1) max.
mm	4.7	0.51	3.7	1.40 1.14	0.53 0.38	0.32 0.23	21.8 21.4	6.48 6.20	2.54	7.62	3.9 3.4	8.25 7.80	9.5 8.3	0.254	2.2
inches	0.19	0.020	0.15	0.055 0.045	0.021 0.015	0.013 0.009	0.86 0.84	0.26 0.24	0.10	0.30	0.15 0.13	0.32 0.31	0.37 0.33	0.01	0.087

Note

1. Plastic or metal protrusions of 0.25 mm maximum per side are not included.

OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	EIAJ			
SOT38-1	050G09	MO-001AE				~~92-10-02~~ 95-01-19

SO16: plastic small outline package; 16 leads; body width 7.5 mm SOT162-1



0 5 10 mm
scale

DIMENSIONS (inch dimensions are derived from the original mm dimensions)

UNIT	A max.	A_1	A_2	A_3	b_p	c	$D^{(1)}$	$E^{(1)}$	e	H_E	L	L_p	Q	v	w	y	$Z^{(1)}$	θ
mm	2.65	0.30 0.10	2.45 2.25	0.25	0.49 0.36	0.32 0.23	10.5 10.1	7.6 7.4	1.27	10.65 10.00	1.4	1.1 0.4	1.1 1.0	0.25	0.25	0.1	0.9 0.4	8° 0°
inches	0.10	0.012 0.004	0.096 0.089	0.01	0.019 0.014	0.013 0.009	0.41 0.40	0.30 0.29	0.050	0.419 0.394	0.055	0.043 0.016	0.043 0.039	0.01	0.01	0.004	0.035 0.016	

Note

1. Plastic or metal protrusions of 0.15 mm maximum per side are not included.

OUTLINE VERSION	REFERENCES				EUROPEAN PROJECTION	ISSUE DATE
	IEC	JEDEC	EIAJ			
SOT162-1	075E03	MS-013AA				~~95-01-24~~ 97-05-22

17 SOLDERING

17.1 Introduction

There is no soldering method that is ideal for all IC packages. Wave soldering is often preferred when through-hole and surface mounted components are mixed on one printed-circuit board. However, wave soldering is not always suitable for surface mounted ICs, or for printed-circuits with high population densities. In these situations reflow soldering is often used.

This text gives a very brief insight to a complex technology. A more in-depth account of soldering ICs can be found in our *"Data Handbook IC26; Integrated Circuit Packages"* (order code 9398 652 90011).

17.2 DIP

17.2.1 Soldering by dipping or by wave

The maximum permissible temperature of the solder is 260 °C; solder at this temperature must not be in contact with the joint for more than 5 seconds. The total contact time of successive solder waves must not exceed 5 seconds.

The device may be mounted up to the seating plane, but the temperature of the plastic body must not exceed the specified maximum storage temperature ($T_{stg\ max}$). If the printed-circuit board has been pre-heated, forced cooling may be necessary immediately after soldering to keep the temperature within the permissible limit.

17.2.2 Repairing soldered joints

Apply a low voltage soldering iron (less than 24 V) to the lead(s) of the package, below the seating plane or not more than 2 mm above it. If the temperature of the soldering iron bit is less than 300 °C it may remain in contact for up to 10 seconds. If the bit temperature is between 300 and 400 °C, contact may be up to 5 seconds.

17.3 SO

17.3.1 Reflow soldering

Reflow soldering techniques are suitable for all SO packages.

Reflow soldering requires solder paste (a suspension of fine solder particles, flux and binding agent) to be applied to the printed-circuit board by screen printing, stencilling or pressure-syringe dispensing before package placement.

Several techniques exist for reflowing; for example, thermal conduction by heated belt. Dwell times vary between 50 and 300 seconds depending on heating method. Typical reflow temperatures range from 215 to 250 °C.

Preheating is necessary to dry the paste and evaporate the binding agent. Preheating duration: 45 minutes at 45 °C.

17.3.2 Wave soldering

Wave soldering techniques can be used for all SO packages if the following conditions are observed:

- A double-wave (a turbulent wave with high upward pressure followed by a smooth laminar wave) soldering technique should be used.
- The longitudinal axis of the package footprint must be parallel to the solder flow.
- The package footprint must incorporate solder thieves at the downstream end.

During placement and before soldering, the package must be fixed with a droplet of adhesive. The adhesive can be applied by screen printing, pin transfer or syringe dispensing. The package can be soldered after the adhesive is cured.

Maximum permissible solder temperature is 260 °C, and maximum duration of package immersion in solder is 10 seconds, if cooled to less than 150 °C within 6 seconds. Typical dwell time is 4 seconds at 250 °C.

A mildly-activated flux will eliminate the need for removal of corrosive residues in most applications.

17.3.3 Repairing soldered joints

Fix the component by first soldering two diagonally-opposite end leads. Use only a low voltage soldering iron (less than 24 V) applied to the flat part of the lead. Contact time must be limited to 10 seconds at up to 300 °C. When using a dedicated tool, all other leads can be soldered in one operation within 2 to 5 seconds between 270 and 320 °C.

กิตติกรรมประกาศ

โครงงานพิเศษฉบับนี้ คุณความดีขอมอบให้แก่บุคคลผู้ให้ความอนุเคราะห์ ตลอดจนแนะนำในด้านต่างๆต่อผู้จัดทำดังนี้

อาจารย์ อาจินต์ น่วมสำราญ อาจารย์ที่ปรึกษา และให้คำแนะนำในการค้นคว้า

อาจารย์ประสิทธิ์ จุลเสรีวงศ์ ที่ให้ข้อมูลเพิ่มเติม

ตลอดจนพี่ ๆ เพื่อนๆผู้ให้คำแนะนำและช่วยเหลือในด้านต่างๆและที่ขาดเสียมิได้ คือบุพการีผู้ให้ความช่วยเหลือในด้านการเงินและกำลังใจด้วยดีตลอดมา

นางสาวกฤษณา เคนหาราช

นายบัณฑิต มาชูวงศ์สกุล

นายสิทธิชัย หวังศิริรุ่งเรือง

ผู้จัดทำ

เอกสารอ้างอิง

1. ธาริน สิทธิธรรมชารี,สุรสิทธิ์ คิวประสพศักดิ์, "คู่มือการเขียนโปรแกรม Advance Visual Basic Version 6.0" ซีเอ็ดยูเคชัน, 354 หน้า
2. ฉัททวุฒิ พืชผล ,พิชิต สันติกุลานนท์, " คู่มือเรียน Visual 6 " , 434 หน้า , 2542
3. รศ. สุเธียร เกียรติสุนทร, ดีซีเอส (Distributed Control Systems), สสท.สมาคมส่งเสริมเทคโนโลยี(ไทย-ญี่ปุ่น), 412 หน้า, 1 มิถุนายน 2542
4. Serial port Interfacing Starter book, Inex
5. B.C.Kuo, " Automatic Control System," Seventh Edittion, Prentice-Hall International Edition, 1995.
6. R.C.Dorf and R.H. Bishop " Modern Control System,"Addison Wesley, Seventh Edition, 1995.
7. N.E. Leonard and W.S.Levine, "Second Edition, Addison Wesley, 1995.